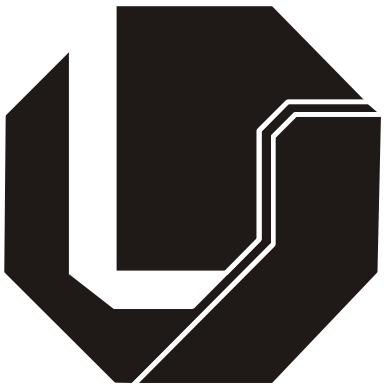
UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA ELÉTRICA
PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA



Um Sistema para Transposição Automática de Seqüências MIDI baseada em Alcance Vocal

Sandra Fernandes de Oliveira Lima

Fevereiro
2006

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA ELÉTRICA
PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Um Sistema para Transposição Automática de Seqüências MIDI baseada em Alcance Vocal

Sandra Fernandes de Oliveira Lima*

Texto da dissertação apresentada à Universidade Federal de Uberlândia, perante a banca de examinadores abaixo, como parte dos requisitos necessários para a obtenção do título de Mestre em Ciências. Aprovada em 22 de fevereiro de 2006.

Banca Examinadora:

Prof. Adriano Alves Pereira, Dr. - Orientador (UFU)

Prof. Alcimar Barbosa Soares, PhD (UFU)

Prof. Alessandro Barros Wellesley, Dr. (UEMG)

Prof. Keije Yamanaka, PhD (UFU)

* A bolsa de estudo para esta pesquisa foi concedida pela CAPES, Brasil.

FICHA CATALOGRÁFICA

Elaborada pelo Sistema de Bibliotecas da UFU / Setor de Catalogação e Classificação

L732s Lima, Sandra Fernandes de Oliveira.
Um sistema para transposição automática de seqüências MIDI baseada em alcance vocal / Sandra Fernandes de Oliveira Lima. - Uberlândia, 2005.
233f. : il.
Orientador: Adriano Alves Pereira.
Dissertação (mestrado) – Universidade Federal de Uberlândia, Programa de Pós-Graduação em Engenharia Elétrica.
Inclui bibliografia.
1. Engenharia elétrica - Teses. 2. Sistema MIDI - Teses. I. Pereira, Adriano Alves. II. Universidade Federal de Uberlândia. Programa de Pós-Graduação em Engenharia Elétrica. III. Título.

CDU:621.3

Um Sistema para Transposição Automática de Seqüências MIDI baseada em Alcance Vocal

Sandra Fernandes de Oliveira Lima*

Texto da dissertação apresentada à Universidade Federal de Uberlândia, como parte dos requisitos para a obtenção do título de Mestre em Ciências.

Prof. Adriano Alves Pereira, Dr.	Prof. Darizon Alves de Andrade, Ph.D.
Orientador	Coordenador do curso de Pós-Graduação

* A bolsa de estudo para esta pesquisa foi concedida pela CAPES, Brasil.

Agradecimentos

Agradeço primeiramente a Deus, Senhor de todas as coisas, detentor de todo o meu conhecimento, de todas as minhas certezas e ideais. Sem Ele nada seria possível.

Agradeço ao meu marido Luciano, meu amor, meu companheiro, meu suporte, meu amigo. Tudo fez para que esse trabalho se concretizasse e esteve sempre ao meu lado nos meus momentos de mau-humor. Você é minha luz, não se esqueça. Eu não me esquecerei de tudo o que fez e faz por mim.

Agradeço aos meus filhos, luzes do meu viver. Obrigado por me apoiarem e também por suportarem os meus momentos ruins. Vocês são pessoas especiais e ocupam um lugar enorme em meu coração.

Agradeço ao meu orientador Adriano, pessoa amiga, compreensiva, interessada e competente. Conviver com você, vale a pena!

Agradeço à minha grande amiga Priscilla: pessoa maravilhosa que me incentivou a cursar o mestrado e sempre esteve comigo em todos os momentos. Esquecer de você: Impossível!!!

Agradeço aos meus pais (Marcílio e Clarice) e irmãos (Viviana, Marlice, Eliane e Sérgio) que sempre estiveram ao meu lado e sempre acreditaram em mim.
Mamãe: pena você não estar mais aqui! Eu sei a admiração que você sempre teve pelo meu trabalho. Papai: que bom você estar aqui! Eu sei que você também sempre admirou os trabalhos que realizei.

Agradeço à minha segunda família: Júnia, Vinícius e Lucília pelo amor, dedicação e apoio que sempre me deram. Agradeço também a todos os meus sobrinhos: Vocês são muito importantes para mim.

Agradeço de uma maneira especial ao Hélcio. Você sempre foi um amigão! Não tenho palavras. Você fez muito por mim.

Agradeço à Lena, à Lolô, à Bel, à Cida, à Andréia. Grandes amigas!!! Obrigada pelo apoio!

Agradeço à Príscila pela sua amizade e pelos desenhos que ilustraram maravilhosamente esta dissertação.

Agradeço enfim, a todos os meus companheiros de laboratório e a todos que estiveram ao meu lado.

Agradeço à Marly. Você é um anjo!

Resumo

LIMA, Sandra Fernandes de Oliveira. **Um Sistema para Transposição Automática de Seqüências MIDI baseada em Alcance Vocal.** Uberlândia: FEELT-UFU, 2006. 233 p.

Este trabalho apresenta uma solução para se transpor seqüências musicais MIDI codificadas em SMF (Standard MIDI Files) em formato 0 ou formato 1, de uma forma automática, aplicada em todas as trilhas e canais MIDI da música, preservando a formatação e conteúdos registrados nos arquivos originais. Esta transposição é baseada em um range vocal explicitado pelo cantor e pela escolha do canal MIDI na qual está registrada a melodia que se deseja cantar. Como fruto desta pesquisa, gerou-se um aplicativo denominado de BestVocal, o qual, além de possuir esta ferramenta de transposição baseada em range vocal de conforto do cantor, possui, também, várias ferramentas adicionais. Dentre elas pode-se citar: uma ferramenta para eliminar pausas iniciais do arquivo MIDI, outra para visualizar os instrumentos e respectivos ranges, outra para salvar e imprimir a letra da música existente no arquivo e outra para modificar (substituir) os instrumentos MIDI de qualquer canal ou trilha da seqüência. Este trabalho apresenta, também, ganhos na preservação da saúde do cantor no tocante ao seu aparelho fonador, já que, cantando apenas na sua região de conforto, o mesmo evitará calos (nódulos), pólipos e outras doenças oriundas do esforço inadequado de suas pregas (cordas) vocais. Um outro ganho obtido pelo cantor, com tal sistema, reside no fato de que cantando apenas em regiões onde sua voz possui boas características timbrais, evita-se, assim, que o mesmo desafine ou que não consiga emitir algumas

notas da seqüência, o que manterá sua credibilidade como profissional do canto (para seu público alvo).

Palavras-chave: MIDI, SMF, transposição, desafinar, range vocal, range vocal de conforto, tessitura, classificação vocal, aparelho fonador, pregas vocais, timbre.

Abstract

LIMA, Sandra Fernandes de Oliveira. **An Autonomus MIDI Sequences Transposition System driven by Vocal Comfort Zone**. Uberlândia: FEELT-UFU, 2006. 233 p.

The author of this work presents a solution to transpose MIDI musical sequences codified in SMF (Standard MIDI File) in format 0 or format 1. This transposition is accomplished automatically, being applied in all the tracks and music MIDI channels. This computational tool preserves the format and contents registered in the original files. This transposition is based on the singer vocal range (comfort zone) and by the choice of the melody MIDI channel that he wants to sing. As a result of this research, an application named BestVocal was created, by incorporate the transposition tool based in the comfort vocal range of the singer and incorporate a tool to eliminate pauses of the MIDI file too. Another inclusion is given by a tool that allows it to visualize, save and print the existent lyric of the present file. The Best Vocal allows the user to modify the MIDI instruments of any channel too. Using these computational tools, the singer will avoid having nodules, polyps and other diseases originating from the inadequate effort of his vocal fold. Another earning obtained by the singer, with such system, resides in the fact that singing only in the vocal range where your voice sounds good, will avoid, to him, sing out of tune, avoiding, too, muting notes out of your vocal range. So, he will maintain his credibility as a good professional to his audience.

Key-words: MIDI, SMF, transposition, vocal range, comfort zone, vocal type, phonetics system, vocal fold, tone, out of key, pitch, untuned.

Publicações

Durante o desenvolvimento dos trabalhos, momento este antecedente o ingresso definitivo no programa de mestrado como aluna regular, participei do mesmo como aluna em disciplina isolada e aluna especial, tendo tido a oportunidade de participar de várias pesquisas na FEELT, de onde surgiram alguns frutos do trabalho científico em que atuei, bem como algumas publicações técnicas, principalmente em uma área que despertou bastante meu interesse: a computação musical. Segue abaixo uma lista de algumas das publicações, onde não foram citados os CDROMS multimídia interativos aplicados à educação, os artigos anteriores a 2003, os últimos 19 artigos publicados em revista internacional nível C CAPES e os softwares desenvolvidos paralelamente com este trabalho de mestrado.

LIVROS

1. LIMA, Luciano Vieira; LIMA; Sandra Fernandes de Oliveira; MACHADO, André Campos; PINTO, Marília Mazzaro.**Computação Musical - Arranjo Automático, Conversão de CD em MIDI e MIDI em Áudio Utilizando o Band-in-a-Box 12.0a, Virtual Sound Canvas 3.23 e Akoff Music Composer 2.0**. 1ª. ed. São Paulo: Editora Érica Ltda, 2004.

2. LIMA, Luciano Vieira; LIMA, Sandra Fernandes de Oliveira; MACHADO, André Campos; PINTO, Marília Mazzaro. **Computação Musical - Sound Forge 8.0 - Gravação ao Vivo, Restauração de Sons de LPs e Masterização Áudio Digital**. 1ª. ed. São Paulo: Editora Érica Ltda, 2005.

3. LIMA, Sandra Fernandes de Oliveira; LIMA, Luciano Vieira; MACHADO, André Campos. **Sound Forge 6.0 - Restauração de LPs e Gravação de CDs.** São Paulo: Editora Érica Ltda, [2003 ou 2004].

Trabalhos completos em congressos e eventos

1. LIMA, Sandra Fernandes de Oliveira; RUFINO, Hugo Leonardo Pereira; LOPES, Guilherme Francisco; GOULART, Reane Franco; LIMA, Luciano Vieira. **MATEMÁTICA APLICADA À DETECÇÃO DE TESSITURA VOCAL E À TRANSPOSIÇÃO AUTOMÁTICA DE MÚSICAS EM ARQUIVOS MIDI PADRÃO**. In: CONGRESSO NACIONAL DE MATEMÁTICA APLICADA E COMPUTACIONAL, São José do Rio Preto, v. 1, p. 10-17.

2. LIMA, Sandra Fernandes de Oliveira; LIMA, Luciano Vieira; LOPES, Guilherme Francisco; NAKAMOTO, Paula Teixeira; RUFFINO, Hugo Leonardo Pereira. **PROCESSAMENTO DIGITAL DE SINAIS E TÉCNICAS DE COMPILADORES APLICADAS AO RECONHECIMENTO DE TESSITURA VOCAL E TRANSPOSIÇÃO AUTOMÁTICA DE MÚSICA**. In: CEEL - CONFERÊNCIA DE ESTUDOS DE ENGENHARIA ELÉTRICA - UFU, Uberlândia. CEEL,v. 1, p. 10-16.

3. LIMA, Sandra F. O.; SILVA, Priscilla T. L.; LIMA, Luciano V.; PEREIRA, Adriano A. **COMPUTATION SYSTEMS APPLIED TO THE TEACHING OF MUSIC FOR THE DEFICIENT VISUAL AND CHILDREN BY USING NUMEROFONIA AND A NEW PROPOSED BRAILLE CODIFICATION.** to GCETE'2005.

4. LIMA, Sandra F. O.; LIMA, Luciano Vieira; SOUZA, Ana Paula Rodrigues; PEREIRA, Adriano Alves. **RECOGNITION OF VOCAL RANGE FOR AUTOMATIC TRANSPOSITION OF MUSIC APPLIED TO THE MUSICAL DOMAIN, HEALTH AND TEACHING** to GCETE'2005.

5. CAMARGO, Hélcio Jr.; LIMA, Luciano Vieira; PINHEIRO Alan Petrônio; LIMA Sandra Fernandes de Oliveira; PEREIRA, Adriano Alves. **TÉCNICAS DE CONSTRUÇÃO DE COMPILADORES APLICADAS À MANIPULAÇÃO NÃO DESTRUTIVA DO PROTOCOLO MIDI PARA GERAÇÃO DE ARQUIVOS SMF MULTISEQÜENCIAIS, COMPATÍVEIS COM OS FORMATOS GM, GS E XG, SEM A PERDA DA ESTRUTURA E EVENTOS INDIVIDUAIS ORIGINAIS** - WCCSETE'2006

6. LIMA, Sandra Fernandes de Oliveira; LIMA, Luciano Vieira Lima; SOARES, Alcimar Barbosa; PAIVA, Lílian Ribeiro Mendes. **SISTEMA DE APOIO AO ENSINO E APRENDIZADO DE GEOMETRIA EUCLIDIANA E NUMERAÇÃO POSICIONAL UTILIZANDO TÉCNICAS DE PIAGET.** In: CEEL - CONFERÊNCIA DE ESTUDOS DE ENGENHARIA ELÉTRICA - UFU, Uberlândia. CEEL, v. 1, p. 20-27.

7. LIMA, Sandra Fernandes de Oliveira; LIMA, Luciano Vieira; CRUZ, Willer Alves da Silva; PEREIRA, Antônio Eduardo Costa. **DESENVOLVIMENTO DE ALGORITMOS DE ALTO DESEMPENHO PARA APLICAÇÕES EM INTERNET COMPILADAS UTILIZANDO A LINGUAGEM PLIANT.** In: SEMINÁRIO DE EDUCAÇÃO À DISTÂNCIA, Uberlândia, v. 1, p. 12-20.

8. CURY, Reny Filho; LIMA, Luciano Vieira; LIMA, Sandra Fernandes de Oliveira; MOURA, Éder Alves. **A PLATFORM OF HIGH PERFORMANCE FOR E-LEARNING IN COMPILED LANGUAGE PLIANT** to GCETE'2005.

9. LIMA, Luciano Vieira Lima; CAMARGO, Helcio Jr; LIMA, Sandra Fernandes de Oliveira; PEREIRA, Adriano Alves; PINHEIRO, Alan Petrônio. **A METHOD FOR PREPARING EXPERTS IN COMPUTER ENGINEERING SUBJECTS -** WCCSETE'2006.

Artigos completos publicados em revista internacional

1. LIMA, Sandra Fernandes de Oliveira; PEREIRA, Adriano Alves; LIMA, Luciano Vieira; MACHADO, André Campos. **BEST VOCAL 2004: DESAFINAR?...NUNCA MAIS. CONVERTA SUAS SEQUÊNCIAS MIDI DE ACORDO COM SEU ALCANCE VOCAL.** Playmusic, São Paulo, v.88, p. 7-10, ISSN/ISBN: 14151871.

2. LIMA, Sandra Fernandes de Oliveira; PEREIRA, Adriano Alves; LIMA, Luciano Vieira; MACHADO, André Campos. **DETERMINANDO SEU ALCANCE VOCAL DE CONFORTO E ADEQUANDO SUAS SEQUÊNCIAS MDI PARA ACOMPANHAR SEU GRUPO VOCAL – Parte I.** Playmusic, São Paulo, v.89, p. 14-17, ISSN/ISBN: 14151871.

3. LIMA, Sandra Fernandes de Oliveira; PEREIRA, Adriano Alves; LIMA, Luciano Vieira; MACHADO, André Campos. **DETERMINANDO SEU ALCANCE VOCAL DE CONFORTO E ADEQUANDO SUAS SEQUÊNCIAS MDI PARA ACOMPANHAR SEU GRUPO VOCAL – Parte II.** Playmusic, São Paulo, v.92, p. 8-11, ISSN/ISBN: 14151871.

4. LIMA, Sandra Fernandes de Oliveira; PEREIRA, Adriano Alves; LIMA, Luciano Vieira; MACHADO, André Campos. **DETERMINANDO SEU ALCANCE VOCAL DE CONFORTO E ADEQUANDO SUAS SEQUÊNCIAS MDI PARA ACOMPANHAR SEU GRUPO VOCAL – Parte II.** Playmusic, São Paulo, v.92, p. 8-11, ISSN/ISBN: 14151871.

5. LIMA, Sandra Fernandes de Oliveira; PEREIRA, Adriano Alves; LIMA, Luciano Vieira; MACHADO, André Campos. **ELIMINANDO PAUSAS INICIAIS E FINAL DE ARQUIVOS MIDI INSTANTANEAMENTE – BEST REST 2005.** Playmusic, São Paulo, v.90, p. 14-17, ISSN/ISBN: 14151871.

6. LIMA, Sandra Fernandes de Oliveira; PEREIRA, Adriano Alves; LIMA, Luciano Vieira; MACHADO, André Campos. **AFINANDO SUA VOZ AUTOMATICAMENTE DE ACORDO COM SUA SEQUÊNCIA MIDI E QUEIMANDO SEU CD DE ÁUDIO – BAND-IN-A-BOX.** Playmusic, São Paulo, v.91, p. 14-17, ISSN/ISBN: 14151871.

7. LIMA, Sandra Fernandes de Oliveira; LIMA, Luciano Vieira; MACHADO, André Campos; MENEZES, Carlos R.F. **SONAR 5: AS NOVIDADES.** Playmusic, São Paulo, v.97, p. 8-10, ISSN/ISBN: 14151871.

8. LIMA, Sandra Fernandes de Oliveira; LIMA, Luciano Vieira; CAMARGO, Hélcio Jr.; PEREIRA, Adriano Alves; PINHEIRO, Alan Petrônio. **BEST MERGE 2005 – COLOQUE VÁRIAS MÚSICAS EM UM SÓ ARQUIVO MIDI – Parte II-** .Playmusic, São Paulo, v.96, p. 7-9, ISSN/ISBN: 14151871.

9. LIMA, Sandra Fernandes de Oliveira; LIMA, Luciano Vieira; PEREIRA, Adriano Alves; MACHADO, André Campos. **NO SYSEX 2005 – ELIMINANDO MENSAGENS EXCLUSIVS DE SISTEMA E ATIVANDO O MODO GM DE SEUS ARQUIVOS MIDI PROBLEMÁTICOS**. Playmusic, São Paulo, v.96, ISSN/ISBN: 14151871.

10. LIMA, Sandra Fernandes de Oliveira; LIMA, Luciano Vieira; VIEIRA, Júnia Helena; MACHADO, André Campos. **BAND-IN-A-BOX 12 - AS NOVIDADES DA NOVA VERSÃO.** Playmusic, São Paulo, v.63, p. 7-9, ISSN/ISBN: 14151871.

11. LIMA, Sandra Fernandes de Oliveira; LIMA, Luciano Vieira; CAMARGO JR, Hélcio; MACHADO, André Campos. **COMO CONVERTER PARA PARTITURA O QUE ESTAMOS CANTANDO OU TOCANDO AO MICROFONE DO COMPUTADOR? - AUTOSCORE.** Playmusic, São Paulo, v. 64, p. 16-18, ISSN/ISBN: 14151871.

12. LIMA, Sandra Fernandes de Oliveira; LIMA, Luciano Vieira; CAMARGO JR, Hélcio; MACHADO, André Campos. **COMO CONVERTER PARA PARTITURA O QUE ESTAMOS CANTANDO OU TOCANDO AO MICROFONE DO COMPUTADOR? - AUTOSCORE - PARTE II.** Playmusic, São Paulo, v. 65, p. 68-71, ISSN/ISBN: 14151871.

13. LIMA, Sandra Fernandes de Oliveira; LIMA, Luciano Vieira; PINTO, Marília Mazzaro; MACHADO, André Campos. **COMPUTER MUSIC - FINALE 2003 - AS NOVIDADES DA NOVA VERSÃO - PARTE II**. Playmusic, São Paulo, v. 65, p. 7-9, ISSN/ISBN: 14151871.

14. LIMA, Sandra Fernandes de Oliveira; LIMA, Luciano Vieira; MACHADO, André Campos; PINTO, Marilia Mazzaro. **COMPUTER MUSIC - HARMONIZAÇÃO AUTOMÁTICA DE MELODIA UTLIZANDO O PLUG-IN DO FINALE 2003: BAND-IN-A-BOX AUTO-HARMONIZER**. Zoological Journal Of The Linnean Society, São Paulo - SP, v. 66, n. 1, p. 10-12, ISSN/ISBN: 14151871.

15. LIMA, Sandra Fernandes de Oliveira; LIMA, Luciano Vieira; MACHADO, André Campos; CAMARGO JR., Hélcio. **CONVERSOR POLIFONICO DE ÁUDIO PARA MIDI - AMAZINGMIDI.** Playmusic, São Paulo, v. 66, n. 1, p. 15-18, ISSN/ISBN: 14151871.

16. LIMA, Sandra Fernandes de Oliveira; MACHADO, André Campos; SILVA, Carlos Alberto Lopes da; LIMA, Luciano Vieira. **COPYSCAT E VOCAL REMOVER - PROGRAMAS PARA REMOVER O VOCAL DE UMA MÚSICA**. Playmusic, São Paulo, v. 67, n. 1, p. 10-12, ISSN/ISBN:00141518.

17. LIMA, Sandra Fernandes de Oliveira; LIMA, Luciano Vieira; MACHADO, André Campos; PINTO, Marília Mazzaro. **FINALE 2003 - AS NOVIDADES DA NOVA VERSÃO**. Playmusic, São Paulo, v. 64, p.7-9, ISSN/ISBN: 14151871.

18. LIMA, Sandra Fernandes de Oliveira; LIMA, Luciano Vieira; MACHADO, André Campos; PINTO, Marília Mazzaro. **HARMONIZAÇÃO AUTOMÁTICA DE MELODIA COM O BAND-IN-A-BOX AUTO-HARMONIZER UTILIZANDO O FINALE 2003.** Playmusic, SÃO PAULO, v. 66, ISSN/ISBN: 14151871.

19. LIMA, Sandra Fernandes de Oliveira; VIEIRA, Júnia Helena; LIMA, Luciano Vieira; MACHADO, André Campos. **Interpretação automática de acordes e impressão de partituras com melodias e cifras - Band-in-a-box 11 - Parte III.** Playmusic, São Paulo, v. 63, p. 16-18, ISSN/ISBN: 14151871.

20. LIMA, Sandra Fernandes de Oliveira; SILVA, Carlos Roberto Lopes da; LIMA, Luciano Vieira. **PLUG-INS VST E DIRECTX.** Playmusic, São Paulo, v. 67, n. 1, p. 68-71, ISSN/ISBN: 00141518.

21. LIMA, Sandra Fernandes de Oliveira; VIEIRA, Júnia Helena; LIMA, Luciano Vieira MACHADO, André Campos. **BAND-IN-A-BOX 11 - INTERPRETAÇÃO AUTOMÁTICA DE ACORDES E IMPRESSÃO DE PARTITURAS COM MELODIAS E CIFRAS**. Playmusic, São Paulo, v. 62, p. 70-71, ISSN/ISBN: 14151871.

22. LIMA, Sandra Fernandes de Oliveira; LIMA, Luciano Vieira; VIEIRA, Júnia Helena; CAMARGO JR, Hélcio; MACHADO, André Campos. **BAND-IN-A-BOX VERSÃO 11**. Playmusic, São Paulo, v. 52, p. 7-9, ISSN/ISBN: 14151871.

Conteúdo

Capítulo 1

Capítulo 2

Capítulo 3

Capítulo 4

Capítulo 5

Capítulo 6

Lista de diagramas

Capítulo 2

Capítulo 4

Lista de figuras

Capítulo 1

CAPÍTULO 2

Capítulo 3

Capítulo 4

Capítulo 5

Lista de tabelas

Capítulo 2

Capítulo 3

Capítulo 4

Capítulo 1

Introdução

A matemática, a música e outras áreas do conhecimento, desde Platão, Sócrates e tantos outros, nasceram juntas como ciência, e, até hoje, vários desenvolvimentos e ferramentas técnicas [1] têm sido concretizados para resolverem problemas comuns a estes domínios, tais como: processos estocásticos, Wavelets [2], transformada de Fourier [3][4], Sistemas inteligentes de composição musical [5][6][7][8], modelagem do conhecimento humano[5][6][7][8] e outros mais. Mesmo em sistemas de educação e aprendizado, verifica-se que o cérebro humano, a cada grupo de pessoas, aprende de forma personalizada (GARDNER, 1994), conforme o tipo de inteligência de cada um [9], e, dentre estas inteligências, a inteligência musical é uma das que mais se destaca, mesmo quando o tema a ser ministrado é exclusivamente na área da matemática e da lógica [10].

Além do domínio musical interagir com outras áreas do conhecimento, o mesmo apresenta problemas que também carecem de soluções tecnológicas interdisciplinares que respondam e apresentem soluções exclusivas ao mesmo.

Como exemplo de tais problemas, pode-se apresentar um que possui grande relevância para os cantores, principalmente os populares: definir com precisão se um determinado cantor conseguirá ou não executar um determinado repertório musical, sem desafinar ou

comprometer a qualidade timbral[1] de sua *performance*.

A princípio pode parecer uma questão simples de ser respondida ou determinada através de experimentações, mas, quando observada com mais propriedade, nota-se ser a mesma de extrema complexidade, principalmente devido ao aparelho fonador humano variar continuamente de *performance* e range[2] de acordo com o clima, tensão emocional, estresse, alimentação e outros fatores mais.

Outro problema técnico se dá devido ao fato de que poucos músicos (principalmente os amadores) conhecem a fundo a teoria musical, sendo difícil, para os mesmos, determinar o melhor range e tessitura[3] para executar uma música em um determinado instante. Este problema se agrava quando se percebe o quão difícil é encontrar profissionais competentes que possam realizar esta tarefa com precisão.

Por outro lado, implementar sistemas especialistas e sistemas inteligentes para trabalhar com timbres e arquivos sonoros é uma tarefa complexa que demanda do programador um conhecimento matemático e musical avançado, bem como conhecimento sedimentado em técnicas computacionais de análise de sinais digitalizados[4]. Novamente, encontrar profissionais com tais características não é uma tarefa simples, já que poucos se dedicam a esta formação multidisciplinar.

Pesquisando soluções para tal problema, identifica-se a existência de alguns sistemas computacionais (programas de computador) que auxiliam o músico a realizar a tarefa de transpor[5] suas melodias para determinadas tonalidades[6] musicais nas quais um cantor se

[1] Timbre – conjunto de parâmetros do som que caracterizam e diferenciam uma fonte sonora de outra. É através do mesmo que podemos distinguir o tipo de instrumento ou cantor que está executando uma determinada melodia. O mesmo é composto de uma envoltória (**A**ttack **D**elay **S**ustain **R**elease) de uma freqüência fundamental e suas parciais harmônicas.
[2] Range – extensão vocal - faixa de freqüência do espectro audível produzida por um determinado instrumento. Faixa de freqüência audível percebida por uma pessoa ou equipamento.
[3] Tessitura – região vocal de conforto – faixa do espectro audível onde o som é emitido com qualidade, sem distorção.
[4] Digitalizar – converter um sinal analógico (produzido por um elemento físico real) em um sinal digital. Um sinal digitalizado é uma representação discreta, ou seja, representada por alguns pontos do sinal, cujos valores são convertidos em notação binária.
[5] Transpor – alterar a faixa de freqüência das notas musicais de uma determinada música (mudar a tonalidade)
[6] Tonalidade – faixa do espectro sonoro representada por um conjunto de notas musicais que combinam, que soam agradavelmente quando tocadas em seqüência. Para que este conjunto de notas pertença a uma mesma tonalidade, estas deverão obedecer a um conjunto de regras musicais específicas.

julga capaz de executar com conforto e qualidade timbral. Dentre os sistemas existentes, pode-se citar: Cakewalk [11], Band-in-a-Box[12], Finale[13], Sibelius[14], Encore [15] e outros mais.

Infelizmente tais sistemas não fazem transposições baseadas no range do cantor, e sim conforme o músico solicitar. Se o mesmo solicitar uma transposição inadequada, fora de seu range, o sistema simplesmente o fará, sem qualquer questionamento.

Assim, o domínio musical carece de sistemas especialistas autônomos, alguns com certo grau de inteligência, que atendam uma gama maior de usuários que possuam pouco ou nenhum conhecimento de teoria e técnicas musicais.

1.1 Justificativas

Um sistema que identifique a tessitura do cantor (*vocal range*), a partir do som discretizado e digitalizado, transpondo-a automaticamente para sua tonalidade de conforto vocal (tessitura), conforme sua classificação vocal (*vocal type*), seria uma ferramenta relevante tanto para quem canta quanto para quem escuta. A implementação de um sistema que apresente uma solução computacional especialista para a resolução deste problema, seria de grande valia para o domínio musical, para o ensino de música, para a saúde do próprio cantor e para o deleite do ouvinte.

Assim, justifica-se este trabalho na inexistência de um sistema especialista, inteligente ou não, que identifique e classifique automaticamente e dinamicamente o range e a tessitura de um cantor, bem como o range de cada instrumento de uma música. Através desta classificação seria viável a implementação de um sistema que determine qual a melhor tonalidade para execução de uma determinada música que traga um maior conforto ao cantor e satisfação ao ouvinte.

Um sistema com tais características auxiliaria a evitar que o cantor venha a ser acometido de lesões em suas pregas vocais[7], traria uma maior satisfação ao ouvinte, ao evitar que o cantor desafine, e, também, minimizaria e otimizaria o trabalho do arranjador ou maestro na formação de seus grupos vocais e instrumentais.

[7] Pregas vocais – não tecnicamente denominadas de Cordas vocais – órgão responsável pela emissão de freqüências audíveis do sistema fonador

A seguir, são apresentados exemplos de lesões causadas nas pregas vocais do ser humano.

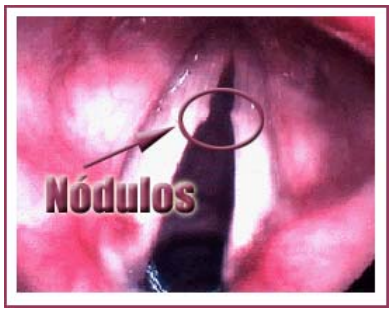


Figura 1.1 *– Nódulos*

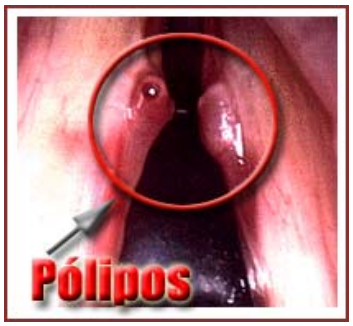


Figura 1.2 *- Pólipos*

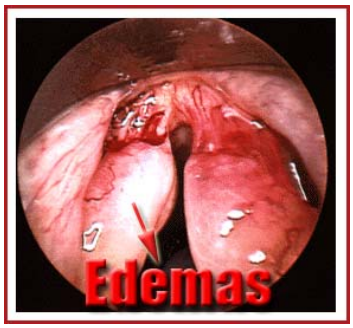


Figura 1.3 *- Edemas*

Lesões orgânicas causadas pelo uso excessivo da voz. Os Edemas são removidos apenas cirurgicamente sendo depois seguidos de reabilitação fonoaudiológica

A identificação, automática ou não, da tonalidade para se cantar uma música, a qualquer instante, não é uma tarefa simples de ser executada, seja por um especialista humano, seja por um sistema computacional especialista.

Conforme já comentado, implementar um sistema de reconhecimento de tessitura a partir de sinais analógicos digitalizados é uma tarefa que demanda um especialista com competência em várias áreas e domínios do conhecimento, tais como matemática, física, música, engenharia de som, computação e linguística. Uma solução, para este problema, que não necessite de um conhecimento tão vasto e multidisciplinar, seria de grande valia para o domínio musical, a qual permitiria que novos desenvolvimentos e aplicações se tornem possíveis e passíveis de serem implementadas por músicos relativamente fluentes na arte da programação.
Para tanto, uma solução emergente seria trabalhar com o paradigma musical abrangido pela tecnologia MIDI[8] [11]. Esta tecnologia, prescinde o profissional de possuir conhecimentos profundos (ou nenhum) na área de análise de sinais digitalizados.

No padrão MIDI (ver detalhes no capítulo 2), as informações musicais, tais como: nota (frequência musical), timbre, andamento e outros parâmetros utilizados em uma música, estão explicitamente registrados no protocolo de transmissão de dados, bem como nos arquivos gerados em formatos também padronizados. Cabe ao sistema especialista

[8]MIDI – Musical Instrument Digital Interface – um protocolo padrão utilizado para representação e comunicação de mensagens musicais entre equipamentos digitais.

apenas manipular estas informações através de um modelo do conhecimento humano existente na área afim, evitando erros de interpretação e de análise. Tais erros são comumentemente existentes na análise convencional de sinais digitalizados devido às limitações das ferramentas matemáticas existentes para tal finalidade.

O grande problema a ser enfrentado na utilização deste paradigma é a falta de informações específicas de como extrair com segurança e precisão todos estes parâmetros deste protocolo e seus arquivos, ficando muitas das informações relegadas apenas a desenvolvedores de equipamentos e softwares proprietários, os quais inserem mensagens personalizadas (exclusivas) nos pacotes MIDI, de forma que somente os equipamentos de sua fabricação tenham um pleno acesso às informações ali contidas.

A despeito desta aparente “barreira”, justifica-se o ingresso nos desenvolvimentos utilizando o protocolo MIDI, por possuir a Universidade Federal de Uberlândia, na Faculdade de Engenharia Elétrica, um grupo de pesquisadores com várias publicações, pesquisa e desenvolvimentos nesta área. Este grupo possui um bom domínio do protocolo MIDI e das características exclusivas de cada fabricante de equipamento e softwares do mercado, permitindo que se possa implementar qualquer aplicativo no domínio musical utilizando o paradigma MIDI, transformando inseguraça e expectativas em certezas e confiabilidade.

O conhecimento musical existente e necessário para se resolver tal problema, aliado ao suporte tecnológico de um paradigma que permite a implementação de sistemas especialistas de análise no domínio musical, em especial análise de range e tessitura vocal, justificam a viabilidade de se propor tal projeto de pesquisa, o qual é objeto desta dissertação.

1.2 Objetivos gerais

O objetivo geral deste trabalho iniciado é o projeto e implementação de um sistema de determinação de análise de range e tessitura vocal de conforto a partir de qualquer mídia

sonora, tais como: arquivos Wave, MIDI, Mp3[9] ou direto de uma fonte sonora através de transdutores[10] (microfones, captadores, outros).

A meta final seria projetar e implementar um sistema que conseguisse determinar, em tempo real, o range vocal de um cantor e sua classificação, transpondo automaticamente o repertório musical a ser executado em determinado instante, ou, em casos extremos, eliminando as músicas que não fossem passíveis de execução pelo respectivo cantor.

1.3 Objetivo específico

Projetar e implementar um sistema computacional, um software, que permita a análise assistida de range e classificação vocal, tanto do cantor quanto dos instrumentos existentes em uma sequência musical MIDI, bem como a transposição automática do range dos instrumentos da música analisada para a tonalidade de conforto do cantor em questão, centrada no range da melodia a ser executada pelo cantor. A tonalidade de conforto a ser adotada será aquela que fique centrada no range de cada cantor, conforme ilustrado nas figuras 1.4 e 1.5.

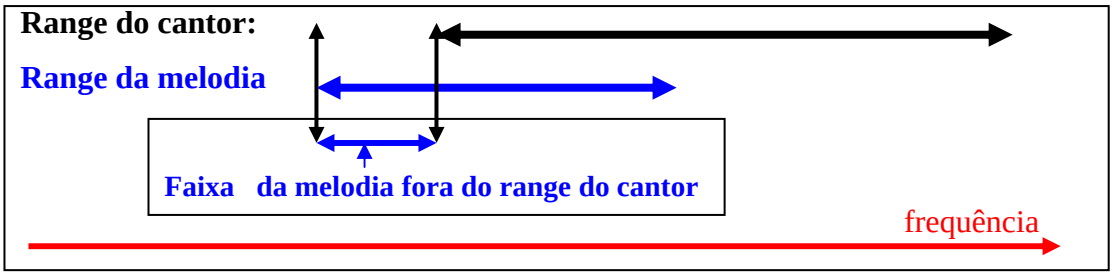


Figura 1.4 *– Range do cantor e range da melodia a ser interpretada*

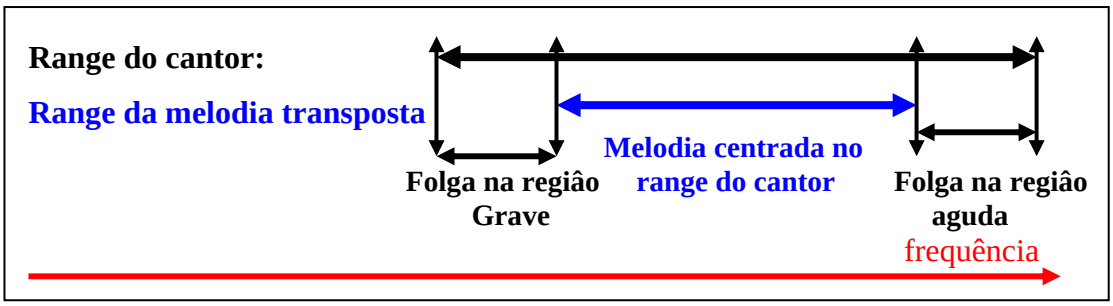


Figura 1.5 *– Range do cantor e range da melodia transposta*

[9]Mp3 – formato de arquivo contendo uma informação musical compactada através de algoritmos específicos.

[10] Transdutor – elemento capaz de converter um sinal proveniente de um dispositivo ou fenômeno físico em um sinal elétrico equivalente (exemplo: a agulha de um toca-discos antigo – transforma a pressão nos sulcos dos discos em um sinal elétrico equivalente para excitação de um alto-falante - um outro transdutor)

A análise da tessitura vocal, ou seja, da região de conforto dentro do range de um cantor onde o mesmo produza sons de qualidade sonora, sem distorção, é feita através de fonetogramas [16][17] por pesquisadores em fonoaudiologia. Os fonetogramas demandam análise de sinais digitalizados no domínio do tempo e da frequência, utilizando, para tanto, transformadas e técnicas matemáticas complexas. Os mesmos apresentam algumas restrições quanto à precisão e fidelidade dos resultados obtidos, devido tanto à dificuldade de utilização das ferramentas matemáticas aplicadas a sinais digitalizados, bem como à deficiência na utilização correta dos conceitos matemáticos e físicos envolvidos no processo.

1.4 Metas

Para cumprir os objetivos específicos traçados, as seguintes ações foram realizadas:

- Estudo da teoria e conceitos sobre sinais, ruídos e do processo de digitalização de sinais sonoros.
- Estudo dos formatos e protocolos de registro dos sinais sonoros.
- Estudo do protocolo MIDI, hardware e software, bem como dos formatos de arquivos para registro de músicas sequenciadas.
- Projeto e implementação de um sistema de transposição automática de sequências musicais MIDI de acordo com a tonalidade de conforto de um determinado cantor.
- Um estudo de caso prático utilizando um usuário profissional da música sequenciada para validação do sistema gerado.
- Conclusão pela confrontação dos resultados e estudos de casos obtidos, com os objetivos propostos inicialmente.
- Levantamento e identificação de propostas para trabalhos futuros.

1.5 Estrutura da dissertação

- No Capítulo 1 são apresentadas a Introdução, as Justificativas, os Objetivos gerais e específicos, as Metas e a Estrutura desta dissertação, onde são abordados os problemas a serem enfrentados na identificação e classificação

vocal de um determinado indivíduo, bem como apresentar o trabalho, sua relevância, potencialidades e restrições.

- No Capítulo 2 são apresentados os conceitos de sinais analógicos e digitais, a definição do que vem a ser o formato Wave e detalhes do protocolo MIDI.

- No Capítulo 3 são apresentados conceitos e teoria musical básica, focando nos métodos e tipos de classificação vocal.

- No Capítulo 4 é apresentado o projeto e implementação do sistema de transposição automática de sequências musicais MIDI baseada na tessitura de um determinado cantor.

- No Capítulo 5 são apresentados dois estudos de casos, real e simulado, empregando as ferramentas implementadas nesta dissertação, as quais deram origem ao software denominado por Best Vocal 2005. Também são apresentados alguns programas que realizam algumas das potencialidades do sistema proposto nesta dissertação.

- No Capítulo 6 é apresentada a Conclusão final do trabalho, comparando as metas e objetivos propostos com os resultados obtidos. No mesmo também são apresentadas sugestões de trabalhos futuros para melhoria do sistema existente, bem como para implantação de novas potencialidades.

Capítulo 2

Sinais, formatos e protocolos de som

Neste capítulo serão abordadas as diferenças entre sinal analógico e digital, o princípio básico de conversão e digitalização destes sinais, nível DC, relação sinal ruído, decibéis, formato Wave de arquivos de sinais digitalizados e sobre o protocolo de comunicação MIDI e seus formatos de arquivos para registro de seqüências MIDI.

2.1 Sinal, Sinal Analógico e Sinal Digital

Antes de conceituar sinal analógico e sinal digital, é necessário abordar alguns conceitos sobre sinal, ruído e relação sinal ruído.

Sinal: Um sinal é composto por uma forma de onda produzida por um elemento físico em um meio de propagação qualquer. No caso da música, estar-se-á tratando de sinais produzidos por instrumentos musicais ou sistema fonador humano.

Ruído: No caso de sinais musicais, é todo som que esteja incluído no som musical que não pertença à informação (à música) que se deseja transmitir ou capturar.

Relação Sinal Ruído: É a razão entre o nível de pressão sonora do sinal que se deseja transmitir ou aquisicionar em relação ao nível de pressão sonora dos demais sinais

expúrios à informação musical (o ruído). Quanto maior esta relação, melhor será a qualidade do sinal emitido ou capturado.

2.1.1 Sinal analógico

Um sinal analógico, no caso de uma onda sonora, é um sinal que possui características semelhantes ao objeto físico que as produziu. Assim, no caso de um cantor, as cordas vocais vibram com um determinado movimento, provocando uma vibração, um movimento do ar ao ser redor, o qual produz uma onda cujo perfil possui o mesmo aspecto do movimento que a produziu. Diz-se, assim, que esta onda é análoga ao corpo que a produziu. Daí o nome: sinal analógico. A figura 2.1 ilustra um sinal analógico.

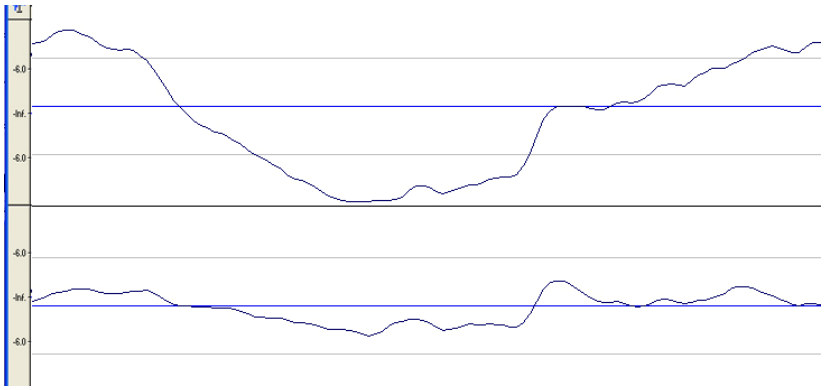


Figura 2.1 *– Sinal analógico*

2.1.2 Sinal digital

O computador não possui uma memória infinita, e, portanto, pode guardar uma certa quantidade finita de informações. Esta característica faz com que seja impossível o armazenamento de um sinal analógico no mesmo, já que no processo de digitalização do sinal apenas o valor do sinal em um determinado tempo poderá ser armazenado. Como entre dois pontos quaisquer pode-se obter infinitos pontos, está fora de cogitação o armazenamento completo do sinal. Assim, um sinal digital é uma representação discreta, um conjunto de amostras, finita, de alguns pontos do sinal analógico original, cujos valores são armazenados em notação binária. A figura 2.2 ilustra um sinal analógico apresentando os pontos amostrados do mesmo que comporão o sinal digital.

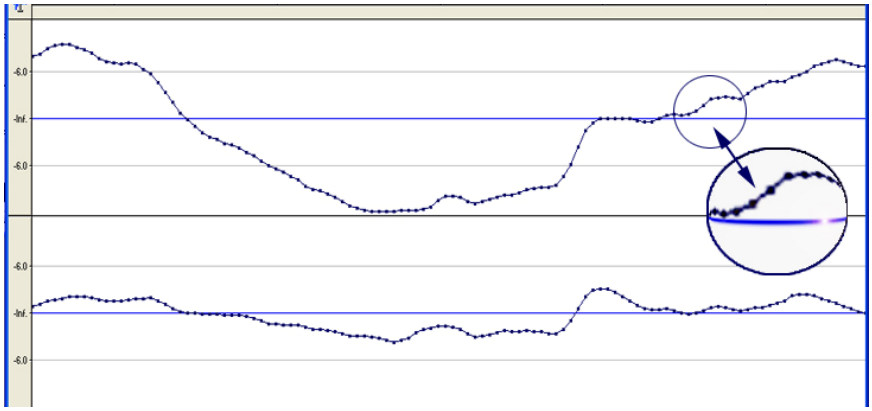


***Figura 2.2** – Sinal analógico com os pontos de amostragem que serão digitalizados*

Obs. Os princípios básicos do processo de conversão A/D (analógico para digital) e dos processos de amostragem, quantização e digitalização de um sinal sonoro musical podem ser vistos com detalhes nas dissertações de mestrado de MACHADO [19] e de LOPES [22] defendidas na Universidade Federal de Uberlândia – FEELT.

2.2 Formato Wave

Como existem vários processos de amostragem de um sinal para posterior armazenamento em memória de computador, bem como os mesmos podem ser amostrados com resoluções diferentes, variando taxa de amostragem, número de bits de quantização, tipo de armazenagem (estéreo ou mono), tipo de compressão e outras características mais [18][19], foram criados vários formatos padrões de armazenamento para garantir uma portabilidade e confiabilidade na recuperação destes sinais por qualquer tipo de sistema computacional ou equipamento multimídia.

Entre estes formatos, o mais utilizado e difundido é o formato Wave [18]. Conforme a faixa audível do sinal que se deseja registrar, adota-se uma taxa de amostragem que obedeça ao teorema da amostragem de Nyquist [20].

Conforme a complexidade da forma de onda (o número de parciais harmônicas com respectivas amplitudes) e sua relevância no registro timbral do som, varia-se o número de bits para representar cada ponto amostrado.

Estas características são registradas no formato Wave em um cabeçalho padrão, permitindo que o sinal seja recuperado e analisado com precisão pelas ferramentas matemáticas dedicadas a este fim.

Assim, o formato Wave possui em seu cabeçalho os seguintes parâmetros:

- **tamanho do arquivo em bytes**: Número de bytes de dados do arquivo, incluindo o cabeçalho.
- **Número de bits por amostra**: indica quantos bits serão utilizados para representar cada ponto do sinal de áudio amostrado. O número de bits é fundamental para a qualidade da reprodução do sinal digitalizado. Quando o número de bits é baixo, pode-se acrescentar ruídos de digitalização na reprodução, ou seja, a inserção de parciais harmônicas de alta freqüência no sinal (ruído de quantização), provocando a sensação da existência de ruídos (chiados) no áudio.
- **Número de canais**: informa ao usuário quantos canais simultâneos de áudio existem no arquivo. Um valor igual a 2 indica que o arquivo é estéreo. Valor igual a 1 indica um arquivo mono, e, atualmente, pode-se ter valores superiores a 2 para sistemas quadrafônicos, sistemas para *home theather* e outras aplicações mais.
- **Taxa de amostragem**: a taxa de amostragem é definida para capturar o sinal com uma qualidade ótima para o ouvido humano, e, neste caso, a mesma não precisa exceder o valor de 44.100 Hz, já que o limite da audição humana está em torno de 22.000 Hz, obedecendo os critérios definidos por Nyquist. Por outro lado, o sistema de reprodução do áudio também limita esta taxa de amostragem. No caso de sons trafegando em linhas telefônicas com pares de fios, a taxa de amostragem é em torno de 11.025 Hz. No caso de transmissão via rádio, a taxa é de 22.050 Hz.

2.2.1 Cabeçalho do arquivo WAVE

O arquivo Wave inicia por um cabeçalho de 36 Bytes, sendo:

- 4 (quatro) Bytes contendo a string “RIFF” (caracteres 52 49 46 46 em Hexadecimal).
- 4 (quatro) Bytes contendo o tamaho total do arquivo.
- 4 (quatro) Bytes contendo a string “WAVEfmt” (caracteres 57 41 56 45 66 6D 74 20 em Hexadecimal).
- 2 (dois) Bytes para estrutura, como, por exemplo: PCM (Pulse Code Modulation).
- 2 (dois) Bytes para número de canais.
- 4 (quatro) Bytes para a taxa de amostragem.
- 4 (quatro) Bytes para a taxa média de transferência de dados.
- 2 (dois) Bytes para representar a quantidade mínima de bytes utilizados para representar uma ponto amostrado: 8 bits mono: 01, 8 bits estéreo: 02, 16 bits mono: 02, 16 bits estéreo: 04.
- 2 (dois) Bytes para o número de bits por amostra. Oito bits = 08, dezesseis bits = 16.
- 4 (quatro) Bytes para representar a string “data”, caracteres 64 61 74 61 em hexadecimal.
- 4 (quatro) Bytes para registrar o número de Bytes de dados (pontos digitalizados) a serem lidos.

Obs. Quando uma informação possui mais de um Byte no arquivo Wave, o primeiro Byte da informação é o Byte mais significativo.

Conforme mencionado no Capítulo 1, este trabalho de dissertação não utilizará o arquivo Wave ou seus derivados para análise de uma música. Esta escolha se deu devido à complexidade das ferramentas matemáticas de análise e da imprecisão dos resultados obtidos pelas mesmas quando não são observados os parâmetros ótimos de aplicação de cada uma, tais como: a aquisição de períodos completos de sinais e uma taxa de amostragem dinâmica que garanta uma aquisição de múltiplos $\mathbf{2^n}$ pontos por período.

2.3 A Idealização do MIDI

Nos anos 80, do século passado, houve um crescente interesse por música eletrônica, surgindo inúmeros equipamentos musicais que interligavam entre si para gerar novos tipos de sons, permitir *performances* originais e permitir que um músico apenas pudesse realizar, ao mesmo tempo, diversas tarefas que demandariam um grupo bem maior. Para tanto, exigiam-se destes músicos um conhecimento razoável de eletrônica, bem como, do sistema, complexo arranjo de fios de interconexão entre os equipamentos.

Na busca de soluções para estes problemas, e, com o objetivo de popularizar este novo paradigma de produção musical, alguns grandes fabricantes de equipamentos musicais (inicialmente três): a Roland Corporation, a Sequential Circuits e a Oberheim Electronics, se reuniram, em junho de 1981, na feira do NAMM (National Association of Music Merchants) (RATTON, 1995), onde definiram a criação de uma interface padrão (um sistema para interconectar equipamentos eletrônicos) que permitisse a simplificação do interfaceamento padrão entre equipamentos digitais eletrônicos voltados à produção musical, tanto para equipamentos proprietários, quanto para equipamentos de marcas e destinações diferentes.

Esta interface, este padrão, inicialmente foi denominado por **USI** (Universal Synthesizer Interface). Para interligar esta interface com os diversos teclados e equipamentos MIDI, surgiu a necessidade de se criar um protocolo padrão de comunicação com a mesma, denominado de **MIDI** (**M**usical **I**nstrument **D**igital **I**nterface). A primeira divulgação da interface **USI** e do protocolo **MIDI**, ao público, foi na mostra do **NAMM** de junho de 1982, disponibilizando, em janeiro de 1983, a primeira especificação do protocolo MIDI: a MIDI **Specification 1.0** [21] contendo todos os detalhes do protocolo[1] MIDI e da interface de comunicação - especificação esta que até hoje sobrevive sem modificações, apenas com alguns acréscimos já previstos na estrutura inicial do projeto.

O protocolo MIDI disponibiliza todas as informações e ferramentas para que se possam interconectar vários equipamentos digitais que possuam uma interface USI (MPU401 ou

[1] Protocolo - linguagem de comunicação. Uma ferramenta que permite a duas pessoas ou dois dispositivos se comunicarem (Ex. inglês, português, música....MIDI).

similar), bem como, também, um conjunto de especificações de como armazenar as informações musicais em arquivos digitais, para que, futuramente, o músico possa reproduzí-las em seu equipamento. Estes tipos de arquivos, projetados para registro das seqüências MIDI, foram denominados de **SMF**: **S**tandard **M**IDI **F**iles.

Inicialmente foram criados 3 (três) tipos básicos de formatos: o formato 0, o formato 1 e o formato 2. Destes, sobreviveram, comercialmente, os formatos 0 e 1.

O formato 0 é mais utilizado para equipamentos de *performance* ao vivo (teclados musicais, seqüenciadores, mesas de luz, mesas de som e outros) e o formato 1 para músicos que trabalham com edição de partituras por computador.

2.3.1 O que é MIDI?

MIDI é um protocolo padrão universal que apresenta um conjunto de mensagens capazes de levar toda a informação necessária a um equipamento musical eletrônico digital para torná-lo capaz de gerar ou reproduzir músicas ou fenômenos associados às mesmas.

Diferente do arquivo Wave que carrega em si uma cópia digitalizada da forma de onda contendo o sinal de uma música, um arquivo MIDI carrega em si apenas mensagens de como uma música deverá ser executada por um equipamento de síntese[2] musical.

Uma informação MIDI gera mensagens do tipo:

- Ative uma nota musical C4 (dó da oitava 4) durante 2 segundos utilizando o timbre de violão de Nylon.

Se fosse utilizado um arquivo Wave para armazenar tal som, no mesmo deveria constar o registro de 2 segundos de execução de uma nota C4 tocada ao violão. Se o som for armazenado em um arquivo wave de boa qualidade (44KHz, stereo e 16 bits de resolução), gerar-se-ia um espaço mínimo para registro do mesmo, em memória, de 44.000 pontos de armazenagem, onde cada ponto ocuparia 4 (quatro) Bytes, sendo dois

[2] Sintetizador – um equipamento capaz de gerar sons e timbres de instrumentos musicais

devido a resolução de 16 bits por amostra, e, mais dois, para a armazenagem do sinal em dois canais (estéreo).

Calculando o espaço de memória para armazenagem destes 44.000 pontos em 2 segundos, ter-se-ia: (44.000 x 2 x 2 x 2) Bytes = 352.000 Bytes, ou seja:

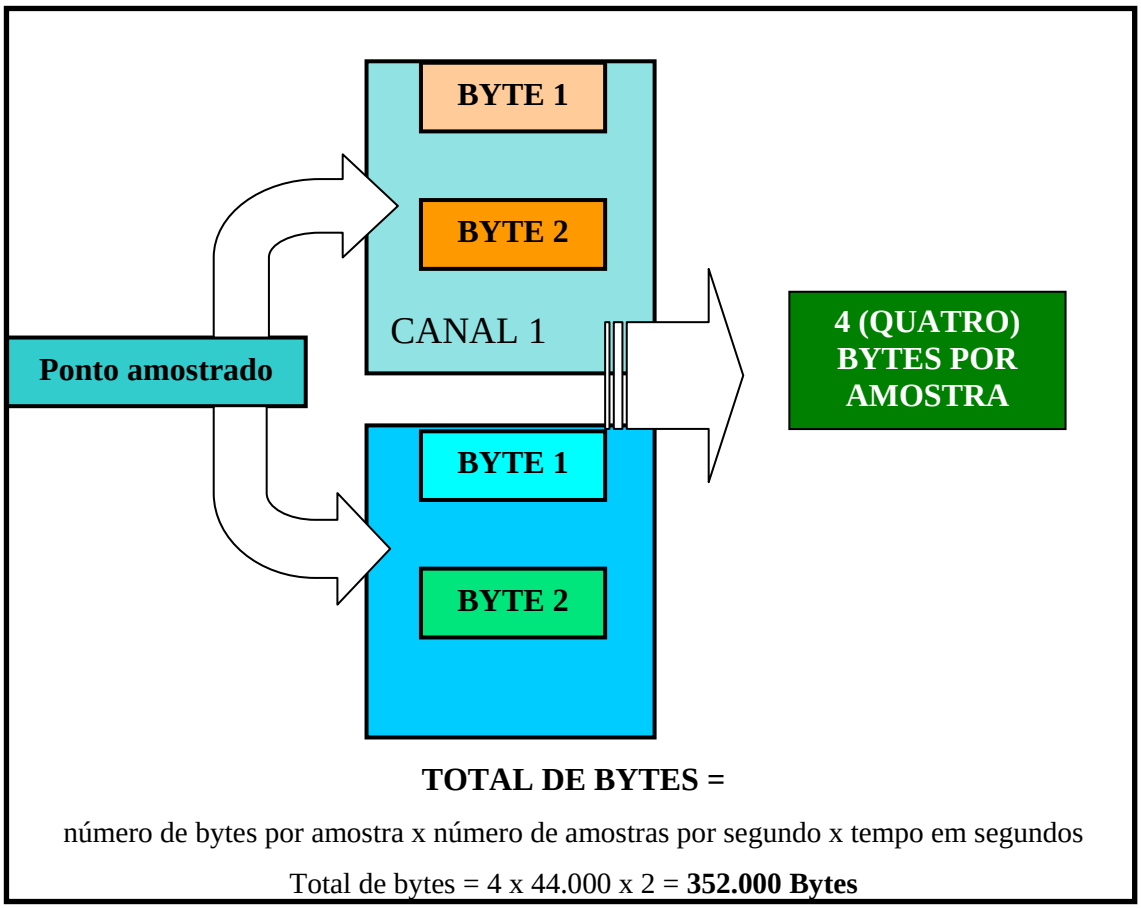


Diagrama 2.1 *– Bytes por ponto amostrado*

Por analogia, um minuto de música gastaria um espaço de memória de 10.560.000 (aproximadamente 7 disquetes de computador de 1.44MBytes[3]), ou seja: 4 x 44.0000 x 60 que é aproximadamente 10MBytes.

Em um arquivo MIDI, esta informação ocuparia apenas alguns bytes de dados, já que se armazena, nos arquivos SMF, apenas o que se deseja fazer, ou seja: tocar a nota C4 por 2 segundos com o timbre de violão.

[3] 1 M = 1 Mega = 1024 x 1024 Bytes = 1048576 bytes

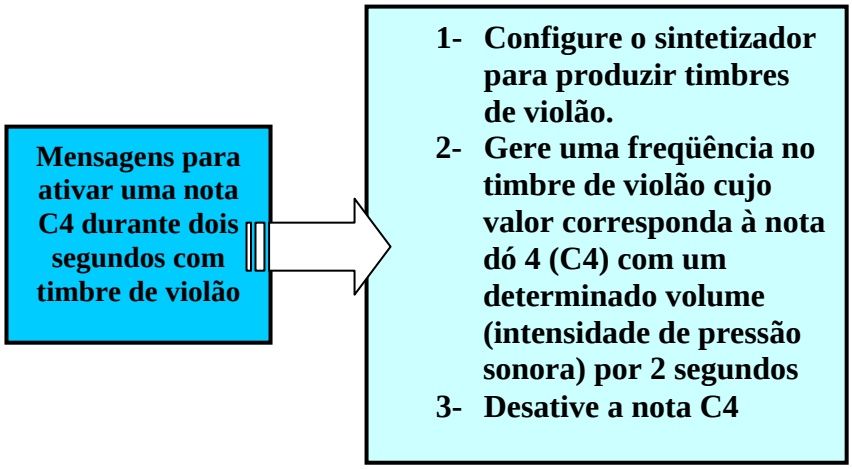


Diagrama 2.2 *– Informação em um arquivo MIDI para tocar a nota C4 por 2 segundos com o timbre de violão.*

Para exemplificar como nos SMF esta tarefa ocupa poucos Bytes de memória. Observe o código MIDI necessário, eliminando-se o cabeçalho do arquivo MIDI SMF, em hexadecimal[4], para executar tal tarefa.

C0 18 DT1 90 3C 64 DT2 80 3C 00 C0

- C0 informa que se deseja ativar um instrumento.
- 18 informa que este instrumento é o violão de Nylon.
- DT1 informa o tempo que se deseja esperar para a nota musical seja iniciada após a ativação do instrumento.
- 90 indica que o sintetizador deve iniciar a execução de uma nota.
- 3C indica que a nota deve ser o C4.
- 64 indica o volume da nota (vai de 0 a 7F).
- DT2 indica o tempo que se deverá esperar para desativar a nota (duração da nota).
- 80 indica que uma nota deverá ser desativada.
- 3C indica que a nota a ser desativada é a nota C4.
- 00 indica que o volume desta nota deverá ser 0 (desativada).

Obs. O tempo de duração DT1 e DT2 dependem de alguns parâmetros a serem definidos no cabeçalho do arquivo MIDI, fator tal que não carece ainda de maiores detalhes para que se possa entender o exemplo dado.

[4] Hexadecimal – sistema de numeração que possui 16 símbolos: os 10 símbolos do sistema decimal (de 0 a 9) acrescido das 6 primeiras letras do alfabeto (de A a F em maiúsculo).

Observe que armazenar 2 (dois) segundos de música com qualidade em mídia digital (Wave) gasta muito mais espaço de memória de computador do que armazenar a informação de como e qual nota musical deverá ser tocada (MIDI).

- No arquivo Wave (sem cabeçalho) ocupa-se 352.000 Bytes de memória.
- No arquivo MIDI (também sem cabeçalho), a informação **C0 18 DT1 90 3C 64 DT2 80 3C 00 C0** ocupa, aparentemente 11 Bytes, ou seja, 9 bytes de dados e dois de deltaTime[5].

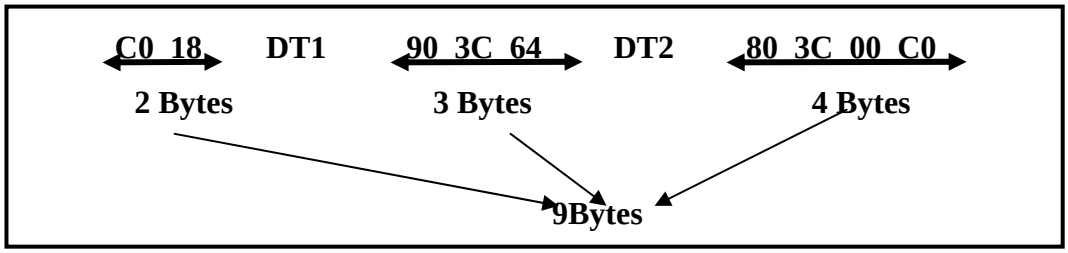


Diagrama 2.3 *– Informação MIDI contendo 9 Bytes, não incluindo os deltaTimes (DT1 e DT2)*

O que ocorre é que um deltaTime pode ter de 1 a 4 Bytes, conforme descrito anteriormente. Desta forma, esta informação final pode ter 11 Bytes, caso o DT1 e DT2 possuam apenas 1 Byte .

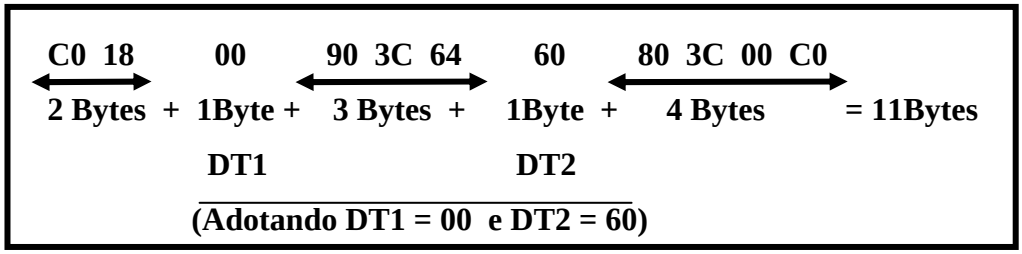


Diagrama 2.4 *– Informação MIDI contendo 11 Bytes*

Na pior das hipóteses, esta ação poderá ocupar 17 Bytes, no caso de DT1 e DT2 possuirem 4 Bytes.

[5] deltaTime – ver explicação no tópico DeltaTime, item 2.3.3.4.

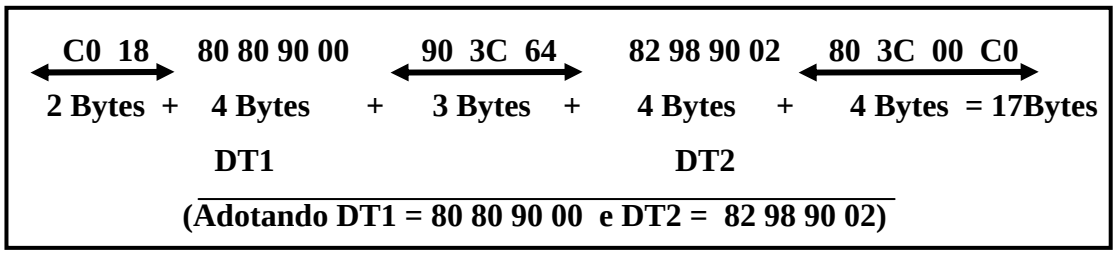


Diagrama 2.5 *– Informação MIDI contendo 17 Bytes*

Este fator, economia de espaço de memória, é um dos que tanto popularizou MIDI para aplicações em Internet, permitindo que músicas sejam transmitidas e recebidas pela WEB em tempos relativamente curtos.

2.3.2 Tamanho de arquivos sonoros de acordo com o tempo de duração da música: Wave x MIDI

É interessante analisar o que ocorre com o tamanho dos arquivos Wave e MIDI com o aumento do tempo de duração das músicas registradas pelos mesmos.

- **Wave**

Quando se deseja armazenar o dobro do tempo de uma execução musical em um arquivo sonoro Wave, isto implicaria em se dobrar a quantidade de memória para tal finalidade.

- **MIDI**

No caso de um arquivo MIDI, o aumento do tempo de execução musical não implica em nenhum aumento significativo de utilização de memória, já que tal fato só alteraria o valor do Byte responsável pela informação da duração da nota musical (DT2).

2.3.3 O Protocolo MIDI

MIDI é um protocolo criado inicialmente para o registro de músicas, crescendo em abrangência e potencialidades no controle automático de mesas de som, em sistemas de iluminação de shows e outras aplicações que surjam de acordo com os avanços tecnológicos e divulgação do protocolo. Alguns conceitos devem ser exemplificados para que se possa entender como as músicas são armazenadas e reproduzidas em sistemas MIDI. Dentre estes conceitos, serão abordados neste capítulo:

- Canal MIDI
- *Track* MIDI
- Ppq (ticks)
- DeltaTime
- Mensagens MIDI
- Arquivos SMF formato 0 e formato 1

2.3.3.1 Canais MIDI

O protocolo MIDI permite que se registre ou gere no mesmo até 16 instrumentos diferentes simultaneamente, o que, neste protocolo, é denominado por **canais MIDI**. Assim, com este protocolo pode-se representar músicas que tenham até 16 instrumentos tocando ao mesmo tempo para cada módulo[6] MIDI. Isto se dá devido ao fato de que no protocolo MIDI a mensagem de escolha ou mudança de instrumento utiliza apenas 4 bits de dados pra representar um canal MIDI (instrumento MIDI), e, desta forma, tem-se um possibilidade de se representar 16 opções de canais (de 0000 a 1111). Caso se deseje representar mais de 16 instrumentos simultaneamente existe uma fórmula adotada pelos implementadores de equipamentos e de softwares MIDI: utilizar mais de um módulo MIDI para geração e reprodução de mais 16 canais. Inclusive, para simplicidade e transparência de tal fato ao usuário, hoje em dia existem equipamentos que já internamente possuem mais de um módulo MIDI, fazendo crer que os mesmos trabalham com mais de 16 canais MIDI.

Assim, **canal MIDI = instrumento MIDI** que está sendo sintetizado ou reproduzido.

[6] Módulo MIDI – equipamento capaz de gerar, sintetizar e reproduzir timbres de instrumentos musicais.

2.3.3.2 *Track* MIDI

O conceito de *track* MIDI é o mesmo de uma pista (trilha) de gravação utilizada nos sistemas analógicos de gravação e reprodução sonora em fita magnética. Assim, podem-se utilizar "infinitos" *tracks* de gravação em programas de seqüenciamento musical utilizando computadores (dependendo da memória de cada computador). O fato de se ter apenas 16 canais MIDI por módulo não corresponde a dizer que se possui apenas 16 trilhas para se gravar uma música. Pode-se, por exemplo, desejar gravar um quarteto de flautas transversais, com cada melodia, de cada flauta, gravada em uma trilha independente. Neste caso, estar-se-á utilizando apenas um instrumento (um canal) MIDI para gerar o som de flauta transversal, mas, no caso, quatro trilhas independentes. Isto facilita a edição posterior das músicas gravadas, já que, se gravadas em um só *track*, ocorrendo um erro de gravação ter-se-ia de gravar todas as quatro flautas outra vez. Se gravadas em *tracks* separados, bastaria gravar novamente apenas a trilha onde o erro ocorreu.

2.3.3.3 Ppq

Ppq significa: Pulses per quarter note, ou seja, número de pulsos por unidade de tempo musical igual a uma semínima[7]. Em outras palavras, ppq significa em quantas partes temporais será dividida uma semínima. Em alguns programas de computador, no lugar de se utilizar a nomenclatura ppq utiliza-se ticks.

Sem entrar em detalhes de teoria musical, basta saber que em notação musical, os tempos são grafados de modo relativo, ou seja, não possuem uma duração fixa temporal. Assim, uma unidade de tempo relativa como a figura musical semínima, pode durar quanto tempo se queira, dependendo de um parâmetro musical denominado de metrônomo.

[7] Semínima – unidade de tempo de duração de uma nota musical. A mesma foi criada para se representar uma música no modo gráfico denominado de CPN (Commom Practice Notation) ou notação musical tradicional.

Um metrônomo informa quantas figuras musicais (no caso presente, a semínima) deverão ser tocadas no tempo de um segundo. Assim, se adotar-se um metrônomo[8] de 60 (60 pulsos por minuto), com uma figura de semínima, indica que ter-se-ia uma semínima sendo tocada por cada segundo (60 por minuto).

Uma ppq de valor 120, por exemplo, indica que cada tempo de uma semínima será dividido em 120 partes, e, desta forma, um sistema de gravação MIDI capturará eventos musicais com precisão de 1/120 avos de segundo (0.00833s), ou seja, com 8 milésimos de segundo.

Uma ppq de 480, por exemplo, indica que cada tempo de uma semínima será dividido em 480 partes, ou seja, um sistema de gravação MIDI capturará, com tal valor, eventos com precisão de 1/480 avos de segundo, ou seja, com precisão de 0.0020833s, ou seja, com uma precisão de 2 milésimos de segundo.

Atualmente existem sistemas que conseguem uma precisão de ppq com valor 1024, ou seja, com 1/1024 avos de um segundo, o que equivale a 0.0009765s (aproximadamente 1 milésimo de segundo).

Observa-se que a ppq é uma medida de precisão, de fidelidade da música registrada em um arquivo MIDI em comparação com a interpretação original que a produziu. Quanto maior for a ppq, mais definição, maior a resolução da música gravada ou reproduzida.

2.3.3.4 DeltaTime

Para que se entendam os deltaTimes e seu conceito, é necessário a compreensão de como funciona a máquina MIDI. A arquitetura da mesma consiste basicamente de um registrador de eventos e de um contador up->down (decrescente).

[8] Metrônomo - A figura musical (semínima, colcheia,...) de um metrônomo varia em uma peça musical de acordo com seu compositor, mas, no arquivo MIDI, a figura será sempre a semínima.

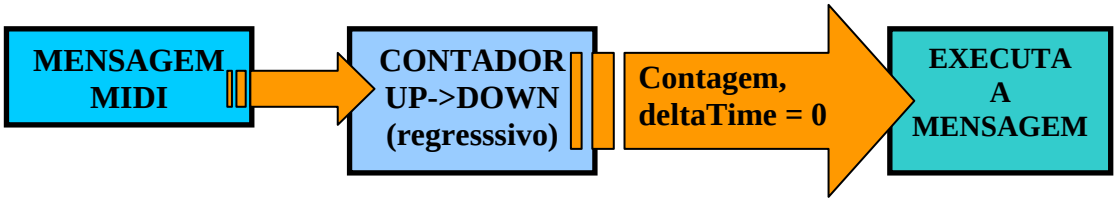


Diagrama 2.6 *- Arquitetura da máquina MIDI*

A estrutura da máquina MIDI funciona da seguinte forma:

1- O sistema recebe de um a quatro Bytes de informação (deltaTime) indicando uma contagem equivalente ao tempo em que a máquina MIDI deve esperar para executar a mensagem que virá logo após esta contagem.

2- Quando o contador chegar a zero (o clock –relógio- do contador regressivo é fixo e padrão em todas as máquinas MIDI), o sistema lê (ou recebe) e executa uma ação contida na mensagem MIDI em questão.

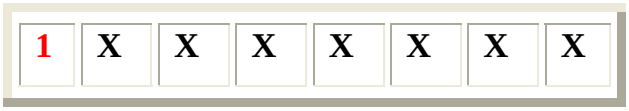
3- A máquina MIDI lê ou recebe uma nova contagem (deltaTime) e repete o processo até que uma mensagem de fim de *track* seja recebida.

Assim, uma mensagem MIDI completa possui um deltaTime seguido da mensagem de status contendo o código da ação que se deseja fazer, seguido ou não de um ou mais Bytes de dados necessários para o status, a ação, a ser executado.

Diagrama 2.7 *– Mensagem MIDI completa: DT1->Byte de Status->Byte de dados*

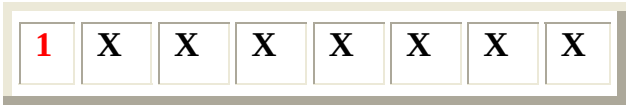
Se o bit mais significativo do primeiro Byte de deltaTime for 1, isto indica que este deltaTime terá um novo Byte, se este novo byte também possuir o bit mais significativo setado em 1, isto significa que o deltaTime terá mais um terceiro Byte, neste caso, o quarto Byte deverá ter o bit mais significativo setado em 0, já que o número máximo de Bytes de um deltaTime é 4.

Exemplo de deltaTime com quatro Bytes

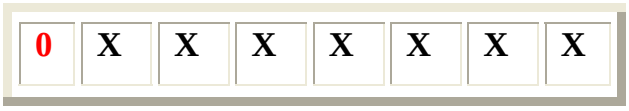


Primeiro Byte do deltaTime

Segundo Byte do deltaTime



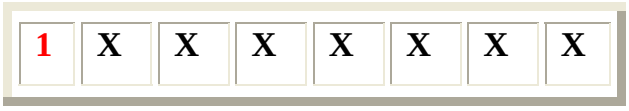
Terceiro Byte do deltaTime



Quarto Byte do deltaTime

X = tanto faz se 1 ou 0

Exemplo de deltaTime com 2 Bytes



Primeiro Byte do deltaTime

Segundo Byte do deltaTime

X = tanto faz se 1 ou 0

Exemplo de deltaTime com 1 Byte

Primeiro Byte do deltaTime

X = tanto faz se 1 ou 0

Diferença entre deltaTime e ppq

A diferença está na forma com que os dois são registrados.

- A ppq é registrada utilizando Bytes completos, ou seja, os 8 bits de um Byte.
- Os deltaTimes são registrados utilizando apenas os 7 bits menos significativos de um byte.

O motivo de se grafar os deltaTimes com apenas os 7 (sete) bits menos significativos de um byte reside no fato de que o oitavo bit é utilizado para informar à máquina MIDI quantos Bytes de contagem o deltaTime possuirá. Pode ser que, em alguns casos, um Byte apenas de contagem não seja o suficiente para representar o tempo de um determinado evento, e, desta forma, o sistema deverá ter um conhecimento prévio desta informação para que possa ativar corretamente o evento registrado na mensagem MIDI.

Exemplo de representação de ppq e deltaTime

Exemplo 1 –> Ppq = $90_{(10)}$

O valor 90 é menor que 127_{10} ($0111\ 1111_2$)

Conforme mostrado anteriormente, o maior valor que um deltaTime pode representar com um byte apenas é 127_{10} ($0111\ 1111_2$) , ou seja, com os 7 bits menos significativos do Byte igual a 1 e o mais significativo igual a 0.

Assim, uma Ppq de 90_{10} pode ser representada por um deltaTime de um Byte.

Neste caso, o Byte de ppq e do deltaTime serão iguais, ou seja:

0	1	0	1	1	0	1	0

Ppq = 90

0	1	0	1	1	0	1	0

DeltaTime de um Byte = 90

Exemplo 2 -> Ppq = $240_{(10)}$

O valor de ppq = 240 ultrapassa o maior valor que se pode representar com um Byte de deltaTime. Assim, necessitar-se-á de pelo menos dois bytes. O primeiro Byte terá o bit mais significativo setado em 1 para informar que o deltaTime terá mais um byte, e, no segundo byte, o bit mais significativo dele deverá ser setado em 0 para informar que o deltaTime acabou, ou seja, não possuirá mais Bytes.

1	?	?	?	?	?	?	?

DeltaTime – Byte 1

0	?	?	?	?	?	?	?

DeltaTime – Byte 2

O Byte de Ppq fica da seguinte forma para um valor de **240**:

1	1	1	1	0	0	0	0

Ppq = 240

Os Bytes de deltaTime serão montados utilizando os bits do Byte da ppq, ou seja, os 7 primeiros bits menos significativos da ppq vão para os sete primeiros bits menos significativos do Byte 2 e o bit mais significativo vai para o bit menos significativo do Byte 1.

Os demais bits do Byte 1 são preenchidos com 0, conforme mostrado a seguir:

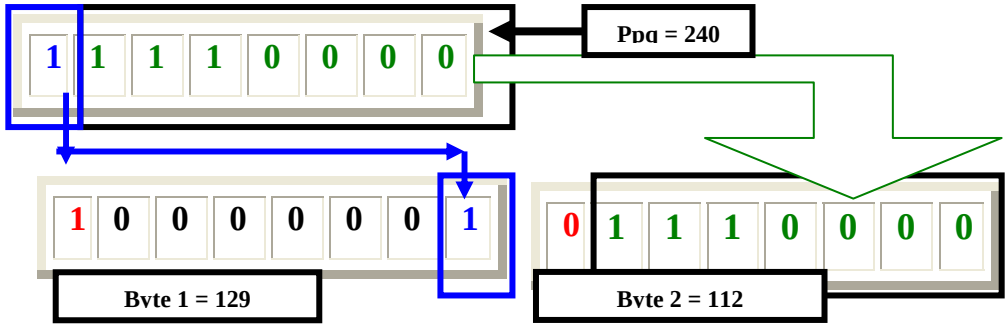


Diagrama 2.8 *– Cálculo da Ppq -> Ppq = 240*

Assim, o Byte de **ppq = 240** fica:

E os dois Bytes de deltaTime ficam:

Byte 1 = 129

Byte 2 = 112

Maiores detalhes podem ser vistos na dissertação de mestrado de Machado [19] e de Lopes [22].

2.3.3.5 Os arquivos SMF padrões

A construção dos Standard MIDI Files (SMF) está centrada na estrutura da máquina MIDI, ou seja, no cálculo correto dos deltaTimes através da ppq escolhida. Todos os cálculos, conforme já afirmado, são baseados no tempo de uma semínima (existe uma mensagem que permite informar o tempo de uma semínima em micro segundos).

SMF Formato 0 e Formato 1: diferença básica

A diferença básica entre os formatos 0 e 1 é que no formato 0 todas as notas de todos os canais são registradas no arquivo SMF em apenas um *track*. No formato 1, cada canal MIDI é registrado no arquivo SMF em um *track* independente.

2.3.3.6 O Arquivo MIDI SMF

Tanto o arquivo SMF formato 0 quanto o formato 1 possuem um cabeçalho contendo os parâmetros musicais necessários para a grafia da música no formato MIDI, bem como algumas informações sintáticas do arquivo. A seguir, é apresentada a estrutura dos arquivos SMF contendo as informações mínimas necessárias para o entendimento da

hierarquia da mesma e a formatação dos SMF. Os códigos de status e dados são apresentados em hexadecimal para facilitar a visualização dos mesmos.

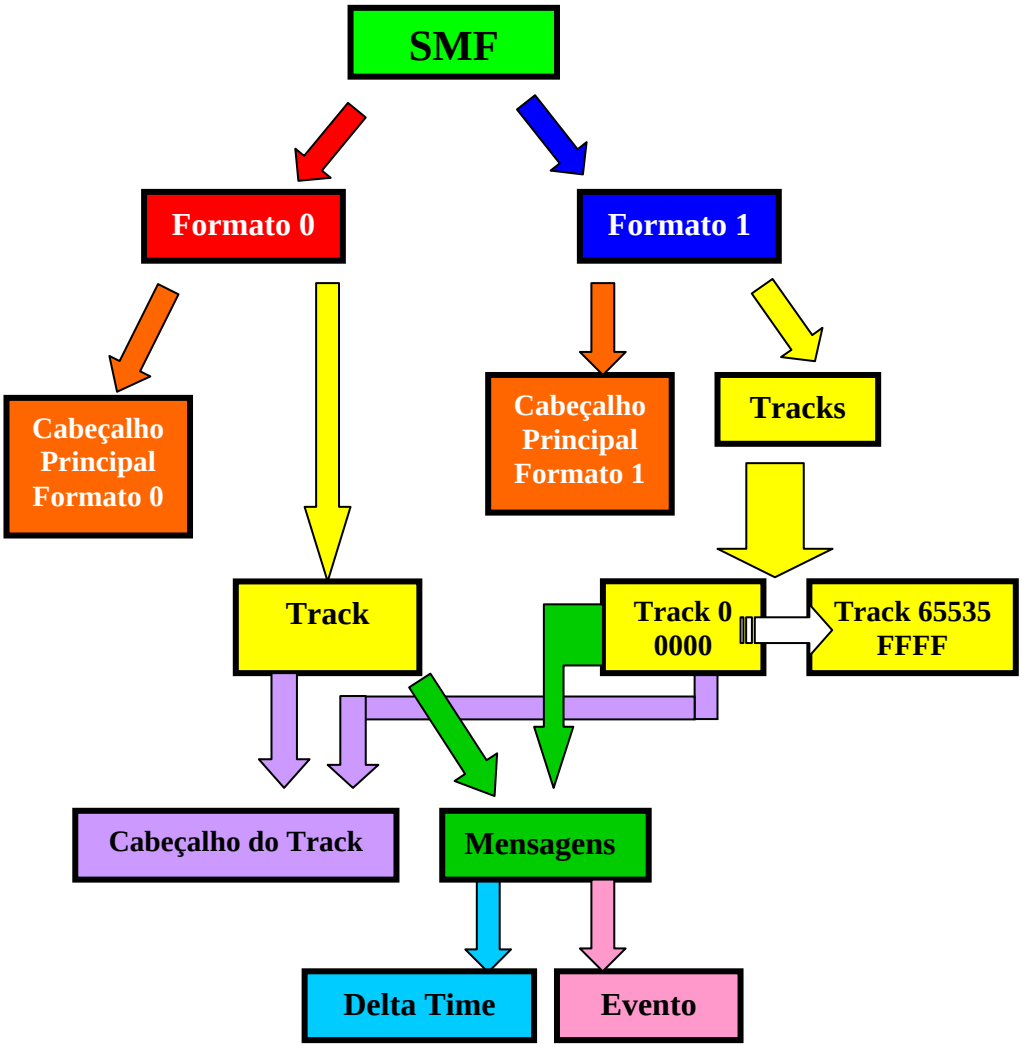


Diagrama 2.9 *– Estrutura dos arquivos SMF*

Cabeçalho Principal

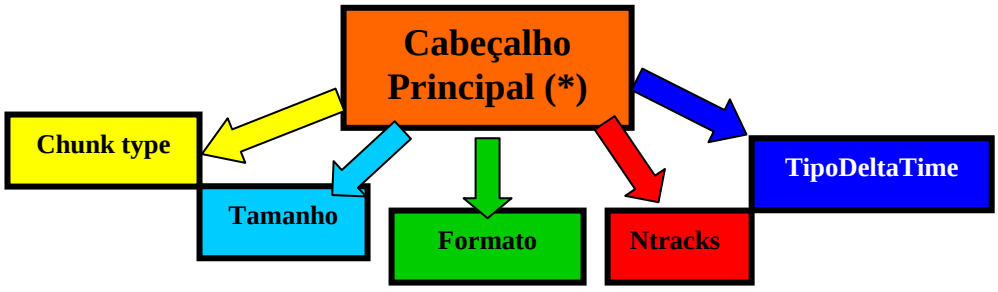


Diagrama 2.10 *– Estrutura dos arquivos SMF -> Cabeçalho Principal*

(*) – Uma diferença entre o formato 0 e o 1, no **Cabeçalho Principal**, está no campo **Formato**, como pode ser visto na tabela a seguir. Pode-se perceber que o tipo de

formato é definido por dois bytes, permitindo que novos formatos possam ser criados, até o limite de 65.536 (de 00 00 a FF FF) . Outros formatos foram idealizados inicialmente, tal como o formato 2, mas, com o tempo, chegou-se à conclusão de que apenas os formatos 0 e 1 são suficientes para as aplicações existentes.

- Outra diferença está no campo **Ntracks**, onde, no formato 0, sempre será 0001, ou seja, o formato 0 só possui um *track* para registrar as mensagens de todos os canais MIDI, enquanto, no formato 1, pode-se ter até 65.536 *tracks* (de 00 00 a FF FF).

Chunk type	: :	4 bytes: Mthd = 4D 54 68 64.
Tamanho	: :	Tamanho do Cabeçalho Principal: 6 bytes, sendo 2 bytes para indicar o Formato, 2 bytes para indicar Número de *Tracks*, 2 bytes para indicar o Tipo de DeltaTime.
Formato	: :	Indica o Formato com 2 bytes.: Formato 0 = 00 00, Formato 1 = 00 01
Ntracks	: :	Número de *Tracks* da música, sendo um *track* para cada canal MIDI e mais um para as configurações (metrônomo, armadura de clave, fórmula de compasso, etc). Possui 2 bytes.
TipoDeltaTime	: :	Indica o Tipo de DeltaTime. Possui 2 bytes. Se o bit mais significativo for 0 (zero) o DeltaTime será do tipo ppq. Se o bit mais significativo for 1 será do tipo SMPTE.

***Tabela 2.1** – Informações mínimas necessárias de um arquivo MIDI SMF -> Cabeçalho Principal*

Cabeçalho dos *Tracks*

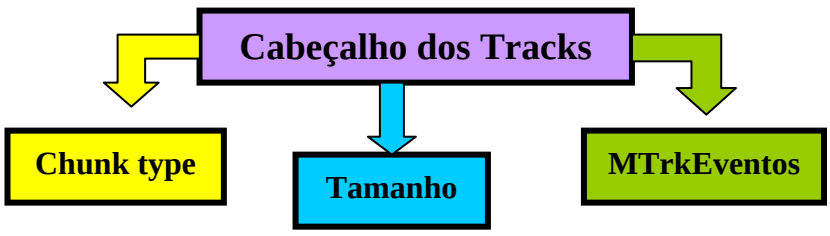


***Diagrama 2.11** – Estrutura dos arquivos SMF -> Cabeçalho dos Tracks*

Chunk type	:	4 bytes: Mtrk = 4D 54 72 6B.

Tamanho	: :	Possui 4 bytes. Indica a soma de todos os bytes do *track*, incluindo os 3 bytes indicativos do fim do *track*: FF 2F 00.

MtrkEventos	: :	DeltaTime	Eventos
DeltaTime	: :	Possui de 1 a 4 bytes. Indica quanto tempo o dispositivo, que está lendo e executando o arquivo MIDI, deverá esperar para iniciar a execução do evento que o segue.	

Eventos	: :	EventosMIDI	EventosSysex	MetaEventos
EventosMIDI	: :	A quantidade de bytes dependerá do tamanho do arquivo. Evento MIDI é qualquer mensagem de canal, ou seja, eventos de notas e os controles aplicados à elas.		
EventosSysex	: :	$F0_H$	Tamanho	BytesTrans
		São eventos utilizados para mandar mensagens exclusivas para um determinado equipamento.		
Tamanho	: :	O número de bytes dependerá do tamanho do Evento Sysex		
BytesTrans	: :	São os Bytes Transmitidos pelo Evento Sysex. Deverá terminar com o byte $F7_H$.		
MetaEventos	: :	Tipo	Tamanho	Texto
		São eventos não-MIDI contendo informações úteis, e necessárias para os equipamentos que executarão os eventos MIDI. Um Meta-Evento inicia-se com o byte FF. Os equipamentos que não reconhecem todos os tipos de Meta-Eventos devem ignorá-los sem emitir mensagem de erro.		
Tipo	: :	A quantidade de bytes dependerá do tipo de Meta-Evento. Os principais são: FF 51 = Set Tempo (Metrônomo). Possui 3 bytes. FF 58 = Fórmula de Compasso (Time Signature). Possui 4 bytes FF 59 = Armadura de Clave (Key signature). Possui 2 bytes. FF 2F = Fim de *Track*.		

		FF 01 = Texto. A quantidade de bytes varia de acordo com o texto. FF 04 = Nome do Instrumento. A quantidade de bytes varia de acordo com o instrumento. FF 05 = Letra da Música. A quantidade de bytes varia de acordo com o tamanho da letra. FF 06 = Marcas. A quantidade de bytes é variável.
Texto	: :	A quantidade de bytes varia de acordo com o texto.
Fim de *Track*	: :	FF 2F 00

***Tabela 2.2** – Informações mínimas necessárias de um arquivo MIDI SMF -> Cabeçalho dos Tracks*

Os arquivos MIDI apresentam uma solução compacta para armazenamento de uma música. O grande problema na sua utilização na implementação de sistemas de análise reside no fato de que a maioria destas informações não estarem explicitamente registradas nos mesmos conforme teoria musical, carecendo o programador de implementar sistemas de reconhecimento das estruturas musicais, tais como: duração de notas, tonalidade, metrônomo, divisão de compassos e outros mais.

Capítulo 3

Conceitos e Teoria Musical Básica

A música possui conceitos e sintaxe que a difere radicalmente de outras áreas do conhecimento, principalmente por trabalhar com um paradigma que exige uma grande abstração de quem pretende trabalhar em seu vasto domínio. Neste capítulo serão abordados os principais conceitos, termos e definições necessários para um melhor entendimento dos objetivos desta dissertação, bem como das soluções idealizadas para solucionar as metas descritas no capítulo 1. Assim, a seguir são apresentados os conceitos fundamentais deste domínio, partindo dos mais simples aos mais complexos pertinentes ao domínio deste trabalho.

3.1 Notas Musicais

Quando um músico provoca um movimento, por exemplo, em uma corda de um violão, a mesma produz um determinado som. Dependendo da velocidade com que esta corda vibra, sentimos uma sensação sonora diferente. Quanto mais fina a corda for e quanto mais esticada ela estiver, mais agudo é o som gerado, e, consequentemente, quanto mais frouxa a corda e quanto mais grossa ela for, mais grave será o som gerado. Percebe-se, nestes exemplos, que as freqüências[1] de cada som gerado, são diferentes.

[1] Frequência: variação de movimento de determinado dispositivo físico por um período de tempo. A frequência é medida em Hertz (Hz), ou seja, número de variações, vibrações, de movimentos, por segundo.

O ser humano, analogamente, produz sons através da passagem do ar por suas cordas vocais (tecnicamente denominadas de pregas vocais) fazendo-as vibrarem ao passar pelas mesmas (as cordas) gerando um determinado som. Na formação do som, além da vibração das cordas vocais pela passagem do ar, participam a anatomia humana, servindo de caixa de ressonância (vibração) a cavidade bucal, nasal, palato, língua, dentes, lábios e outros órgãos por onde o ar em vibração passará. A figura 3.1 mostra os pontos principais do aparelho fonador humano responsáveis pela geração do som após o ar ser expelido (expirado) pelo pulmão. A formação do som e explicação de como os fonemas são diferenciados pelos órgão são citados na figura 3.1, onde maiores detalhes poderão ser vistos no adendo 1.

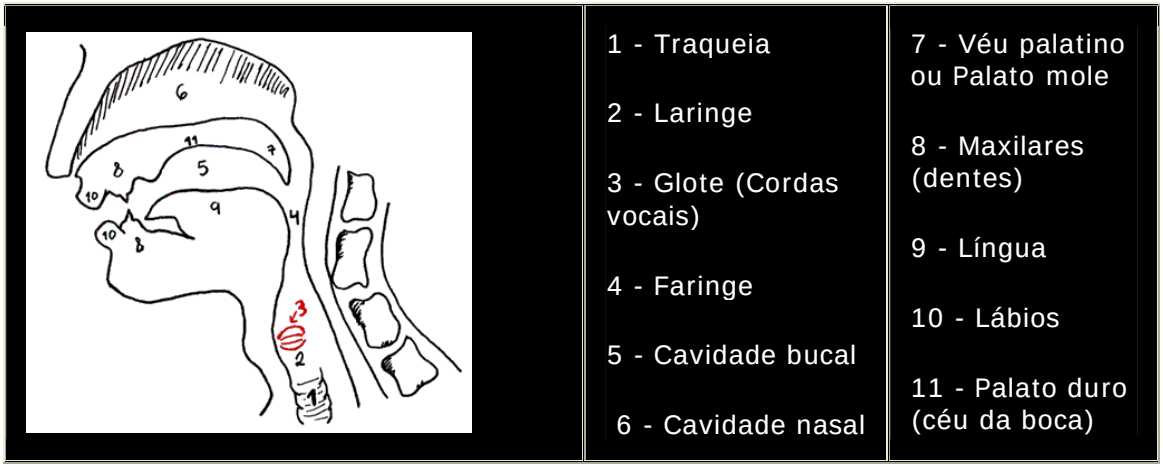


Figura 3.1 *– Aparelho Fonador humano*

A voz feminina é mais aguda do que a masculina devido às cordas vocais das mulheres serem menores, e, portando, vibram mais rápido. As cordas vocais dos homens, por analogia, são maiores, produzindo sons mais graves (vibram mais devagar). Observe a figura a seguir:

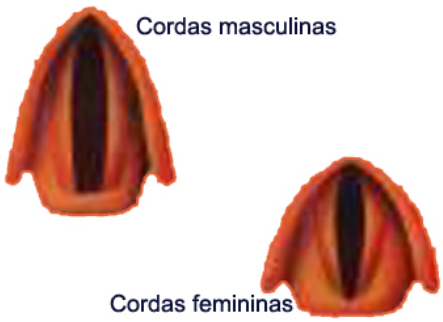


Figura 3.2 *– Cordas (Pregas) Vocais Masculinas e femininas*

3.1.1 A Escala bem temperada

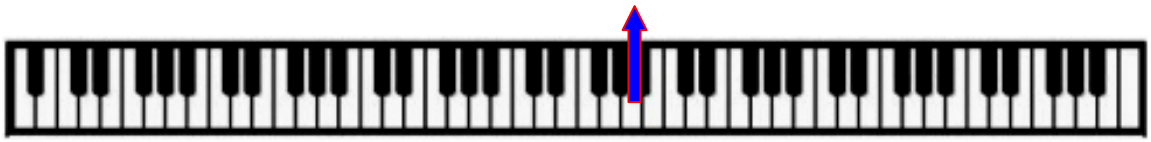
A partir do século XVIII, através do músico J.S.Bach, institui-se uma padronização para os sons produzidos por instrumentos musicais, adotando-se uma afinação padrão de onde gera-se uma escala[2] chamada de: **Escala Bem Temperada** [24]. Cada som nesta escala ou, cada freqüência de som desta escala, é denominado de Nota Musical.

Existem basicamente sete nomes de notas musicais, a saber:

Dó, Ré, Mi, Fá, Sol, Lá e Si

Cada uma destas notas musicais possui, assim, uma freqüência diferente e determinada. Uma destas notas foi adotada como freqüência de referência, denominada de diapasão. Esta nota musical, diapasão, é a nota musical **Lá**, cuja freqüência de vibração é de 440 Hertz.

Para localizá-la em um piano, basta saber que a mesma está aproximadamente no meio do teclado deste, sendo a 49ª nota deste teclado, conforme mostra a figura 3.3.

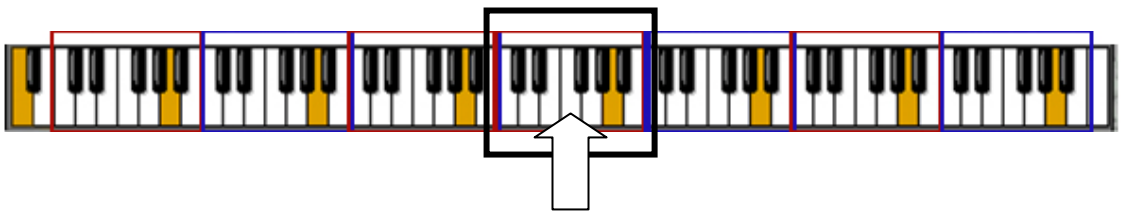


Lá central do piano = Lá 440Hz

Figura 3.3 *- Teclado de um piano com a nota lá 440hz*

3.1.2 Oitavas

Ao se observar este teclado de piano, percebe-se que existe um padrão, um conjunto de 12 teclas, que se repetem várias vezes. A este conjunto de doze teclas denomina-se de oitava, conforme pode ser visto na figura 3.4.



Oitava do Lá diapasão de 440 Hz

Figura 3.4 *– Oitavas Musicais*

[2] Escala musical - conjunto de sons de freqüências determinadas

Na figura 3.4 percebe-se que várias teclas foram marcadas. As mesmas correspondem, em cada oitava, à 10ª tecla de cada uma. O motivo deste desenho, deste arranjo de teclas existir nesta escala temperada, é devido ao fato de que o som produzido[3] por cada uma das notas musicais soa semelhante (varia apenas em altura : ou é mais grave ou mais agudo).

3.1.2.1 Numeração das oitavas e relação entre suas freqüências

As notas em cada oitava possuem o mesmo nome, sendo diferenciadas apenas por um índice representativo de sua oitava.

No sistema adotado no Brasil, nomeia-se o Lá diapasão de 440Hz por Lá3, e desta forma, o Lá da oitava anterior seria o Lá 2 e o da posterior o Lá 4, conforme mostrado na figura 3.5.

Figura 3.5 *– Numeração das oitavas*

Esta sensação, de serem as notas iguais, se dá devido ao fato de que a cada oitava na escala bem temperada, notas de mesmo nome possuem freqüências múltiplas da escala anterior, pelo fator de 2, ou seja: as notas de mesmo nome de uma oitava conseqüente de outra possuem o dobro da freqüência da mesma, e, da oitava anterior, a metade, conforme pode ser visto na figura 3.6.

Figura 3.6 *– Numeração das notas com suas freqüências múltiplas pelo fator de 2*

[3] Som produzido no piano – o mesmo é produzido pela percussão de um martelo, acionado por uma das teclas do piano, em uma determinada corda, pré-afinada na freqüência da nota correspondente.

3.1.3 Regra de geração das freqüências das notas musicais na escala bem temperada

Os valores das freqüências das notas musicais de cada oitava na escala bem temperada obedecem a seguinte regra:

- A freqüência de uma determinada nota musical é dada pelo valor da freqüência da nota anterior multiplicado pelo fator $2^{(1/12)}$.

Partindo de uma nota cuja freqüência é conhecida, como, por exemplo, Lá3 diapasão (440Hz), tem-se que a próxima nota musical terá freqüência igual à do Lá3 multiplicado por $2^{(1/12)} = (1.0594630)$.

Para exemplificar, observe que a nota musical Lá3 possui uma freqüência igual a 440hz e, portanto, a freqüência da nota musical Lá4 (**fLá4**) possuirá:

fLá4 = fLá3 x $2^{(1/12)}$ x $2^{(1/12)}$ x $2^{(1/12)}$ x $2^{(1/12)}$ x $2^{(1/12)}$ x $2^{(1/12)}$ x $2^{(1/12)}$ x $2^{(1/12)}$ x $2^{(1/12)}$ x $2^{(1/12)}$ x $2^{(1/12)}$ (o Lá4 está doze notas após o Lá3)

Ou seja:

fLá4 = fLá3 x $2^{(1/12)x12}$

-> **fLá4 = fLá3 x 2^{1}**

ou seja, **fLá4 = 2 x fLá3 ->** fLá4 possui o dobro da freqüência de Lá3

3.1.4 Nomenclatura utilizada para as notas musicais, sustenidos e bemóis

Cada nota musical pode ser também denominada por uma letra. A tabela a seguir mostra os nomes das notas e as letras correspondentes nas nomenclaturas européia* e americana:

* Na nomenclatura alemã, a nota musical **Si** é representada pela letra **H**, no lugar da letra **B**.

Nome	Letra
Dó	C
Ré	D
Mi	E
Fá	F
Sol	G
Lá	A
Si	B

Tabela 3.1 *- Nome das notas com suas respectivas letras*

A figura a seguir mostra um trecho do teclado do piano com as teclas e seus nomes.

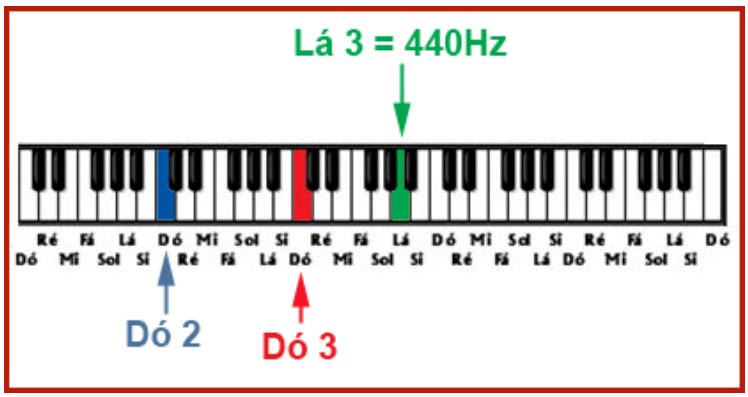


Figura 3.7 *– Notas Musicais e suas denominações*

Na figura 3.7 observa-se que apenas as teclas brancas do teclado do piano foram nomeadas, mas, em cada oitava, ainda existem 5 teclas (pretas) sem nome.

Estas teclas recebem o mesmo nome das teclas brancas que as precedem ou das que vêm após, acrescentadas de um símbolo, ou seja:

- uma tecla preta possui o nome da tecla (nota) anterior seguido do sinal # (lê-se sustenido), ou
- uma tecla preta possui o nome da tecla (nota) seguinte acrescentado o símbolo b (bemol).

A figura 3.8 mostra o teclado do piano com as teclas nomeadas utilizando letras para nomeá-las.

Figura 3.8 *- Teclas nomeadas com os símbolos (# ou b) acrescidos*

Assim, observe o seguinte exemplo:

Após qualquer nota musical Dó (C), tem-se uma tecla preta. Esta tecla possuirá o nome da nota musical em questão, ou seja: como ela é uma nota (tecla) após a nota Dó, a mesma poderá ser chamada de Dó# (C#) – dó sustenido. Por outro lado, esta mesma tecla (nota) está localizada uma tecla (nota) abaixo da nota Ré (D), e, desta forma, poderá ser chamada de Réb (Db) – ré bemol.

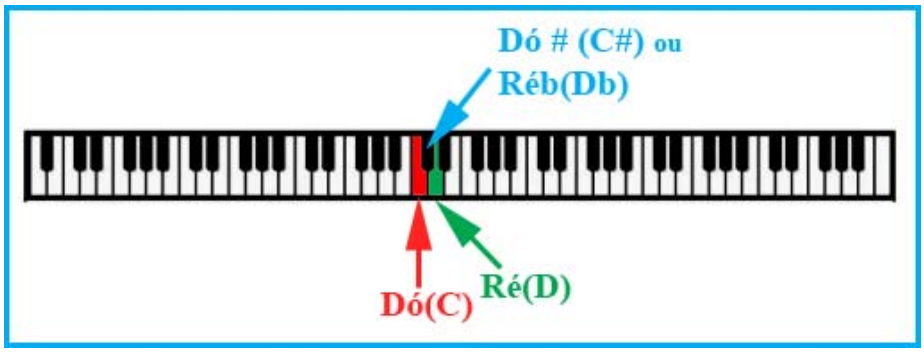


Figura 3.9 *– Nomeação da nota Dó (C) em relação aos sustenidos e bemóis: Dó# (C#) ou Réb (Db)*

3.1.5 Notas musicais e suas respectivas freqüências

A partir da regra básica de determinação da freqüência das notas musicais na escala bem temperada, pode-se obter todos os valores dentro da faixa do espectro audível pelo ser humano (entre 20 Hz e 22KHz).

Notas musicais	Freqüências
re#-1	19.4454 Hz
mi-1	20.6017 Hz
fa-1	21.8267 Hz
fa#-1	23.1246 Hz
sol1	24.4997 Hz
sol#-1	25.9565 Hz
la-1	27.50000 Hz
la#-1	29.1352 Hz
si-1	30.8677 Hz
do0	32,7032 Hz
do#0(reb0)	34,6478 Hz
re0	36,708 Hz
re#0(mib0)	38,8908 Hz
mi0	41,2034 Hz
fa0	43,6536 Hz
fa#0(solb0)	46,2494 Hz
sol0	48,9994 Hz
sol#0(lab0)	51,913 Hz
la0	55 Hz
la#0(sib0)	58,2704 Hz
si0	61,7354 Hz
do1	65,4064 Hz
do#1(reb1)	69,2956 Hz
re1	73,4162 Hz
re#1(mib1)	77,7818 Hz
mi1	82,4068 Hz

fa1	87,307 Hz
fa#1(solb1)	92,4986 Hz
sol1	97,9988 Hz
sol#1(lab1)	103,8262 Hz
la1	110 Hz
la#1(sib0)	116,541 Hz
si1	123,4708 Hz
do2	130,8128 Hz
do#2(reb2)	138,5914 Hz
re2	146,8324 Hz
re#2(mib2)	155,5634 Hz
mi2	164,8138 Hz
fa2	174,6142 Hz
fa#2(solb2)	184,9972 Hz
sol2	195,9978 Hz
sol#2(lab2)	207,6524 Hz
la2	220 Hz
la#2(sib0)	233,0818 Hz
si2	246,9416 Hz
do3 (dó central do piano)	**261,6256 Hz**
do#3(reb3)	277,1826 Hz
re3	293,6648 Hz
re#3(mib3)	311,127 Hz
mi3	329,6276 Hz
fa3	349,2282 Hz
fa#3(solb3)	369,9944 Hz
sol3	391,9954 Hz

sol#3(lab3)	415,3046 Hz
La3 (diapasão)	**440 Hz**
la#3(sib0)	466,1638 Hz
si3	493,8834 Hz
do4	523,2512 Hz
do#4(reb4)	554,3652 Hz
re4	587,3296 Hz
re#4(mib4)	622,254 Hz
mi4	659,2552 Hz
fa4	698,4564 Hz
fa#4(solb4)	739,9888 Hz
sol4	783,9908 Hz
sol#4(lab4)	830,6094 Hz
la4	880 Hz
la#4(sib0)	932,3276 Hz
si4	987,7666 Hz
do5	1046,5022 Hz
do#5(reb5)	1108,7306 Hz
re5	1174,659 Hz
re#5(mib5)	1244,508 Hz
mi5	1318,5102 Hz
fa5	1396,913 Hz
fa#5(solb5)	1479,9776 Hz
sol5	1567,9818 Hz
sol#5(lab5)	1661,2188 Hz
la5	1760 Hz
la#5(sib0)	1864,655 Hz

si5	1975,5332 Hz
do6	2093,0046 Hz
do#6(reb6)	2217,461 Hz
re6	2349,3182 Hz
re#6(mib6)	2489,0158 Hz
mi6	2637,0204 Hz
fa6	2793,8258 Hz
fa#6(solb6)	2959,9554 Hz
sol6	3135,9634 Hz
sol#6(lab6)	3322,4376 Hz
la6	3520 Hz
la#6(sib0)	3729,31 Hz
si6	3951,0664 Hz
do7	4186,009 Hz
do#7(reb7)	4434,922 Hz
re7	4698,6362 Hz
re#7(mib7)	4978,0318 Hz
mi7	41,2034 Hz
fa7	5587,6518 Hz
fa#7(solb7)	5919,9108 Hz
sol7	6271,927 Hz
sol#7(lab7)	6644,8752 Hz
la7	7040 Hz
la#7(sib0)	7458,6202 Hz
si7	7902,1328 Hz
do8	8372,018 Hz
do#8(reb8)	8869,8442 Hz
re8	9397,2726 Hz

re#8(mib8)	9956,0634 Hz
mi8	10548,0818 Hz
fa8	11175,3034 Hz
fa#8(solb8)	11839,8216 Hz
sol8	12543,854 Hz
sol#8(lab8)	13289,7504 Hz
la8	14080 Hz
la#8(sib0)	14917,2404 Hz
si8	15804,2656 Hz
do9	16744,0362 Hz
do#9(reb9)	17739,6884 Hz
re9	18794,5452 Hz
Re#9(mib9)	19912,127 Hz
mi9	21096,1636 Hz
fa9	22350,6068 Hz

***Tabela 3.2** – Notas Musicais e suas respectivas freqüências*

3.2 Timbre

Ao se escutar uma flauta, um violão, um piano ou diferentes tipos de vozes cantando, o indivíduo é capaz de diferenciar cada um deles. Isso se deve ao fato de cada um deles possuir uma qualidade de som diferenciada a qual denominamos de **Timbre.** Tanto os instrumentos musicais quanto os diversos tipos de vozes possuem timbres específicos que nos permitem diferenciá-los.

Tecnicamente, o **Timbre** de cada instrumento é formado pelo seu conteúdo harmônico (espectro de freqüências) e pelo envelope sonoro das notas musicais que ele emite. Como exemplo, tem-se a seguinte afirmação no livro do Sound Forge 8.0:

> *"(...) podemos perceber facilmente a modificação do timbre do instrumento original. (...) se cortarmos as freqüências das notas de uma flauta transversal acima de 12.000 Hz, perderemos a percepção do sopro produzido pelo intérprete. Isso faz com que o som seja mais puro, mas o torna irreal, artificial, mais eletrônico."*
> (LIMA; MACHADO; PINTO, 2005, p.204)

Essa afirmação mostra a importância do espectro de freqüências na formação do timbre. A outra afirmação retirada do mesmo livro fala da importância do envelope sonoro:

> "(...) Temos instrumentos que, após a nota ser tocada, demoram mais tempo que outros para parar de soar; temos instrumentos cujo som começa mais baixo e aumenta lentamente; com outros ocorre o inverso, enfim, possuem particularidades que determinam juntamente com as harmônicas, o timbre do instrumento.(...) podemos definir estas características como **envelope sonoro** que é composto de: **ADSR** – **A**ttack, **D**ecay, **S**ustain e **R**elease."
> (LIMA; MACHADO; PINTO, 2005, p.205)

Pode-se definir melhor o **ADSR**:

Attack: Tempo no qual o volume chega ao pico máximo;

Decay: tempo no qual o volume chega ao nível normal;

Sustain: tempo em que o volume se mantém estável;

Release: Tempo em que o volume passa do nível normal ao mudo.

Os três exemplos a seguir ilustram a importância do envelope sonoro, composto de regiões de ADSR. Ele é um dos elementos responsáveis pela diferenciação timbral dos instrumentos musicais, já que cada instrumento possui **Ataque** (Attack), **Decaimento** (Decay**)**, **Sustentação** (Sustain) e **Término** (Release) diferentes ao entoar uma nota ou uma melodia.

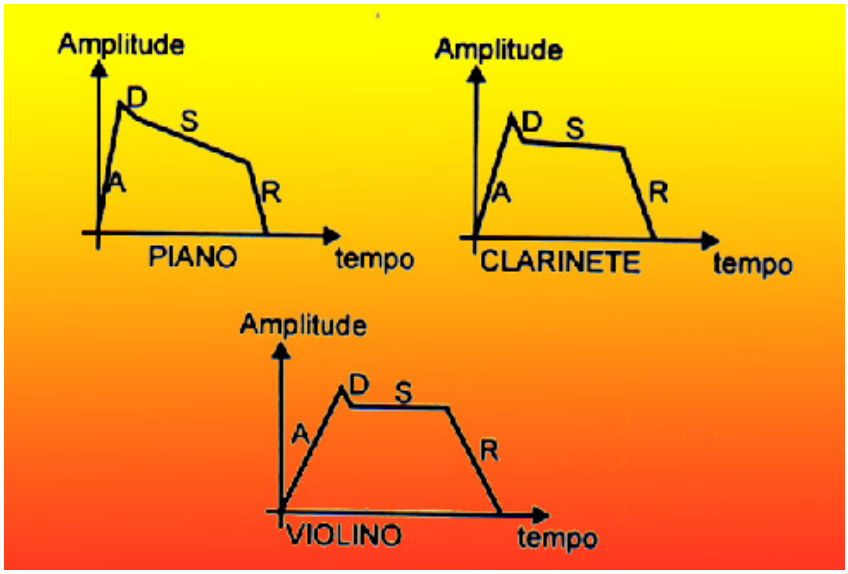


Figura 3.10 *– Envelope sonoro de três instrumentos musicais: Piano, Clarinete e Violino.*

3.3 Extensão Vocal ou Instrumental

A extensão vocal ou instrumental corresponde a todas as notas que uma pessoa ou instrumento musical consegue emitir. Estas notas não precisam ser emitidas necessariamente com qualidade vocal ou instrumental.

Na figura a seguir é ilustrado, em um teclado de piano, um exemplo de uma pessoa com alcance vocal de dezoito notas.

Figura 3.11 *– Extensão vocal de um indivíduo qualquer (notas alcançadas sem total qualidade vocal)*

Alguns instrumentos musicais possuem a capacidade de geração de um range de notas maior que sua tessitura, ou seja, geram notas musicais que não possuem um timbre agradável ou descaracterizado do instrumento. O mesmo acontece com o ser humano: nem todas as freqüências, notas musicais, produzidas pelo aparelho fonador são agradáveis, de qualidade timbral, de onde conclui-se que o range vocal é geralmente maior que a tessitura vocal de um instrumento ou cantor.

A seguir é mostrado como exemplo a diferença entre o range de um Clarinete Alto (Clarone) e sua respectiva tessitura.

- **Range** de Sol 1 a Láb4
- **Tessitura** de Sol 1 a Fá 4

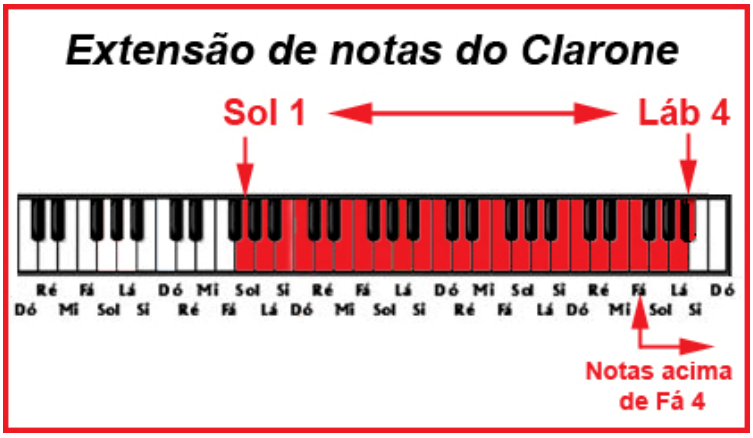


***Figura 3.12** – Clarinete Alto em Mi Bemol[4] com sua respectiva extensão de notas[5]*

3.4 Tessitura ou *Vocal Range*

Sob o ponto de vista da voz humana, a Tessitura é a extensão vocal ou musical de um cantor onde o som produzido é emitido com qualidade, sem que o indivíduo desafine ou force a sua voz de uma maneira indesejada. A tessitura vocal de um indivíduo é analisada por um especialista quando o mesmo vai identificar a classificação vocal (*vocal type*) de um determinado cantor. Assim, como existes pessoas com características e extensão vocais diferentes, existem, também, diversas classificações vocais para identificar cada tipo de cantor.

Observe o exemplo a seguir que apresenta dois tipos de tessitura vocal, uma com extensão vocal maior que a outra, determinada por um profissional (especialista em canto ou um fonoaudiólogo). Isto não significa que a voz de um cantor seja melhor que a do outro, apenas que um alcança mais notas que o outro. Como está se tratando de tessituras, os dois cantores cantarão agradavelmente em suas respectivas tessituras. Caberá ao ouvinte humano julgar qual voz é melhor, ou qual mais lhe agrade.

[4] Clarinete Alto em Mi Bemol[4] - denominado de Clarone, no Brasil.
[5] Extensão de notas do Clarone - *"As notas acima de Fá4 são de difícil acesso e de timbre desagradável" (LIMA,1958, p.23).*

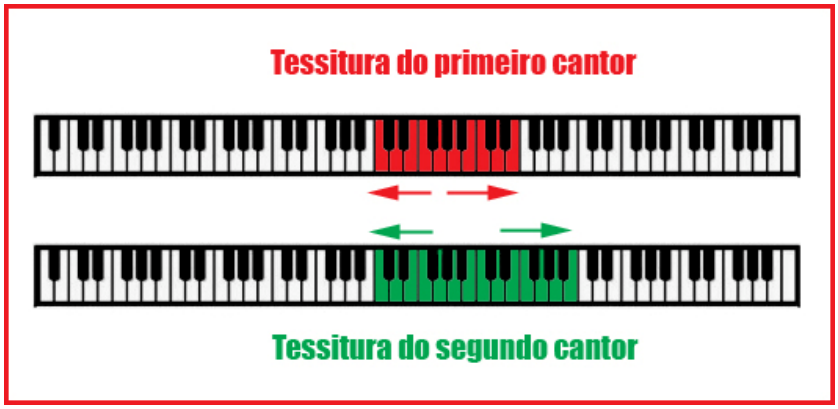


Figura 3.13 *– Tessituras vocais de dois indivíduos sendo que o segundo possui um alcance vocal de sete notas acima*

3.5 Classificação Vocal ou *Vocal Type*

3.5.1 O porquê de se classificar as vozes

Cantar e não conseguir alcançar as notas de uma determinada música é um problema que causa sérios transtornos a um cantor, principalmente aos profissionais. Muitas vezes isto se dá devido ao fato do cantor estar cantando fora de sua tonalidade de conforto. Para evitar que isto ocorra, é importante que o cantor conheça sua **classificação vocal**[6]**,** e que, conhecendo-a, converta, transponha, todas músicas de seu repertório para a tonalidade adequada à sua qualificação.

3.5.2 Classificação Vocal (Em inglês:*Vocal Type*)

É a extensão de notas que alguém consegue emitir confortavelmente e com qualidade, dentro de uma determinada região musical (grave ou aguda). A classificação vocal deve ser feita por um profissional da área, o qual avaliará o cantor através de testes padrões de emissão de notas, precedidos de exercícios vocais (vocalizes) que preparem a voz do cantor para produzir o maior range possível, com a maior qualidade também..

[6] Classificação Vocal – naipe vocal característico de cada pessoa, classificado de acordo com o alcance das notas a serem emitidas de uma maneira agradável, e de exercícios vocais. Exemplo: Existem pessoas que alcançam regiões mais agudas, outras pessoas alcançam regiões mais graves.

A classificação vocal é diferente para os homens e para as mulheres pois ambos alcançam regiões musicais diferentes. Normalmente as mulheres alcançam regiões musicais mais agudas e, os homens, regiões mais graves.

Existem três classificações vocais básicas (fundamentais) para homens e mulheres:

HOMENS	MULHERES
Baixo (Voz Grave)	**Contralto** (Voz Grave)
Barítono (Voz Média – entre o Baixo e o Tenor)	**Meio-Soprano ou Mezzo-Soprano** (Voz Média – entre o Contralto e o Soprano)
Tenor (Voz Aguda)	**Soprano** (Voz Aguda)

Tabela 3.3 *– Classificação Vocal Básica Masculina e Feminina*[7]

Para se entender melhor essa classificação, observe, em um teclado de piano, algumas ilustrações de classificações em um teclado de piano.

Classificação Masculina:

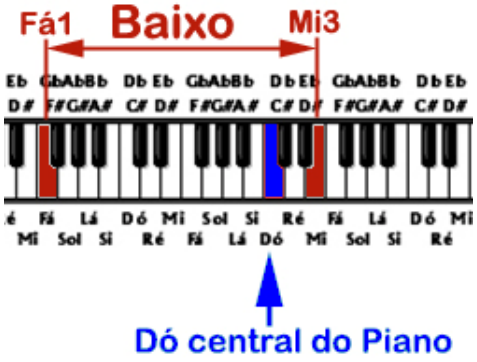


Figura 3.14 *– Extensão de notas alcançadas com qualidade pelo Baixo masculino (Tessitura - Fá1 a Mi3)*

[7] Classificação vocal masculina e feminina – (ver Referências - numeração [26]).

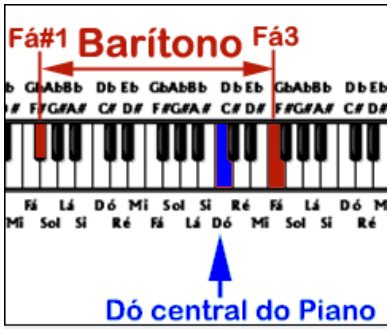


Figura 3.15 *– Extensão de notas alcançadas com qualidade pelo Barítono (Tessitura - Fá#1 a Fá3)*

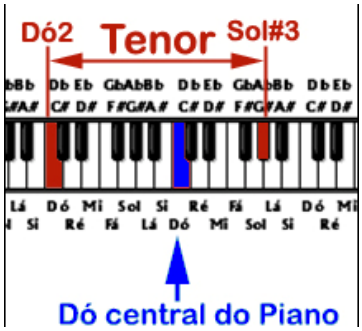


Figura 3.16 *– Extensão de notas alcançadas com qualidade pelo Tenor (Tessitura - Dó2 a Sol#3)*

Classificação Feminina:

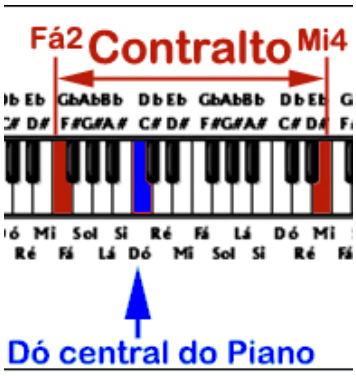


Figura 3.17 *– Extensão de notas alcançadas com qualidade pela Contralto (Tessitura - Fá2 a Mi4)*

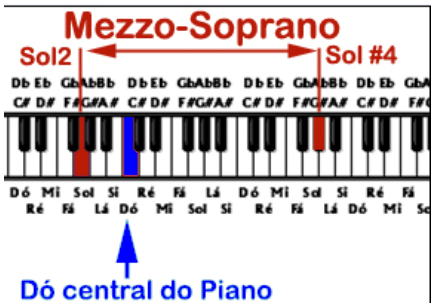


Figura 3.18 *– Extensão de notas alcançadas com qualidade pela Mezzo-Soprano (Tessitura - Sol2 a Sol#4)*

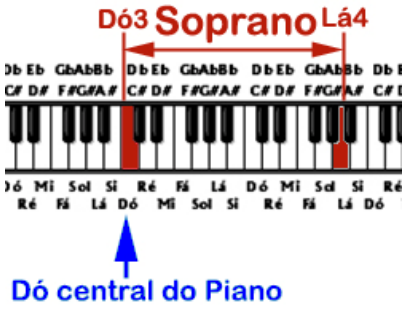


Figura 3.19 *– Extensão de notas alcançadas com qualidade pela Soprano (Tessitura - Dó3 a Lá4)*

- **Observações:**

A classificação das vozes é diversificada (sem padronização), variando de autor para autor, entre profissionais do canto e entre escolas. A classificação adotada nessa dissertação segue a tendência da maioria dos programas de edição de música por computador, programas estes consagrados em todo o mundo, tais como: Sibelius[14], Finale[13], Band-in-a-Box [12] e outros mais.

3.6 A importância da identificação da tonalidade ideal de um cantor, além de sua classificação vocal

Identificando-se a tessitura de um cantor pode-se determinar sua classificação vocal. A grosso modo, pode-se perceber onde os limites da tessitura do cantor quando o mesmo começa a forçar a voz ou quando o som da mesma se torna desagradável, desafinado (não produzindo notas musicais de acordo com suas freqüências).

Quando o indivíduo canta juntamente com uma melodia de um CD, rádio e outros, e força a sua voz, desafinando ou não, diz-se que ele está cantando fora de sua tonalidade, fora de sua classificação vocal. Quando isto ocorre, o mesmo estará desagradando o público ouvinte, assim como estará correndo o risco de causar danos às suas cordas vocais.

Dessa forma, antes de iniciar um canto, o cantor deverá cantar a nota mais grave e a mais aguda da música e verificar se o range está dentro de sua tonalidade de conforto. Caso isto não ocorra, deve-se eliminar a peça musical de seu repertório, ou, melhor, transpô-la para sua tonalidade.

3.6.1 Conceito de Tonalidade

Uma tonalidade, de uma forma macroscópica, é um conjunto de notas musicais que soam agradavelmente quando tocadas em seqüência ou simultaneamente.
Ela é formada por um conjunto de notas de mesmo nome, podendo estas notas pertencer a qualquer oitava musical.

Cada tipo de tonalidade possui uma regra de formação para determinação do conjunto de notas musicais que pertencem à mesma.

O conjunto de notas de uma tonalidade se inicia por uma nota musical cujo nome é o mesmo da tonalidade, seguida por novas notas distanciadas das precedentes conforme a regra desta tonalidade.

Para que se possa entender estas “distâncias”, é necessário que se conceitue o que é a distância de **um tom** e a distância de um **semitom** (meio tom).

Para facilitar, serão ilustrados estes conceitos pela observação de um teclado de um piano.

Distância de um semitom (meio-tom)

Uma nota musical está distanciada de outra por um semitom quando ela é a próxima nota ou a nota anterior a uma determinada nota. Em um teclado de piano pode-se dizer que um semitom é a distância entre uma tecla e outra adjacente.
Observe a figura 3.20 onde se tem exemplos da distância de um semitom.

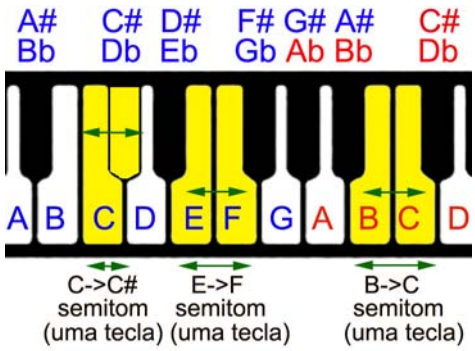


Figura 3.20 *– Distância de um semitom*

Distância de um tom

Uma nota musical está distanciada de outra por um tom quando ela é a segunda nota antes ou após uma determinada nota. Ou seja, para se ter a distância de um tom entre duas notas musicais, deve-se ter uma nota musical entre elas. Em um teclado de piano pode-se dizer que um tom é a distância entre duas teclas adjacentes separadas por uma outra tecla, seja esta preta ou branca. Para tornar mais clara esta explicação, observe a figura 3.21 onde são apresentados exemplos da distância de um tom.

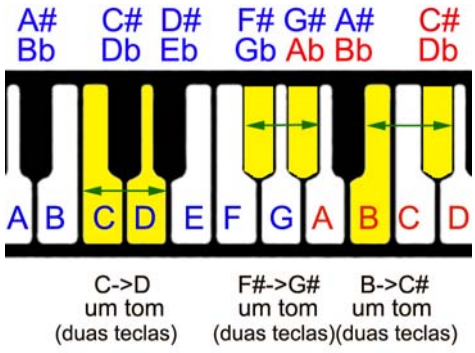


Figura 3.21 *– Distância de um tom*

Observe que entre as notas **C** e **D** existe uma tecla (uma nota) preta (**C#**), entre as notas **F#** e **G#** existe uma tecla correspondente à nota **G**, e, entre as notas **B** e **C#** existe a nota **C**.

3.6.2 Tipos de tonalidade

A seguir é mostrado, como exemplo, duas das principais tonalidades: a tonalidade maior e a menor natural.

3.6.2.1 Tonalidade maior

A tonalidade maior apresenta uma seqüência de notas com uma distância determinada entre elas, repetindo esta seqüência nas demais oitavas.
Observe, como exemplo, a tonalidade de dó maior e suas respectivas notas.

- **Notas da tonalidade de Dó maior** = **C D E F G A B** (dó, ré, mi,fá,sol,lá,si)

Observando estas notas em um teclado de piano, em uma oitava qualquer, pode-se notar que nesta tonalidade apenas são utilizadas as teclas brancas, que estão na figura salientadas em amarelo.

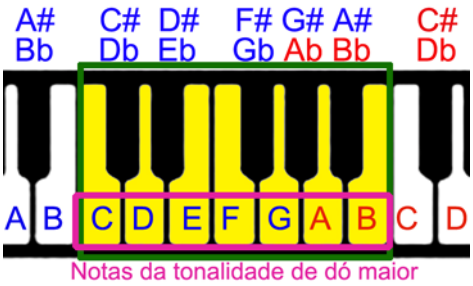


***Figura 3.22** – Notas da Tonalidade de Dó Maior*

Observe que as distâncias das notas são:
C->1 tom->**D**->1 tom->**E**->1/2 tom->**F**->1 tom->**G**->1 tom->**A**->1 tom->**B**

T T S T T T

Diz-se que a fórmula da tonalidade maior é: **TTSTTT**, onde: **T** = tom e **S** = semitom.

- **Notas da tonalidade de Si maior**

Para se montar as notas da tonalidade de Si maior, deve-se seguir a mesma regra da de dó maior, assim como de qualquer outra tonalidade maior.

Assim, tem-se:

B->1 tom->**C#**->1 tom->**D#**->1/2 tom->**E**->1 tom->**F#**->1 tom->**G#**->1 tom->**A#**

Tonalidade de Si maior = **B C# D# E F# G# A#** (Si, Dó#,Ré#, Mi, Fá#, Sol#, Lá#)

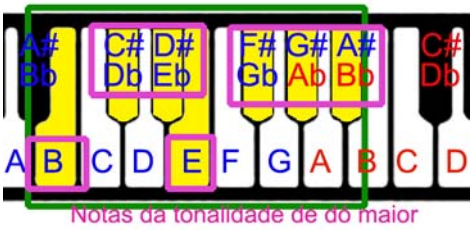


Figura 3.23 *– Notas da Tonalidade de Si Maior*

Qualquer outra escala maior pode ser montada facilmente obedecendo esta regra de distância.

3.6.2.2 Tonalidade menor

Existe mais de um tipo de tonalidade menor, possuindo cada uma sua respectiva regra. A tonalidade menor natural, assim como a maior já apresentada, segue uma regra para gerar uma seqüência de notas com uma distância fixa entre elas. Apesar das modificações serem mínimas, a sensação sonora percebida por um ouvinte é bem diferente quando se ouve uma escala em tonalidade maior e menor. A tonalidade menor dá uma sensação maior de paz, de tranqüilidade. A regra que vale para as notas de uma oitava, vale, também, para as demais.

Observe, como exemplo, a tonalidade de lá menor e suas respectivas notas.

- **Notas da tonalidade de lá menor natural** = **A B C D E F G** (lá, si, dó, ré, mi, fá, sol)

Observando estas notas em um teclado de piano, em uma oitava qualquer, pode-se notar que nesta tonalidade apenas são utilizadas, também, as teclas brancas, destacadas na figura a seguir, em amarelo.

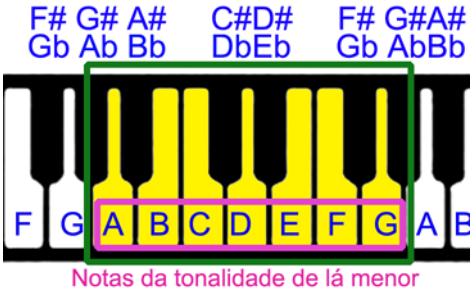


***Figura 3.24** – Notas da Tonalidade de Lá Maior*

Observe que as distâncias das notas são:
A->1 tom->**B**->1/2 tom->**C**->1 tom->**D**->1 tom->**E**->1/2 tom->**F**->1 tom->**G**

T S T T S T

Diz-se que a fórmula da tonalidade menor natural é: **TSTTST**, onde: **T** = tom e **S** = semitom.

- **Denominando todas as notas a partir de uma tonalidade**

Foi visto que cada tonalidade possui sete notas musicais básicas. Cada nota desta tonalidade passa a possuir um número de 1 a 7. No caso da tonalidade de dó maior ter-se-ia:

NÚMERO	1	2	3	4	5	6	7
NOTA	C	D	E	F	G	A	B

***Tabela 3.4** – Notas com seus números correspondentes*

Foi visto, também, que em cada oitava existem 12 notas (cinco a mais que as notas da tonalidade). Apesar destas outras cinco notas não pertencerem à tonalidade, elas podem ser utilizadas em uma música em condições especiais, conforme teoria musical de harmonia [27] [28], e, apesar disto, soarem agradavelmente. Assim, tais notas também carecem de uma numeração que as identifiquem na estrutura musical.

NÚMERO	1	?	2	?	3	4	?	5	?	6	?	7
NOTA	C	C# Db	D	D# Eb	E	F	F# Gb	G	G# Ab	A	A# Bb	B

***Tabela 3.5** – Notas que não pertencem à tonalidade mas que podem ser utilizadas por ela*

Como as notas da tonalidade já estão numeradas, as demais também ganham uma numeração especial, baseada em regras da harmonia musical, mostrada na tabela 3.6 tomando como exemplo novamente a escala de dó maior.

NÚMERO	*1*	*2-*	*2*	*3-*	*3+*	*4*	*5-*	*5*	*5+*	*6*	*7 ou 7-*	*7+*
NOTA	C	C# Db	D	D# Eb	E	F	F# Gb	G	G# Ab	A	A# Bb	B

***Tabela 3.6** – Numeração especial das notas fora da tonalidade*

Assim, mantém-se a numeração para as notas da tonalidade, sendo a terceira nota uma das exceções, onde a terceira nota passa a existir em duas nomenclaturas 3- (terça menor) e 3+ (terça maior). Isto é feito devido ao fato de que, na tonalidade menor, utiliza-se a nota correspondente a (3-) e, na maior, a (3+), o que soa extremamente óbvio.

No caso da sétima nota, a mesma também sofre alteração de nome, para sétima menor (7-) e sétima maior (7+).

Assim, cada nota pertencente a uma tonalidade possui uma denominação especial, de acordo com a função, do contexto de cada uma. A tabela 3.2,a seguir, mostra, para qualquer tonalidade, a numeração e denominação de cada nota musical, iniciando esta numeração a partir da nota do tom. Para simplificação da mostra, apresenta-se apenas uma seqüência de 23 notas numeradas a partir da nota do tom.

Tom_1	2-	2	3-	3+	4	5-	5	5+	6	7- ou 7	7+	8	9-	9	9+	10	11	11+	12	12+	13
A	A#	B	C	C#	D	D#	E	F	F#	G	G#	A	A#	B	C	C#	D	D#	E	F	F#
A#	B	C	C#	D	D#	E	F	F#	G	G#	C	A#	B	C	C#	D	D#	E	F	F#	G
B	C	C#	D	D#	E	F	F#	G	G#	A	A#	B	C	C#	D	D#	E	F	F#	G	G#
C	C#	D	D#	E	F	F#	G	G#	A	A#	B	C	C#	D	D#	E	F	F#	G	G#	A
C#	D	D#	E	F	F#	G	G#	A	A#	B	C	C#	D	D#	E	F	F#	G	G#	A	A#
D	D#	E	F	F#	G	G#	A	A#	B	C	C#	D	D#	E	F	F#	G	G#	A	A#	B
D#	E	F	F#	G	G#	A	A#	B	C	C#	D	D#	E	F	F#	G	G#	A	A#	B	C
E	F	F#	G	G#	A	A#	B	C	C#	D	D#	E	F	F#	G	G#	A	A#	B	C	C#
F	F#	G	G#	A	A#	B	C	C#	D	D#	E	F	F#	G	G#	A	A#	B	C	C#	D
F#	G	G#	A	A#	B	C	C#	D	D#	E	F	F#	G	G#	A	A#	B	C	C#	D	D#
G	G#	A	A#	B	C	C#	D	D#	E	F	F#	G	G#	A	A#	B	C	C#	D	D#	E
G#	A	A#	B	C	C#	D	D#	E	F	F#	G	G#	A	A#	B	C	C#	D	D#	E	F

Tabela 3.7 *– Vinte e três notas com suas denominações para todas as tonalidades*

Tonalidade de dó maior

A escala de dó maior possui as seguintes notas e denominações conforme a tabela 3.8:

Dó maior

1	2 -	2	3 -	3 +	4	5 -	5	5 +	6	7 - ou 7	7 +	8	9 -	9	9 +	10	11	11 +	12	12 +	13
C	C #	D	D # Eb	E	F	F # Gb	G	G # Ab	A	A # Bb	B	C	C # Db	D	D # Eb	E	F	F # Gb	G	G # Ab	A

Tabela 3.8 *– Notas e denominações para a escala de Dó Maior*

Ou seja:

Dó maior = 1, 2, 3+, 4, 5, 6, 7+, 8, 9, 10, 11, 12, 13,....

Tonalidade de dó menor

Dó menor

1	2 -	2	3 -	3 +	4	5 -	5	5 +	6	7 - Ou 7	7 +	8	9 -	9	9 +	10	11	11 +	12	12 +	13
C	C #	D	D # Eb	E	F	F # Gb	G	G # Ab	A	A # Bb	B	C	C # Db	D	D # Eb	E	F	F # Gb	G	G # Ab	A

Tabela 3.9 *– Notas e denominações para a escala de Dó Menor*

Dó menor = 1, 2, 3-, 4, 5, 5+, 7-, 8, 9, 11,12,12+,....

Tonalidade de Lá menor

Assim, olhando o exemplo dado da escala de dó maior, a escala de lá menor possuiria as seguintes notas:

1	2 -	2	3 -	3 +	4	5 -	5	5 +	6	7 -	7 +	8	9 -	9	9 +	10	11	11 +	12	12 +	13
A	A #	B	C	C #	D	D #	E	F	F #	G	G #	A	A #	B	C	C #	D	D #	E	F	F #

***Tabela 3.10** – Notas e denominações para a escala de Lá Menor*

Lá menor = 1, 2, 3-, 4, 5, 5+, 7-, 8, 9,11,12,12+,....

3.7 Entendendo a questão da Transposição

3.7.1 Conceito de Transposição

Transpor uma melodia ou uma música, do ponto de vista da teoria musical, baseando-se em tonalidades, é algo complexo e que demanda um bom conhecimento teórico do músico. Por outro lado, do ponto de vista do resultado (a música transposta), fazer uma transposição é algo relativamente simples.

Transpor uma música é mudar todas as notas musicais pertencente à mesma de uma região (faixa de notas) para uma região mais grave ou mais aguda, mantendo as distâncias originais entre as notas da melodia ou música inicial.

Para tornar mais simples a explicação, observe o exemplo:

Exemplo de transposição

Dada a melodia (um conjunto de notas em seqüência), deseja-se transpor a mesma, quatro semitons acima:

C 4 -> A4 -> E5 -> D5 -> C4 ->G4 -> C4

Para transpor a melodia escrita anteriormente, (**C 4 -> A4 -> E5 -> D5 -> C4 ->G4 -> C4**), basta olhar o nome das notas da seqüência, sem se preocupar com a oitava da mesma. Através do relógio de notas mostrado na figura 3.25, pode-se calcular as distâncias entre elas, tanto olhando no sentido horário quanto anti-horário.

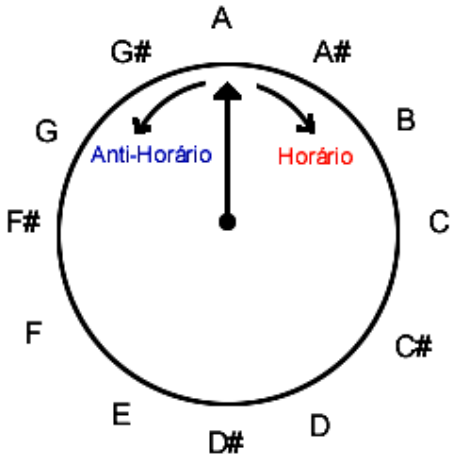


***Figura 3.25** - Relógio de notas com a distância entre elas*

Distância de **C4** a **A4** = distância de **C** a **A** = **-3S** (o sinal negativo (-3S) indica que se deve olhar no relógio no sentido anti-horário), onde S = um semitom. Observe a figura 3.26.

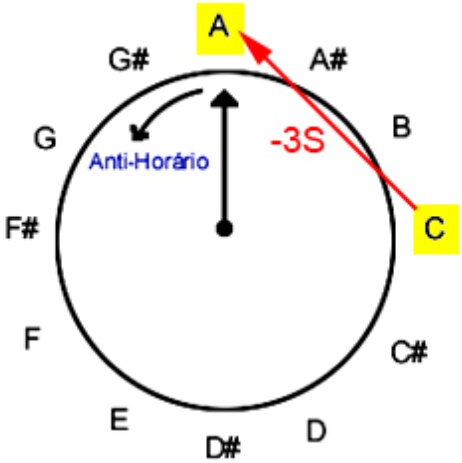


Figura 3.26** - Distância de **C4** a **A4,** igual a **-3S

Distância de **A4** a **E5** = distância de **A** a **E** = **-5S**. Observe a figura 3.27.

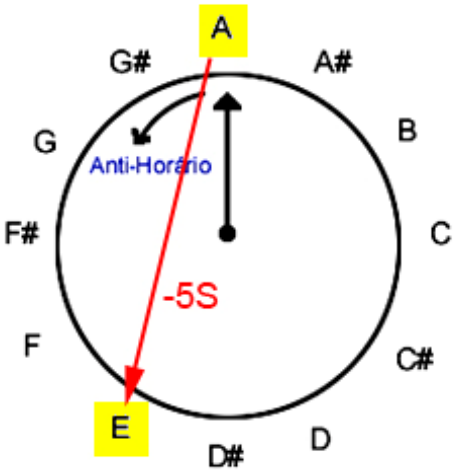


Figura 3.27 *- Distância de **A4** a **E5,** igual a **-5S***

Distância de **E5** a **D5** = distância de **E** a **D** = **-2S**. Observe a figura 3.28.

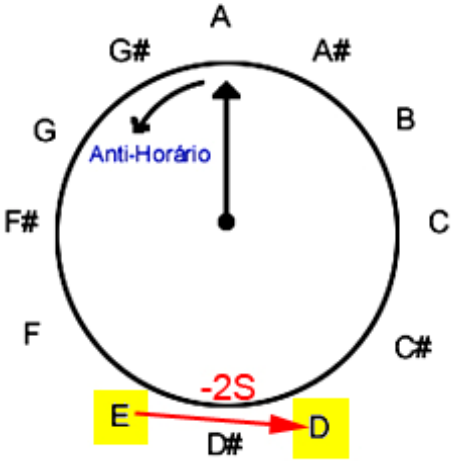


Figura 3.28 *- Distância de **E5** a **D5,** igual a **-2S***

Distância de **D5** a **C4** = distância de **D** a **C** = **-2S**. Observe a figura 3.29.

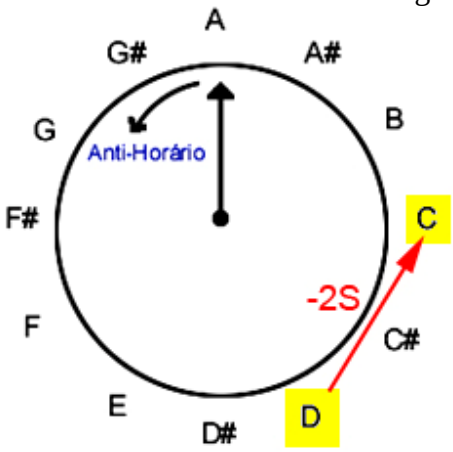


Figura 3.29 *- Distância de **D5** a **C4,** igual a **-2S***

*Distância de **C4** a **G4** = distância de **C** a **G** = **-5S***. Observe a figura 3.30.

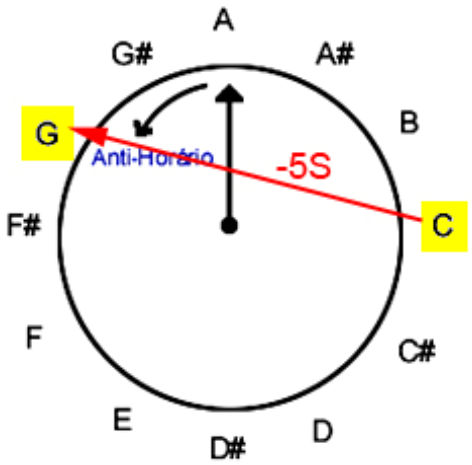


Figura 3.30** - Distância de **C4** a **G4**, igual a **-5S

*Distância de **G4** a **C4** = distância de **G** a **C** = **5S***. Observe a figura 3.31.

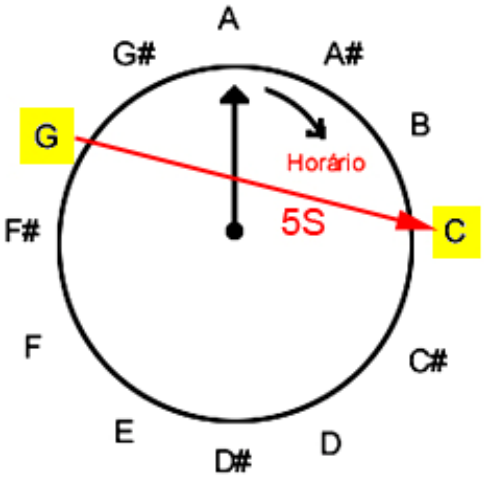


Figura 3.31** - Distância de **G4** a **C4**, igual a **5S

Assim, a seguinte seqüência musical:

C 4 -> A4 -> E5 -> D5 -> C4 ->G4 -> C4

Fica com a seguinte distância entre as notas:

C 4 -> -3S ->A4 -> -5S-> E5 -> -2S-> D5 -> -2S -> C4 -> -5S->G4 -> 5S-> C4

Para transpor esta seqüência 4 (quatro) semitons acima, basta iniciar a seqüência nova (transposta) pela nota musical quatro semitons acima.

No caso, a seqüência original inicia na nota musical **C4**.

A nota musical 4 (quatro) semitons acima da nota **C4**, olhando o teclado do piano ou o relógio, é a nota **E4**.

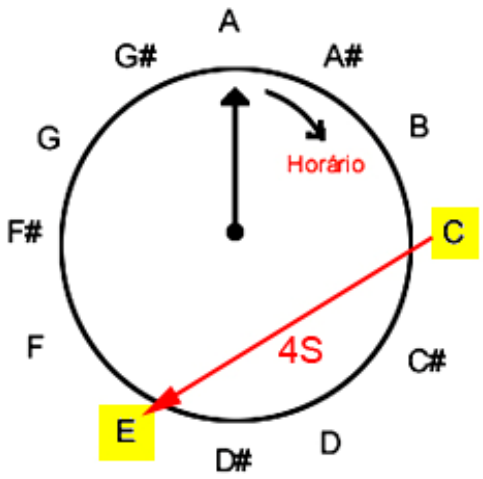


Figura 3.32 *- Nota musical* ***C4****, quatro semitons acima, correspondendo à nota* ***E4***

Assim, a nova seqüência da música transposta 4 (quatro) semitons acima deverá iniciar pela nota musical **E4** mantendo a demais notas da seqüência à mesma distância das notas da seqüência original.

Seqüência original:

C 4 -> -3S ->A4 -> -5S-> E5 -> -2S-> D5 -> -2S -> C4 -> -5S->G4 -> 5S-> C4

A nova seqüência fica:

E 4 -> -3S ->C#4 -> -5S-> G#5 -> -2S-> F#5 -> -2S -> E4 -> -5S->B4 -> 5S-> E4

Assim, com os conceitos e exemplos apresentados, julga-se ter explorado o mínimo necessário para que o leitor possa, sob o ponto de vista musical, acompanhar os desenvolvimentos deste trabalho.

Capítulo 4

Implementação de um sistema de transposição automática de seqüências musicais MIDI baseada na tessitura de um determinado cantor

4.1 Observações Iniciais

Esta dissertação apresenta uma solução de como resolver um relevante problema no domínio musical, podendo, o mesmo, acarretar danos à saúde e transtornos profissionais aos músicos em geral. Tais problemas surgem quando os músicos efetuam *performances* não condizentes com as esperadas tecnicamente na execução de uma determinada peça musical.

Diz-se isso quando um músico, principalmente os cantores, tenta executar uma peça musical e não consegue reproduzir, mesmo forçando, algumas freqüências (notas musicais) existentes na obra em questão. Neste caso, o menor transtorno que poderá ocorrer a este músico é a frustração de produzir uma interpretação desagradável para si mesmo e para o público que estiver presente.

Este tipo de execução forçada pode produzir um efeito colateral grave à saúde deste cantor, onde seu instrumento musical é o seu próprio sistema fonador. Tal problema ocorre devido ao fato do cantor ter que realizar um grande esforço para emitir

freqüências fora de sua tessitura vocal[1], desafinando, ou, até mesmo, não conseguindo emitir determinadas notas musicais fora de seu range[2] vocal.

A repetição sistemática deste tipo de ação, ou seja, esforços excessivos em suas pregas vocais pode acarretar calosidades (nódulos) e outros problemas nas mesmas, conforme mostrado na figura 4.1 e no Capitulo 1 desta dissertação (com mais detalhes), os quais podem comprometer a carreira profissional deste músico.

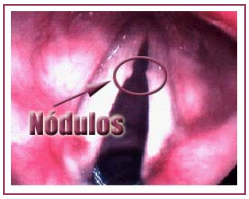


Figura 4.1 *– Nódulos em pregas vocais*

A ocorrência de nódulos vocais poderá levar o músico a ser submetido a longos tratamentos com fonoaudiólogos, e, até mesmo, a ter que fazer cirurgia(s) para remoção dos mesmos, podendo, em alguns casos, tornar tal profissional não mais apto a executar profissionalmente seus trabalhos com a qualidade necessária.

Não menos relevante do que a questão de possíveis danos à saúde do cantor é o dano profissional acarretado à carreira do mesmo. O fato de ele desafinar ao executar uma peça musical ou não conseguir emitir sons registrados na mesma durante uma apresentação, pode acarretar traumas que o incapacite de exercer novamente sua profissão com segurança, bem como pode levar o mesmo ao descrédito e discriminação pelo público em geral.

4.2 O problema da implementação do sistema

Conforme visto no capítulo 2, o protocolo MIDI e os arquivos SMFs carregam potencialmente todas as informações musicais necessárias para que um sistema especialista possa realizar várias tarefas no domínio musical.

[1] Tessitura vocal – faixa de freqüência que um cantor consegue emitir com qualidade, sem distorção e sem esforço excessivo.
[2] Range – faixa de frequência do espectro audível.

Diz-se potencialmente porque muitas das informações relevantes não estão explicitamente registradas nos arquivos SMF ou nas mensagens de tempo real do protocolo MIDI. Vários parâmetros musicais importantes, tais como: duração de uma nota musical (figura musical), divisão de compassos, tonalidade real e metrônomo devem ser identificados e quantificados através de um sistema especialista a ser desenvolvido pelo programador. Implementar tais sistemas demanda do programador um conhecimento profundo do protocolo MIDI, da estrutura dos SMFs, das especificidades dos fabricantes de equipamentos MIDI, de lógica e de manipulação de dados em sistema de numeração binário.

O conhecimento e as informações de como proceder a este tipo de implementação, principalmente no tocante a MIDI, são difíceis de serem obtidos, já que os mesmos não estão registrados em literatura nem disponibilizados pelos desenvolvedores de softwares ou mesmo pela sociedade afim (MMA-MIDI Manufacture Association).

Felizmente, tal tecnologia já é de conhecimento e domínio da equipe de pesquisadores da FEELT-UFU, a qual está registrada em vários trabalhos de mestrado, doutorado e iniciação científica desta instituição, bem como em dezenas de artigos, dicas publicadas e vários livros e revistas especializadas da área.

Um outro fator limitante, cuja solução está na escolha de um paradigma de programação adequado, se dá na programação de sistemas MIDI, onde se necessitam, na maioria das linguagens atuais, principalmente as procedimentais, de drivers e dlls específicos para controle e manipulação das placas e dos recursos sonoros do computador, os quais dependem de máquina e do sistema operacional utilizados.

A escolha pela linguagem funcional CLEAN, vem tornar o desenvolvimento e o código gerado legível e portável, prescindindo-se de ter que atualizar drivers e dlls em cada plataforma e sistema operacional onde o aplicativo for rodar.

4.3 Requisitos mínimos necessários para implementação dos objetivos traçados

Para que se possa implementar um sistema que realize a transposição automática de uma seqüência MIDI para uma região de conforto de um determinado cantor com um range vocal especifico, é necessário que o sistema, antes, seja capaz de identificar e realizar as seguintes tarefas:

1. Ser capaz de abrir um arquivo MIDI;
2. Seja capaz de reproduzir este arquivo para apreciação do cantor;
3. Separar os canais MIDI;
4. Identificar e separar em cada canal MIDI os eventos, as mensagens de ativação e desativação de notas das mensagens de sistema, para que se possa reconhecer as mensagens MIDI, através da transformação dos deltaTimes em unidades de tempo absoluto para cada nota musical existente na seqüência.(ver item 4.3.1);
5. O sistema deverá ser capaz de identificar ou classificar os eventos MIDI para que se possa detectar os status de Program Change (escolha de instrumento), e, desta forma, reconhecer todos os instrumentos existentes na seqüência (ver item 4.4.1.1);
6. Determinar o range de cada instrumento, ou seja, identificar a nota mais grave e a nota mais aguda de cada canal MIDI, para que se possa gerar uma lista ordenada de notas musicais por canal, onde a primeira nota será a mais grave e a última a mais aguda (ver item 4.4.1.1 e Capítulo 5, item 5.3);
7. Permitir que o cantor possa identificar, a qualquer momento, sua tessitura vocal e reconhecer com precisão quais as notas musicais limites de tal tessitura. Para tanto, o mesmo apenas deverá utilizar a ferramenta de transposição para ouvir os ranges de classificação vocal existentes (disponibilizado na ferramenta), identificando os limites que sua voz consiga reproduzir (ver item 4.4.1.1 – figuras 4.13, 4.14, 4.15 e Capítulo 5, item 5.1.2.1);
8. Definida a tessitura do cantor e escolhida a melodia de qual canal o mesmo deseja cantar, o sistema deverá ser capaz de transpor **automaticamente** todos os canais MIDI de forma a enquadrar a melodia escolhida (o canal MIDI) na tessitura (alcance, range vocal de conforto) do cantor. Identificado o range do

canal MIDI escolhido para cantar, bem como o range do cantor, o sistema poderá calcular quantos semitons deverá transpor a melodia para centralizar seu range no range do cantor, e, também, para manter todos os instrumentos da seqüência na mesma tonalidade da melodia. Deverá transpor todos os demais instrumentos (canais MIDI), menos a bateria, com o mesmo valor de semitons (ver item 4.4.1.1 – figuras 4.11, 4.12 e Capítulo 5, item 5.3 a 5.6);

9. Realizada a transposição, o sistema deverá ser capaz de gravar a nova seqüência de acordo com o nome de arquivo definido pelo cantor (ver item 4.4.1.1 e Capítulo 5, item 5.8).

Estes nove pontos básicos são os mínimos necessários para que se possa efetivar os objetivos propostos inicialmente nesta dissertação.

A seguir, serão analisadas as dificuldades e soluções inferidas para que se possa implementar um programa capaz de realizar as tarefas citadas.

4.3.1 Abrir um arquivo MIDI, separar os canais e identificar os eventos musicais

A princípio pode parecer ser esta uma operação simples, já que foi afirmado que em um arquivo MIDI todas as informações necessárias para se registrar uma música estão no mesmo explicitamente registrados. Realmente esta afirmação é verdadeira, mas, devido à simplicidade da estrutura da máquina MIDI (ver capítulo 2), as informações musicais contidas na mesma devem passar por filtros de análise para reconstituir as formas musicais necessárias para um sistema de análise de range instrumental, bem como para uma futura transposição [35].

Um outro problema relevante é a operação inversa, ou seja, após a separação e identificação dos eventos e realizada a transposição, reconstituir o arquivo MIDI, sem perder nenhuma informação pré-existente no mesmo, tais como direitos autorais, letra da música, nome da música, formato, etc., é uma tarefa tão ou mais complexa do que a primeira.

A figura 4.2 mostra uma partitura musical contendo quatro *tracks* de música, cada *track* com um canal MIDI (Instrumento) específico. Logo a seguir é apresentado, em código hexadecimal, o arquivo MIDI equivalente em formato 1 (cada canal MIDI é colocado em um *track* específico).

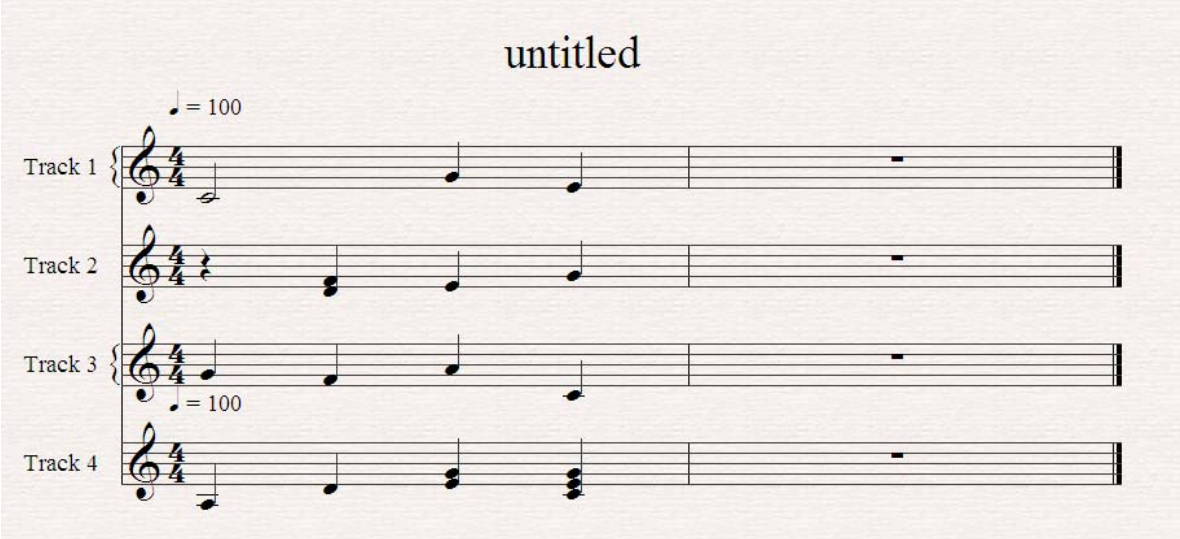


Figura 4.2 *- Partitura da música a ser analisada*

No primeiro *Track* da partitura[3] tem-se um compasso musical contendo:

- O compasso inicia com a nota C3 tocando com duração de uma mínima (duas semínimas);
- Logo após a nota G3 é ativada com duração de uma semínima;
- Depois a nota E3 é ativada com duração de uma semínima;
- O metrônomo[4] da música = 100 bpm.

No segundo *Track* da partitura tem-se um compasso musical contendo:

- O compasso inicia com uma pausa de uma semínima;
- Logo após tem-se as notas D3 e F3 tocadas simultaneamente com duração de uma semínima;
- Depois a nota E3 é ativada com duração de uma semínima;
- Depois a nota G3 é ativada com duração de uma semínima.

[3] O fato de um *track* na partitura ter uma numeração, não significa que o *track* MIDI terá a mesma numeração equivalente. Em MIDI, os canais e *tracks* iniciam pelo índice 0.
[4] Metrônomo – determina o número de uma figura musical especificada por minuto. O mesmo dita o pulso da música. Em MIDI, a figura musical adotada em todos os casos é a semínima.

No terceiro *Track* da partitura tem-se um compasso musical contendo:

- O compasso inicia com a nota musical G3 sendo ativada com a duração de uma semínima;
- Logo após a nota F3 é ativada com duração de uma semínima;
- Depois a nota A3 é ativada com duração de uma semínima;
- Depois a nota C3 é ativada com duração de uma semínima.

No quarto *Track* da partitura tem-se um compasso musical contendo

- O compasso inicia com a nota musical A2 sendo ativada com a duração de uma semínima;
- Logo após a nota D3 é ativada com duração de uma semínima;
- Logo após tem-se as notas E3 e G3 tocadas simultaneamente com duração de uma semínima;
- Logo após tem-se as notas C3, E3 e G3 tocadas simultaneamente com duração de uma semínima.

A seguir, apresenta-se o arquivo MIDI equivalente à partitura da Figura 4.2.

4D 54 68 64 00 00 00 06 00 01 00 05 00 A8 4D 54 72 6B 00 00 00 59 00 FF 03 08 75 6E 74 69 74 6C 65 64 00 FF 02 20 43 6F 70 79 72 69 67 68 74 20 A9 20 32 30 30 36 20 62 79 20 6C 75 63 69 61 6E 6F 20 6C 69 6D 61 00 FF 01 0C 6C 75 63 69 61 6E 6F 20 6C 69 6D 61 00 FF 58 04 04 02 18 08 00 FF 59 02 00 00 00 FF 51 03 09 27 C0 00 FF 2F 00 4D 54 72 6B 00 00 00 28 00 FF 03 07 54 72 61 63 6B 20 31 00 C0 00 00 90 3C 64 82 50 3C 00 00 43 64 81 28 43 00 00 40 64 81 28 40 00 00 FF 2F 00 4D 54 72 6B 00 00 00 2F 00 FF 03 07 54 72 61 63 6B 20 32 00 C1 18 81 28 91 41 64 00 3E 64 81 28 3E 00 00 41 00 00 40 64 81 28 40 00 00 43 64 81 28 43 00 00 FF 2F 00 4D 54 72 6B 00 00 00 2F 00 FF 03 07 54 72 61 63 6B 20 33 00 C3 0C 00 93 43 64 81 28 43 00 00 41 64 81 28 41 00 00 45 64 81 28 45 00 00 3C 64 81 28 3C 00 00 FF 2F 00 4D 54 72 6B 00 00 00 41 00 FF 03 07 54 72 61 63 6B 20 34 00 C6 34 00 96 39 64 81 28 39 00 00 3E 64 81 28 3E 00 00 40 64 00 43 64 81 28 43 00 00 40 00 00 43 64 00 40 64 00 3C 64 81 28 3C 00 00 40 00 00 43 00 00 FF 2F 00

Um sistema que analise o código MIDI registrado neste arquivo deverá obter os mesmos resultados da partitura grafada e detalhada anteriormente. Para tanto, a seguir será mostrado com detalhes como fazer a leitura e análise correta de um arquivo MIDI formato 1.

4.3.1.1 Identificando o cabeçalho principal do arquivo MIDI

O Cabeçalho principal de um arquivo MIDI possui um número fixo de bytes. No *track* principal, diferente dos demais *tracks* de música, utiliza-se os 8 bits[5] de cada byte.
O número de Bytes do cabeçalho do *track* principal[6] é 14 Bytes.
Assim, o cabeçalho deste arquivo é mostrado a seguir:

4D 54 68 64 00 00 00 06 00 01 00 05 00 A8

Onde:

4D 54 68 64	Caracteres **M (4D) T (54) h (68) d (64)**
00 00 00 06	Número de Bytes que faltam para completar o *track*
00 01	Formato do arquivo = formato 1
00 05	Número de *tracks* = 5
00 A8	Valor da ppq, do tempo relativo de uma semínima = $A8_H = 168_{10}$

***Tabela 4.1** – Cabeçalho de um arquivo MIDI*

O sistema projetado deve extrair deste cabeçalho as informações sobre o tipo de formato. Cada formato possui uma sintaxe diferente, ou seja, no formato zero tem-se um *track* musical (MTrk) para cada canal MIDI da seqüência, já no formato 1 se tem apenas um *track* musical para todos os canais.
Assim, a forma de armazenar as informações nestes dois tipos de formatos difere estruturalmente uma da outra.
Apenas o *track* principal MThd é sintaticamente igual em todos os formatos, de onde um programa de análise poderá extrair o valor da contagem de pulsos por semínima (ppq), formato e número de *tracks* musicais MTrk.
Através do valor da ppq lida pode-se inferir o valor dos tempos de cada nota musical grafada no arquivo MIDI, partindo de Bytes codificados no formato de deltaTimes.

As funções básicas, em CLEAN, que fazem a leitura e identificação da ppq, formato e número de *tracks* é:

[5] Nos *tracks* de música utilizam-se apenas os 7 bits menos significativos de cada Byte para dados, o oitavo é um sinalizador que informa se o Byte é um byte de status (1) ou um byte de dados (0).
[6] O *track* principal, o primeiro do arquivo MIDI, é denominado de MThd – MasterTrack head.

Formato x = x % (8,9) → **//obs.** lê-se: Pegue todos elementos da lista entre

NumerosTracks x = x % (10,11) // os índices 8 e 9

Ppq x = x % (12, 13)

Onde:

- **x** é uma lista contendo o arquivo binário MIDI lido e convertido para uma lista.
- **%** é uma função de CLEAN que pega elementos de uma lista entre os índices entre parênteses (inclusive)

De posse do conhecimento do número de *tracks* que o arquivo MIDI possui, o sistema, o programa em CLEAN, pode então separá-los. Para tanto, existem duas maneiras possíveis de se separar tais *tracks*:

1. No cabeçalho de cada *track* musical (denominado por MTrk), assim como no MThd, existe a informação de quantos bytes o mesmo possui. Desta forma, é só ler esta quantidade e extrair do arquivo *track* por *track*.
2. A outra opção é baseada no fato de que cada *track* MTrk possui uma mensagem de fim de arquivo contendo três bytes (FF 2F 00). Assim, pode-se efetuar a leitura do arquivo até que esta mensagem seja encontrada, e, desta forma, separar todos os *tracks* do arquivo.

A primeira opção é mais simples de ser implementada, principalmente em paradigma funcional.

Assim, o primeiro procedimento é eliminar da lista o *track* MThd já analisado, liberando o resto do arquivo para análise.

A função principal em CLEAN que faz isto é:

```
eliminaMThd  x = drop 14 x
```

Onde:

- **x** é uma lista contendo o arquivo binário MIDI lido e convertido para uma lista.

- **drop** é uma função do CLEAN que elimina um número especificado de elementos de uma lista qualquer.

A informação de quantos bytes um *track* possui está registrada nos bytes de índice 4 a 7 (quatro bytes) da lista, ou seja, quatro bytes logo após o nome MTrk (em ASCII[7]) que inicia todos os *tracks* musicais. Assim, um MTrk possui o número de Bytes definido por 4 bytes do cabeçalho mais 8 bytes (4 do MTrk e mais 4 da contagem).

A função principal em CLEAN, e suas auxiliares diretas, que devolve um *track* **MTrk** da lista de *tracks* é:

```
PegaTrack x =  take    (contagem +8)  x
where
      contagem = transforma2BemCont  qBytes
       qBytes = x  %  (4,7)
```

```
transforma2BemCont  n=:[y]        =  y
transforma2BemCont  n=:[z,y]      =  z*256 + y
transforma2BemCont  n=:[w,z,y]    =  w* 256*256   +   z*256   +    y
transforma2BemCont  n=:[t,w,z,y]  = t* 256*256 * 256 + w* 256*256 + z*256 + y
```

Onde:

- **qByte = x % (4,7)** é responsável por pegar os 4 bytes contendo o valor da quantidade de bytes restantes do MTrk corrente
- **contagem** é responsável por aplicar uma função (**transforma2BemCont qByte**) que converte n Bytes (de um a quatro bytes) em um valor inteiro.
- **contagem + 8** = número de bytes do *track* MTrk corrente
- **take contagem x** pega da lista x (o arquivo MIDI sem o *track* MThd),um número de bytes igual ao valor de **contagem** (o byte corrente)

A cada *track* lido, elimina-se o mesmo da lista e passa-se a analisar e separar um novo *track* do arquivo MIDI restante, utilizando a mesma função descrita anteriormente. O

[7] ASCII = **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange. Codificação de símbolos, caracteres, dígitos, e outras informações utilizando 127 símbolos ou 256 símbolos (ASCII entendido). Ver Anexo 1.

processo se repete até a lista do arquivo MIDI ficar vazia. Quando isto ocorre, a separação dos *tracks* estará pronta.

Este algoritmo só é utilizado para separar *tracks* de formato 1, onde cada *track* (trilha) já vem separado no arquivo SMF.

No caso de formato 0, o algoritmo é bem mais complexo, já que no mesmo só existe um *track* musical contendo todos os canais misturados.

O formato 0 é montado com os eventos MIDI sendo inseridos no arquivo SMF na seqüência exata com que as notas e eventos musicais vão ocorrendo.

Como o raciocínio da implementação de leitura dos dois formatos é semelhante, o mesmo não será apresentado neste documento para não torná-lo mais extenso ainda.

A função que elimina um *track* após ser lido é semelhante à função de leitura do mesmo, só que, ao invés de se pegar um número x de bytes contendo o *track* MTrk, deve-se eliminá-lo da lista.

```
EliminaTrack x =  drop    (contagem +8)  x
where
      contagem = transforma2BemCont  qBytes
       qBytes = x  %  (4,7)
```

Onde:

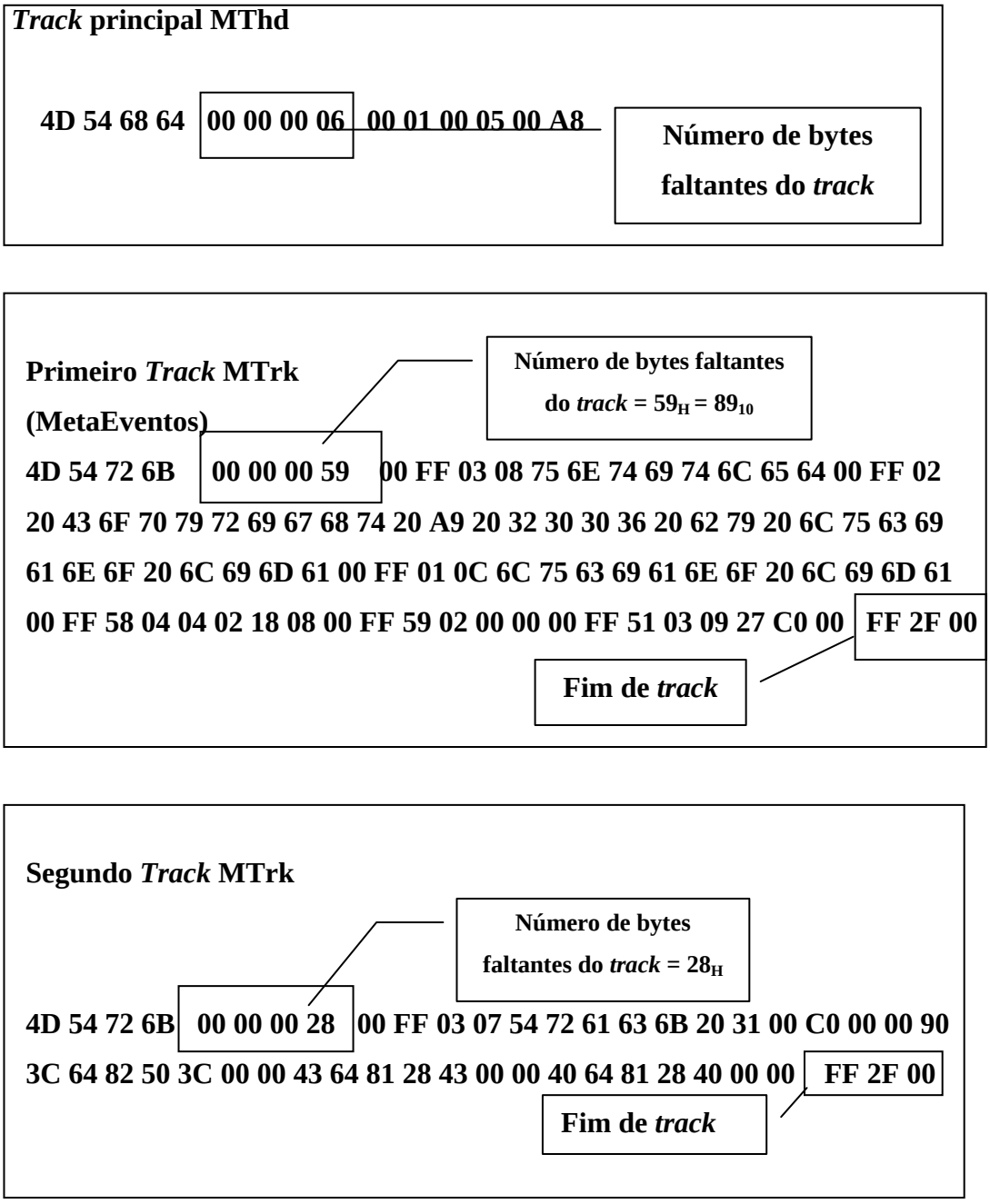
- **drop** é uma função do CLEAN que elimina um número especificado de elementos de uma lista qualquer.

O formato 1 possui uma particularidade de se ter um *track* MTrk a mais do que o número de *tracks* musicais, o qual é responsável por especificar alguns parâmetros musicais necessários à execução da música ou para grafia em partitura, tais como:

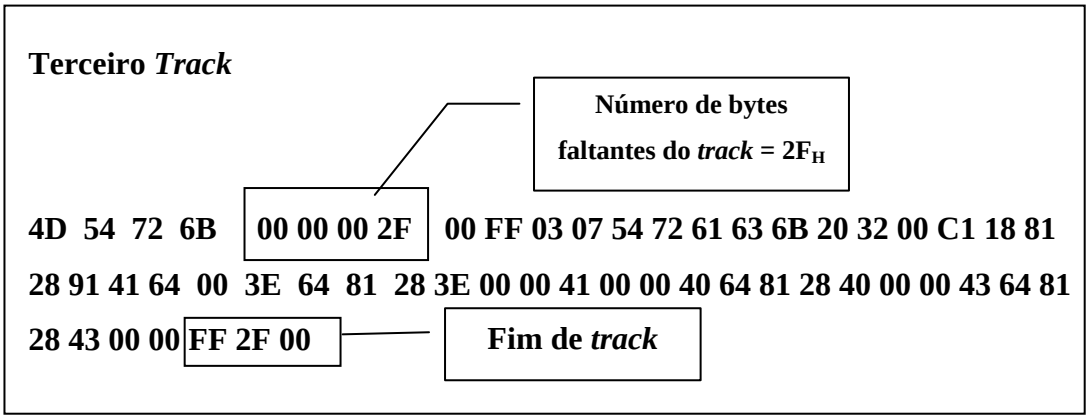
- valor do metrônomo
- tonalidade
- nome da música
- direitos autorais
- armadura de clave
- outros

Este *track* é sempre o primeiro após o *track* principal MThd. A leitura deste *track* é feita da mesma forma com que se procedeu na leitura dos demais *tracks* MTrk.

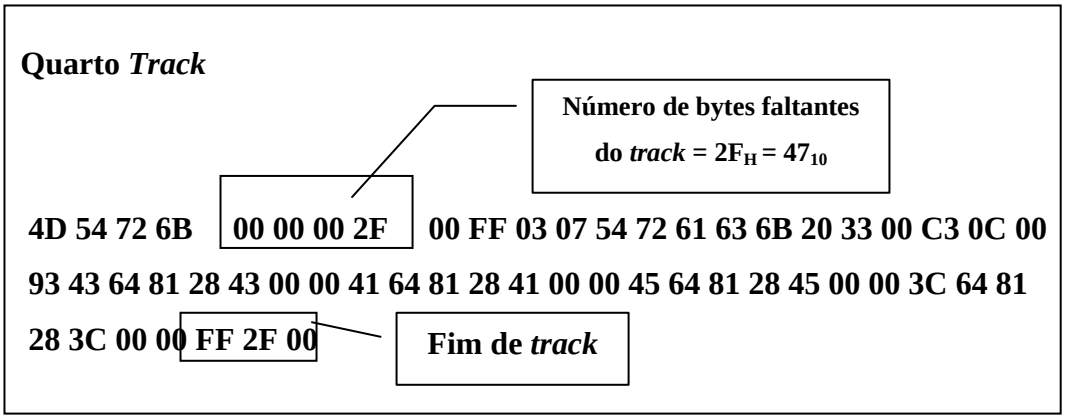
4.3.1.2 Separando os *tracks* do Arquivo MIDI de exemplo



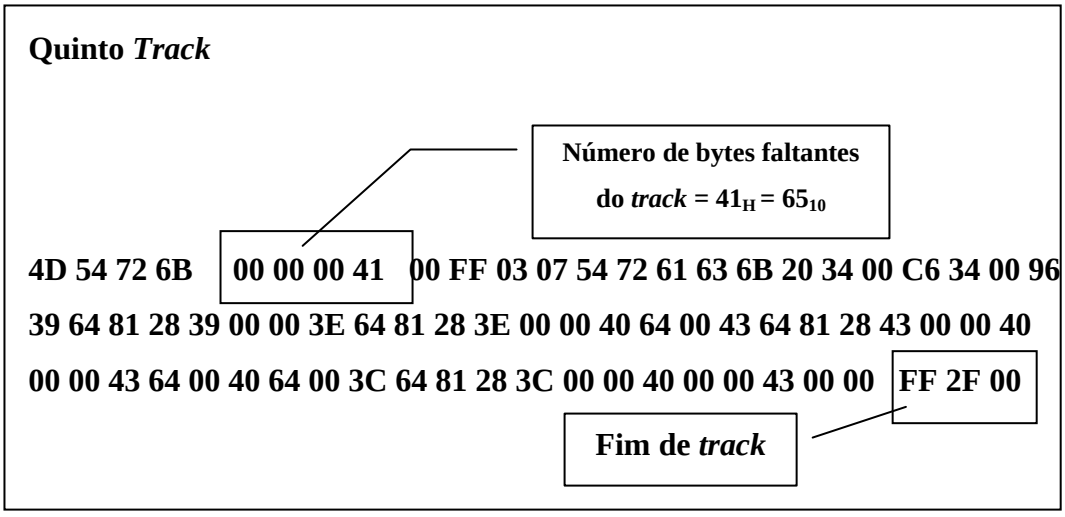
Terceiro *Track*



Quarto *Track*



Quinto *Track*



De posse dos *tracks* separados, cabe ao sistema agora separar e identificar os eventos existentes em cada um.

A análise dos *tracks* musicais é igual em todos os *tracks* MTrk, mesmo no primeiro que traz apenas mensagem contendo metaEventos, os quais são eventos que contêm informações estruturais da música.

4.3.1.3 Analisando o *track* de metaEventos (o *track* 1)

Os metaEventos são musicalmente bem descritos por MACHADO [19] em sua dissertação de mestrado, e formalizados computacionalmente por LOPES [22] também em sua dissertação de mestrado. Desta forma, para não repetir conceitos já suficientemente registrados nesta mesma instituição, será passado direto a implementação de como proceder tais análises.

A máquina MIDI, conforme mostrado no capitulo 2, é uma máquina que recebe um valor de contagem (denominada de deltaTime), e, logo a seguir, uma mensagem contendo uma instrução a ser executada por ela.

Esta contagem é precedida, portanto, de um deltaTime que pode ter de 1 a 4 Bytes.

O problema, portanto, na separação das mensagens MIDI, é determinar quando começa e quando finaliza um deltaTime, e, logo após, de se determinar qual é a mensagem MIDI contida no arquivo e quantos bytes de dados cada uma possui (este valor pode variar desde 0 Bytes até 2139062271 Bytes).
Assim, elimina-se de cada *track* musical 8 bytes correspondendo aos 4 bytes contendo a palavra MTrk e mais 4 bytes contendo a contagem de quantos bytes o *track* ainda possui após o mesmo.
A função que faz isto é:

```
Elimina8Bytes x = drop 8 x
```

Análise e identificação dos MetaEventos do primeiro *Track*:

4D 54 72 6B 00 00 00 59

00	**DeltaTime = 0** -> o evento seguinte deve ser executado instantaneamente, sem nenhuma espera.
FF 03	**MetaEvento de Título**
08	O meta evento possui 8 bytes
75 6E 74 69 74 6C 65 64	**untitled - u=75,n=6E,t=74,i=69,t=64,l=6C,e=65,d=64**
00	**DeltaTime = 0**
FF 02	**MetaEvento de Direito Autoral**
20	O meta evento possui 32 bytes (20_H)
43 6F 70 79 72 69 67 6 8 74 20 A9 20 32 30 30 36 20 62 79 20 6C 75 63 69 61 6E 6F 20 6C 69 6D 61	**Copyright .....**
00	**DeltaTime = 0**
FF 01	**MetaEvento de Texto**
0C	O meta evento possui 12 Bytes ($0C_H$)
6C 75 63 69 61 6E 6F 20 6C 69 6D 61	**luciano lima. l=6C,u=75, ...**
00	**DeltaTime = 0**
FF 58	**Fórmula de Compasso**
04	O meta evento possui 4 bytes
04 02 18 08	**Fórmula 4 por 4 (2^2)**
00	**DeltaTime = 0**
FF 59	**Meta Evento de Armadura de Clave**
02	O meta evento possui 2 bytes
00 00	**Tonalidade de Dó maior**
00	**Delta Time = 0**
FF 51	**Meta Evento de Tempo**
03	O meta evento possui 3 bytes

09 27 C0	**Metrônomo = 100 bpm**[8] **=** Este meta evento informa o tempo de uma semínima em microssegundos. Assim, se tem um tempo de $0927C0_H$ = 600.000 microssegundos = 0,6 segundos -> **metrônomo** = 60s/0,6s = 100 bpm.
00	**Delta Time = 0**
FF 2F 00	**Meta Evento de fim de *track*** FF 2F com 00 bytes de dados

***Tabela 4.2** – Análise e identificação dos MetaEventos do primeiro Track*

Do resultado desta análise pode-se criar uma lista, mostrada a seguir, contendo o *track*, onde cada elemento da lista é uma lista contendo um evento.

[

[00,FF,0,08,75,6E,74,69,74,6C,65,64],

[00,FF,02,20,43,6F,70,79,72,69,67,68,74,20,A9,20,32,30,30,36,20,62,79,20,6C,75,63,69,61,6E,6F,20,6C,69,6D,61],

[00,FF,01,0C,6C,75,63,69,61,6E,6F,20,6C,69,6D,61],

[00,FF,58,04,04,02,18,08],

[00,FF,59,02,00,00],

[00,FF,51,03,09,27,C0],[

00,FF,2F,00]

]

Uma função genérica em CLEAN que busca um determinado evento (uma mensagem de MetaEvento, por exemplo) em uma lista MTrk, como a descrita aqui, é mostrada a seguir:

```
PegaMetaEventoGen  lt me = [ x \\ x<- lt  | (lt!!2) == me]
```

Onde:

[8] bmp – batidas por minuto

- **!!** é uma função do Clean que pega um elemento de uma lista, conforme o índice fornecido logo após esta função.
- **lt!!2** retorna o terceiro elemento da lista, ou seja, o elemento de índice 2.
- **[x \\ x<- lt | (lt!!2) == me]** retorna todos as listas de mensagens pertencentes a **lt** que obedeçam à regra de que o terceiro elemento da lista de mensagem seja igual ao código do metaEvento solicitado (**me**).

Observe que a implementação em CLEAN desta função de busca é extremamente aderente à formalização lógica do problema proposto.

Regra geral dos metaEventos

Existem vários outros tipos de metaEventos, todos possuindo a mesma sintaxe, ou seja, inicia por um Byte = FF (255), seguido de um byte que especifica qual é o metaEvento em questão, seguido do número de bytes de dados (segue a mesma regra do deltaTime de 1 a 4 Bytes), seguido dos bytes de dados do metaEvento.

Analisando os *tracks* de música

O princípio é o mesmo da análise do *track* de metaEventos, o qual é, também, um MTrk.

Assim, após eliminar os 8 (oito) primeiros Bytes do *track*, passa-se à análise do *track* musical, iniciando por identificar o primeiro deltaTime, seguido do primeiro evento e assim sucessivamente até que se atinja o fim de *track* com mensagem de metaEvento FF 2F 00.

A seguir é feita uma análise do *track* 2, canal 0, para exemplificar como o processo é feito.

4D 54 72 6B 00 00 00 28

00	**Delta Time = 0**, isto significa que o evento seguinte deve ser iniciado imediatamente
FF 03	**Meta Evento de Título**
07	Meta Evento com 7 Bytes
54 72 61 63 6B 20 31	***Track* 1, T=54, r=72, …..**
00	**Delta Time = 0**
C0 00	**Mensagem de dois bytes de mudança de instrumento (C0 -> C = mudança de instrumento, 0 = canal 0) = piano acústico[9] (00)**
00	**Delta time = 0**
90 3C 64	**Mensagem de três bytes de ativação de nota no canal 0 (9 = ativa nota, 0 = canal 0). A nota a ser ativada é a $3C_H$ (60_{10})[10] que é a nota C3 (o dó central do piano). O volume da mesma deverá ser $64_H(100_{10})$**
82 50	**Meta Evento com dois Bytes, já que o primeiro byte possui valor superior a $7F_H$ (127_{10}). Para calcular o valor real do tempo em ppqs, deve-se transformar o Delta Time em ppqs, ou seja, transformar a informação de tempo com 7 bits por Byte em uma com 8 bits por Byte. Assim, tem-se** **8 2 5 0** **1000 0010 0101 0000** **Para tanto, elimina-se os bits mais significativos e remontam-se os Bytes com os oito bits, ficando** **0 1 5 0** **0000 0001 0101 0000** O valor, portanto, do tempo, é de **$0150_H = 336_{10}$** **Como uma semínima vale 168, isto significa que o**

[9] Ver tabela de código de instrumentos no Anexo 2.
[10] Ver tabela de códigos de notas musicais no Anexo 3.

	tempo deste delta time equivale ao tempo de duas semínimas (uma mínima), ou seja, deve-se esperar o tempo de duas semínimas para executar o próximo evento MIDI.
3C 00	**Mensagem sem o Byte de status que define a ação do evento. Um byte de status deve iniciar com o bit mais significativo setado em 1, ou seja, com um byte maior ou igual ao valor $80_H(128_{10})$. No caso, o evento começou com um valor igual a 3C, inferior a 80. Quando isto ocorre, adota-se o último status declarado como sendo o status do evento. A isto denomina-se Running Status[11]. No caso, o último status foi 90, ou seja, ativar uma nota no canal zero.** **Esta mensagem, sem o running status, ficaria assim: 90 3C 00, ou seja, ativar a nota C3 ($3C_h$) com o volume igual a 0, ou seja, desativar a nota C3. Ativar nota com volume zero corresponde a se desativar esta nota. O código utilizado para desativar uma nota no canal zero é o código 80, no caso, a mensagem sem running status ficaria: 80 3C 00.**
00	**Delta Time = 0**
43 64	**Novamente a mensagem começa com um byte menor que 80_H. Isto significa que novamente estar-se-á utilizando o running status, ou seja, o último status utilizado, o qual continua sendo o 90_H. Assim, esta mensagem completa seria: 90 43 64, ou seja, ativar a nota G3 (43_H) imediatamente com o volume igual a 64 (100_{10}).**
81 28	**Delta Time com dois bytes, já que o primeiro Byte é**

[11] **O running status foi criado para economizar espaço de memória, o que era um fator importante nos anos 80.**

	maior que 80_H e o segundo menor. Assim, tem-se **8 1 2 8** **1000 0001 0010 1000** **Para tanto, elimina-se os bits mais significativos e remontam-se os Bytes com os oito bits, ficando** **0 0 A 8** **0000 0000 1010 1000** O valor, portanto, do tempo, é de **$A8_H$ = 168_{10},** cujo valor é o valor de uma ppq, ou seja, do tempo de uma semínima.
43 00	Novamente uma mensagem com running status, no caso o 90_H, gerando uma mensagem igual a **90 43 00,** que significa que se deve desativar a nota musical 43_H (G3).
00	**Delta time = 0**
40 64	Mensagem com running status, novamente o 90_H, indicando que se deve ativar a nota 40_H (E3) com o volume 64_H (100_{10}).
81 28	**Delta Time com dois bytes, já que o primeiro Byte é maior que 80_H e o segundo menor. Assim, tem-se** **8 1 2 8** **1000 0001 0010 1000** **Para tanto, elimina-se os bits mais significativos e remontam-se os Bytes com os oito bits, ficando** **0 0 A 8** **0000 0000 1010 1000** O valor, portanto, do tempo, é de **$A8_H$ = 168_{10},** cujo valor é o valor de uma ppq, ou seja, do tempo de uma semínima.
40 00	**Mensagem com running status = 90_H,** indicando que se deve ativar a nota E3 (40_H) com volume zero, ou

	seja, desativar tal nota.
00	**Delta Time = 0**
FF 2F 00	**Meta evento de Fim de *Track***

Tabela 4.3 *– Análise e identificação dos MetaEventos do segundo Track*

A análise mostra que este *track* informa que se deve iniciar ativando uma nota musical C3 com um tempo de uma mínima (duas semínimas), logo a seguir ativar uma nota musical G3 com o tempo de uma semínima, e, para finalizar, ativar a nota musical E3 durante o tempo de uma semínima, confirmando o que está grafado na partitura exemplo e na análise da mesma feita anteriormente.

A partir da análise das mensagens deste *track*, o sistema pode montar uma lista de lista de eventos MIDI, com os running status eliminados contendo o mesmo, conforme mostrado a seguir:

[

[00,FF,03,07,54,72,61,63,6B,20,31],

[00,C0,00],

[00,90,3C,64],

[8250, 90,3C,00],

[00, 90,43,64],

[8128, 90,43,00],

[00, 90,40,64],

[8128, 90,40,00],

[00,FF,2F,00,

]

Montadas as listas de eventos, no formato apresentado, onde o DeltaTime é o primeiro elemento e o status é o segundo, fica simples de se implementar funções de análise e reconhecimento das mensagens MIDI (deltaTime + evento). Para exemplificar, a seguir é mostrada a implementação de uma função genérica que retorna uma lista contendo todas as mensagens especificadas no argumento da mesma. A função é **PegarEvento.** O primeiro argumento da mesma é a lista **lt** de *tracks* e **ev** é o evento desejado.

```
PegarEvento  lt ev = [ x \\ x<- lt  | (lt!!2) == ev]
```

Onde:

- **!!** é uma função do Clean que pega um elemento de uma lista, conforme o índice fornecido logo após esta função.
- **lt!!2** retorna o terceiro elemento da lista, ou seja, o elemento de índice 2.
- **[x \\ x<- lt | (lt!!2) == ev]** retorna todos as listas de mensagens pertencentes a **lt** que obedeçam à regra de que o terceiro elemento da lista de mensagem seja igual ao código do Evento solicitado (**ev**).

Pode-se perceber que realmente as informações estão todas declaradas nos arquivos SMF, mesmo que implicitamente.
Percebe-se, também, um relativo grau de complexidade para "ensinar" o computador a reconhecer, extrair e montar as listas de eventos musicais contidas nos mesmos.
A maior dificuldade, como ficou claro, está em se obter os conhecimentos necessários para que se possa reconhecer todos os tipos de eventos que um arquivo MIDI pode utilizar, para que, posteriormente, funções simples como as que foram mostradas aqui como exemplo, possam realizar as tarefas pretendidas.

4.4 IHM (Interface Homem Máquina) – a interface com o usuário

A maioria dos usuários alvo do sistema proposto nesta dissertação são usuários leigos em computação. Os mesmos possuem relativa dificuldade na manipulação de aplicativos complexos, principalmente quando os sistemas possuem um número significativo de menus, submenus, links e outros recursos interativos que as interfaces visuais possuem atualmente.

Realizada uma pesquisa com os usuários alvos, principalmente músicos da noite, verificou-se que os mesmos possuem pouco tempo e interesse no uso do computador, a não ser para jogos e atividades recreativas sociais (msn, orkut e outros afins).

Assim, desta pesquisa conclui-se que criar uma interface com menus e sub-menus fogem do paradigma de soluções e aplicações que tais profissionais utilizam no dia a dia.

Uma interface simples, aderente de ser utilizada, para tais grupos, seria uma interface apenas com poucos botões, cada um com uma função distinta e específica. Tais interfaces não podem possuir elementos que permitam ações diferentes que necessitem do julgamento do usuário, bem como a mesma não deverá possuir campos de texto onde o usuário tenha que escrever ações que o sistema deverá fazer, evitando que erros na grafia venham a travar o sistema, ou, na melhor das hipóteses, não efetuar o que se quer.

Com base nestas informações e pesquisas, prototipou-se várias interfaces, sendo que a que mais agradou e foi de encontro aos objetivos traçados é a que é apresentada na figura 4.3, onde o sistema implementado foi denominado de BEST VOCAL 2005.

Figura 4.3 *- Interface do Best Vocal 2005*

Estrutura geral da interface

Conforme pode-se ver na figura 4.3, a mesma possui dois conjuntos distintos de botões, cada um com uma funcionabilidade distinta, a saber:

- Gerenciamento de arquivos MIDI formato 0 ou formato 1
- Análise, modificação e transposição dos arquivos MIDI

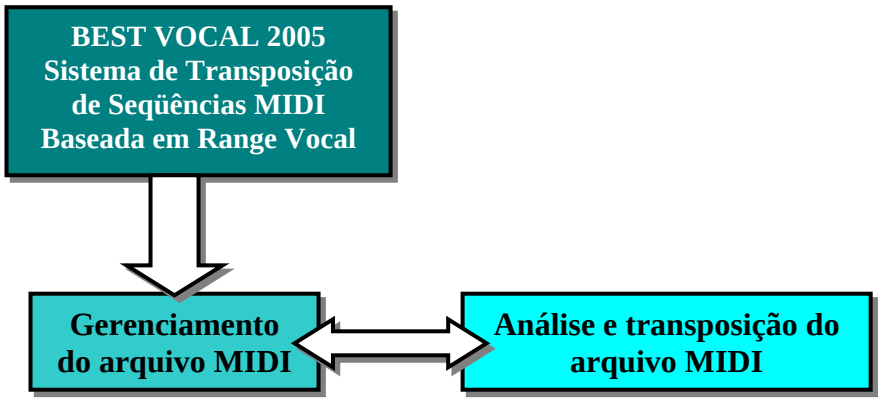


Diagrama 4.1 **– Sistema de transposição de seqüências MIDI baseada em range vocal**

A ferramenta de gerenciamento possui quatro potencialidades: Abrir, Salvar, Tocar e Interromper a execução de um arquivo MIDI, seja em formato 0 ou formato 1.

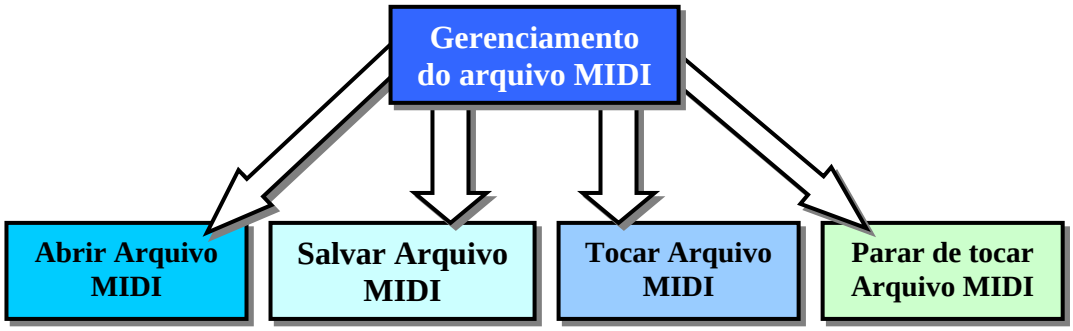


Diagrama 4.2 *– Gerenciamento do arquivo MIDI*

A ferramenta de análise e modificação dos arquivos MIDI, possui também quatro potencialidades básicas: letra da música, tessitura dos instrumentos, substituição de instrumentos da música, Identificação de range vocal do cantor/transposição da música.

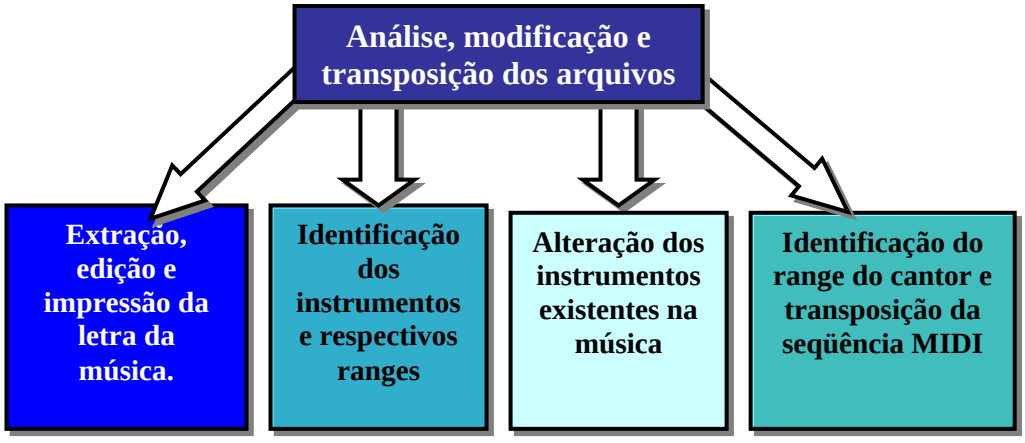


***Diagrama 4.3** – Análise, modificação e transposição dos arquivos MIDI*

Para implementação da interface de entrada, optou-se por construir uma interface principal **MDI**[12] (**M**ultiple **D**ocument **I**nterace) que permitisse que as janelas de todas as ferramentas ficassem abertas e ativas ao mesmo tempo, já que, como será visto ainda neste capítulo, é imprescindível que as ferramentas e respectivas janelas fiquem abertas ao mesmo tempo para que se possa efetivar uma transposição adequada e de uma forma mais simples e intuitiva.

Na implementação das janelas das ferramentas, optou-se pela criação de interfaces **NDI** (**N**o **D**ocument **I**nterface), as quais são simples de serem criadas e facilitam e flexibilizam a construção e retrabalho no layout (as mesmas se auto-ajustam com o layout dos botões e campos de texto).

A seguir são apresentadas, detalhadamente, as ferramentas desenvolvidas.

4.4.1 Descrição da utilização da interface e do sistema de transposição

4.4.1.1 Potencialidades e características

O sistema desenvolvido permite ao usuário realizar as seguintes tarefas:

1-Abrir um arquivo MIDI, seja em formato 0 ou formato 1

Para se utilizar o Best Vocal 2005, é necessário, em primeiro lugar, abrir um arquivo MIDI qualquer.

Apesar de ser uma tarefa complexa, o sistema desenvolvido permite que o usuário abra arquivos MIDI de qualquer formato, evitando que o mesmo tenha que se preocupar em

[12] A implementação deste tipo de interface, em CLEAN, pode ser visto com detalhes no Anexo 4 e na dissertação deo trabalho de mestrado de MARTINS em seu trabalho de mestrado na FEELT-UFU [36].

qual formato o arquivo foi gerado, simplificando a tarefa dele, e, conforme projeto, evitando que tenha que tomar decisões conflitantes e muitas vezes complexas para ele (já que poucas pessoas compreendem as diferenças e aplicações das implementações dos vários formatos de arquivo MIDI).

Enquanto um arquivo não for aberto, nenhuma das demais ferramentas funcionará.

Para abrir um arquivo MIDI, portanto, basta clicar no botão Abrir. Ao fazer isto, o gerenciador de arquivos é aberto para que o usuário escolha um arquivo. Se o arquivo escolhido for um arquivo MIDI válido, o sistema emite a mensagem mostrada na figura 4.4, caso contrário, o mesmo emite uma mensagem de erro.

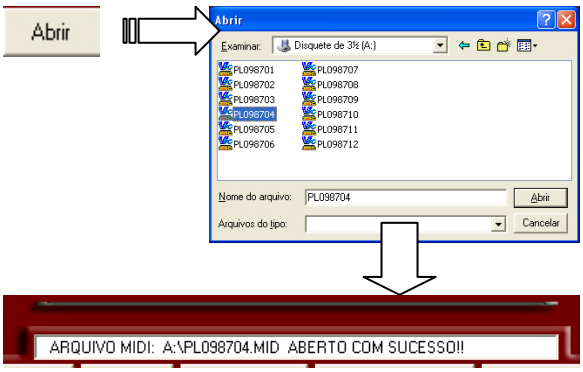


Figura 4.4 *- Abrindo um arquivo MIDI qualquer*

2-Tocar ou interromper a execução do arquivo aberto ou do modificado

Para executar um arquivo MIDI, o sistema detecta a placa de som ativa do computador e a utiliza, sem que seja necessário ao usuário configurar a mesma no sistema. Esta é uma característica que atende a estrutura do projeto que é a de simplificar a utilização do sistema. Assim, para tocar um arquivo MIDI, basta, depois de abri-lo, é claro, clicar no botão Tocar e para interromper a execução, basta clicar no botão Parar (ações bastante intuitivas).

3-Eliminar pausas existentes no início do arquivo

Este recurso foi bastante solicitado pelos usuários que testaram o sistema, e, apesar de não ser relevante para o objetivo desta dissertação, ou seja, a transposição automática da seqüência MIDI, é uma ferramenta bastante útil e atrativa para o público alvo. Na realidade, muitos usuários adquiriram o programa especialmente para utilizar esta ferramenta em suas seqüências, já que a mesma não existe nos softwares comerciais existentes no mercado.

Eliminar pausas é uma tarefa tão ou mais complexa do que transpor uma seqüência musical. Eliminar pausas em arquivos do tipo Wave é uma tarefa simples, ou seja, é só retirar a região inicial e final do arquivo onde a amplitude do mesmo estiver abaixo de um determinado valor.

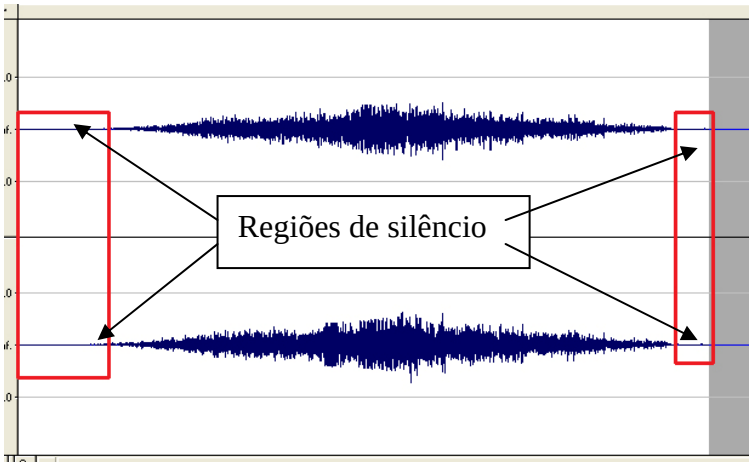


Figura 4.5 *- Arquivo wave com silêncio no início e fim do arquivo*

Já em um arquivo MIDI, não existe uma região explícita de silêncio. Um silêncio em um arquivo MIDI é um Delta Time diferente de zero em algum evento existente antes da primeira nota musical ser ativada em alguns dos 16 canais possíveis em uma seqüência MIDI. Uma pausa no fim de um arquivo MIDI também é um Delta Time diferente de zero em algum evento após a última nota musical ser desativada. Desta forma, um programa que se destine a fazer isto tem que analisar todos os eventos de um arquivo MIDI e zerar todos os delta time existentes antes desta primeira nota em um dos canais do arquivo, bem como, também, zerar o Delta Time de todos os eventos que existirem em todos os canais MIDI após a última nota de um destes canais ser desativada.

O paradigma e linguagem de programação escolhida facilitam bastante esta tarefa, já que, para tanto, pode-se utilizar novamente a notação Zermelo-Frankel para definir o domínio (a lista de *tracks* contendo as listas de eventos) e as restrições do mesmo (deltaTimes diferentes de zero antes da primeira nota de qualquer lista de *track* ser ativada), gerando o conjunto solução (uma nova lista de *tracks* com os deltaTimes zerados de todas as listas de eventos antes da primeira nota de qualquer canal ser ativada).

O **Best Vocal**, portanto, examina todos os canais MIDI existentes no arquivo, identifica a pausa existente no início[13] da execução musical de cada um deles e elimina a menor pausa encontrada em todos os canais. Assim, o arquivo salvo, com pausas iniciais removidas, ao ser executado é iniciado imediatamente.

Para utilizar esta ferramenta, basta clicar no botão **Eliminar Pausa**. Quando o sistema termina de realizar a ação, uma mensagem de sucesso é emitida, conforme mostrado na figura 4.6.

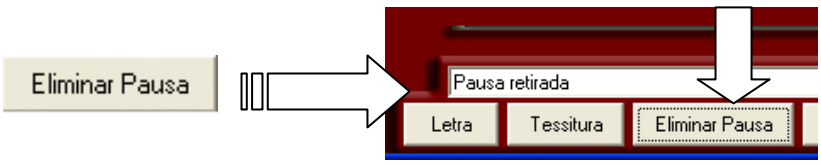


Figura 4.6 *- Eliminando Pausas*

4-Visualizar, salvar e imprimir a letra da música contida no arquivo

Esta ferramenta também não é necessária para que se cumpram os objetivos do projeto. A mesma foi implementada, também, pela solicitação dos usuários do programa sob teste.

[13] Não se implementou a eliminação de pausas finais, já que as mesmas raramente são encontradas em seqüências profissionais e na maioria da amadoras.

Já que o objetivo do sistema é transpor uma seqüência musical MIDI para uma tonalidade que o cantor consiga cantar sem esforço, bem como, também, já que muitos arquivos MIDI possuem a letra da música (lirismo[14]) incorporada no mesmo, permitir que o usuário possa visualizar e imprimir a letra da música é uma potencialidade aderente ao projeto e que o torna um produto também mais atrativo, principalmente para os cantores que não possuem uma boa memória.

Para ativar esta ferramenta, basta clicar no botão **Letra**. Após alguns segundos, a letra aparece em uma interface NDI. Detalhes do uso desta ferramenta pode ser visto no diagrama a seguir e na figura 4.7.

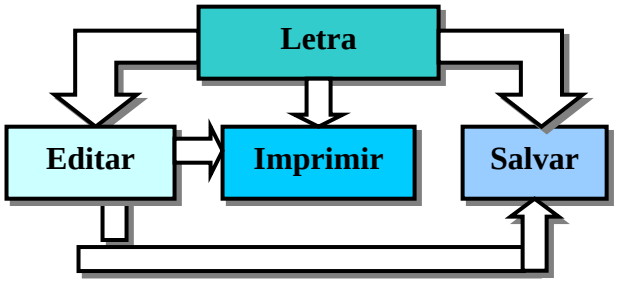


Diagrama 4.4 *– Letra*

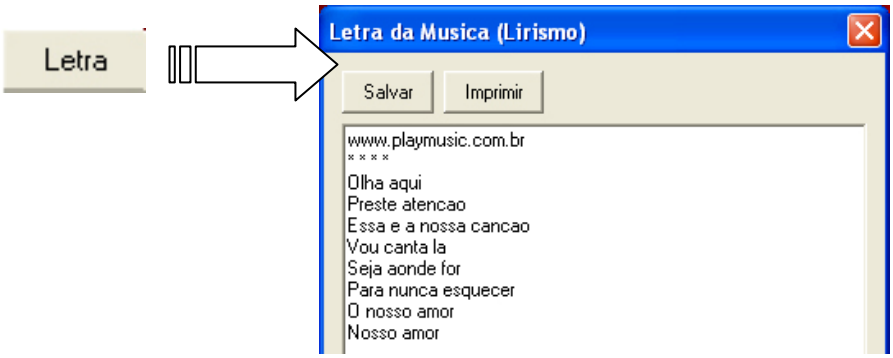


Figura 4.7 *– Janela da ferramenta Letra da Música*

Esta ferramenta (**Letra da Musica**) permite ao usuário editar a letra da música. Como o texto é editável também permite que se acrescente acordes logo acima das palavras e outras características personalizadas que se desejar.

[14] Lirismo – Embora na língua portuguesa esta palavra, traduzida literalmente do inglês Lyrics, dê dupla conotação, vem sendo aceita erroneamente por músicos, principalmente por amadores, como sinônimo de letra de uma música.

Figura 4.8 *– Acréscimo de acordes acima das palavras*

O profissional pode salvar a letra em arquivo texto ou imprimí-la direto da janela. Para isto, basta que ele clique no botão 

ou no botão 

.

5-Visualizar, salvar e imprimir o range (Tessitura) de cada instrumento (canal MIDI) do arquivo

Ao se analisar um arquivo MIDI para determinar as notas limites utilizadas em cada canal, a definição de range e tessitura se confundem, já que em MIDI são utilizados instrumentos virtuais, sintetizados, e, neste caso, os mesmos sempre terão uma qualidade tão boa quanto se queira ou tão boa quanto for a qualidade do sintetizador.

Para realizar esta tarefa, o programa deve implementar uma função que retorne todas as notas de cada canal e faça uma ordenação das mesmas. Feito isto, ter-se-á uma lista contendo mensagens iniciadas pelo status de 90 a 9F (ativar nota no canal 0 ao canal 15). Para determinar o range de cada canal, basta pegar, na lista gerada por canal, a primeira e última nota musical.

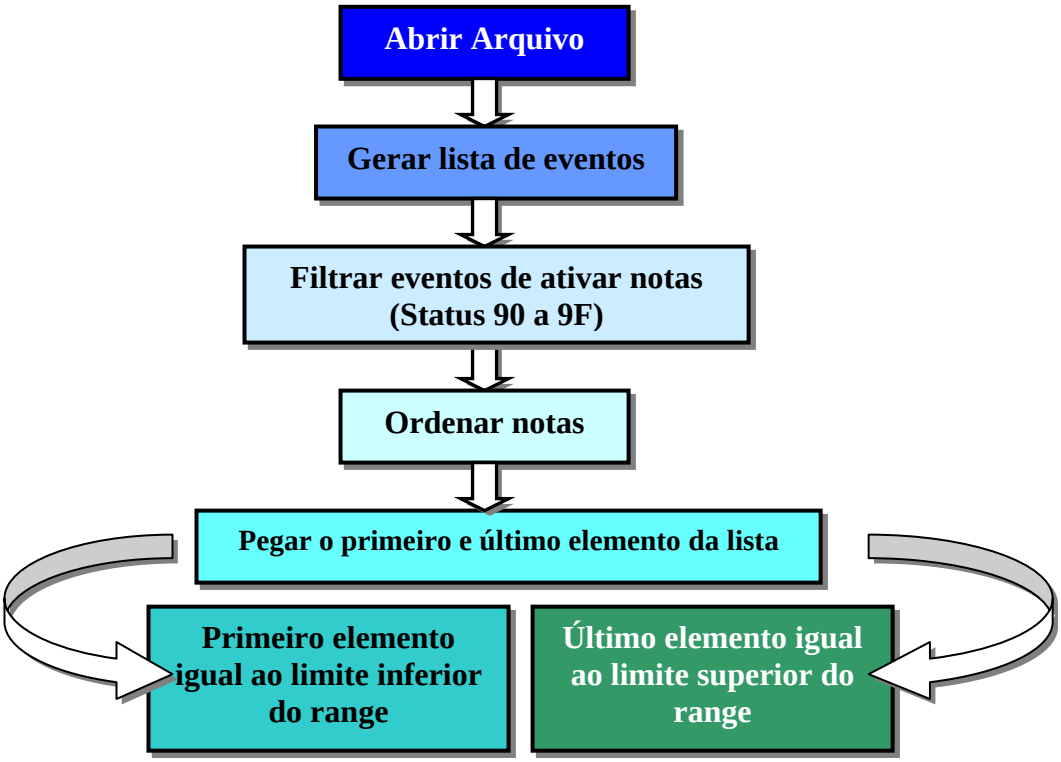


Diagrama 4.5 *– Detecção do range dos canais MIDI*

```
range lt = [primeiro, ultimo]
where
   primeiro = hd (sort [x!!2 \\ x <- lt | isMember x [ "90", "91", "92",
                        "93", "94", "95", "96", "97", "98", "99", "9A",
                        "9B", "9C", "9D", "9E", "9F"]])
   ultimo = last (sort [x!!2 \\ x <- lt | isMember x ["90", "91", "92",
                        "93", "94", "95", "96", "97", "98", "99", "9A",
                        "9B", "9C", "9D", "9E", "9F"]])
```

Obs: x!!2 – Código da nota ativada

hd = função que pega o primeiro elemento de uma lista

last = função que pega o último elemento de uma lista

sort = função que ordena uma lista

Para determinar o instrumento do canal, basta, novamente, implementar uma função utilizando a notação Zermelo-Frankel que devolva somente as mensagens de escolha ou mudança de instrumento (status de C0 a CF). Feito isto, é só ver na tabela de instrumentos MIDI (Anexo 2) qual instrumento corresponde ao código da mensagem. A função, a seguir, ilustra como gerar, em CLEAN (Anexo 4), uma lista contendo

apenas as mensagens de instrumentos (denominada no padrão MIDI por Program Change).

Para tanto, utiliza-se a função **isMember** do CLEAN para checar se um dado elemento pertence ou não a uma determinada lista.

Dada uma música representada, em CLEAN, por uma lista de eventos MIDI:

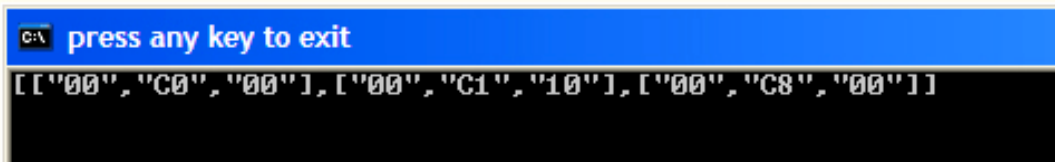
```
arqMIDI = [
 ["00","C0","00"],["00","90","3C","64"],["8250"," 90","3C","00"],
 ["00"," 90","43","64"],["8128","90","43","00"],["00","FF","2F","00"],
 ["00","C1","10"],["00","91","20","64"],["8250"," 91","20","00"],
 ["00"," 91","23","64"],["8128"," 91","23","00"],["00","FF","2F","00"],
 ["00","C8","00"],["00","98","4C","64"],["8250"," 98","4C","00"],
 ["00"," 98","44","64"],["8128"," 98","44","00"],["00","FF","2F","00"]
 ]
```

A função que devolve os eventos MIDI de mudança de instrumento, fica:

```
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
pegaEventoInst lt = [ x \\ x <- lt  | isMember (x!!1) eventoInstrumento ]
where
    eventoInstrumento = ["C0","C1","C2","C3","C4","C5","C6","C7",
                         "C8","C9","CA","CB","CC","CD","CE","CF"]
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

Onde se pode ler: A função ***pegaEventoInst*** devolve uma lista de elementos **x** tal que **x** pertence à lista **lt**, quando o segundo elemento da lista (**x!!1**) for membro (**isMember**) da lista **eventoInstrumento**, ou seja, quando o evento for um evento de **program change** ($C0_H$ a CF_H).

Executando esta função (***pegaEventoInst lt***) no CLEAN, com **lt = arqMIDI** (**Start = pegaEventoInst arqMIDI**), tem-se:



Para ativar a ferramenta de análise de tessitura, basta clicar no botão **Tessitura.** Ao fazer isto, alguns segundos após a janela **NDI** é aberta contendo o range (tessitura) de cada canal MIDI, bem como o nome de cada instrumento dos respectivos canais. Detalhes do uso desta ferramenta pode ser visto no diagrama a seguir e na figura 4.9.

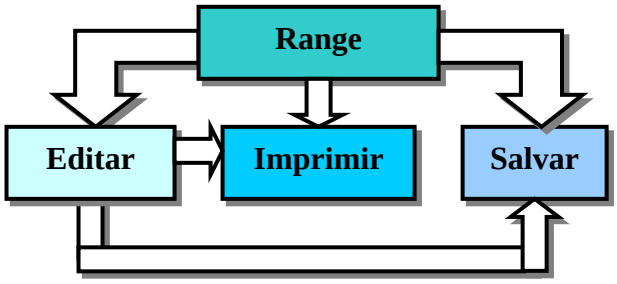


Diagrama 4.6 *– Range*

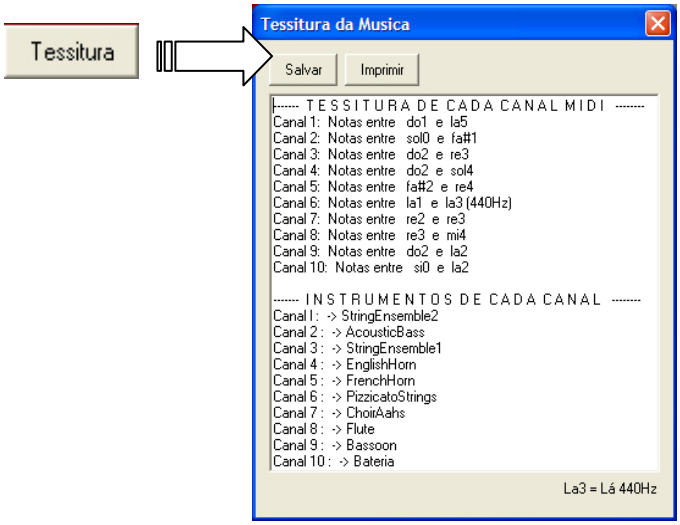


Figura 4.9 *– Range dos instrumentos*

O conhecimento do range e dos instrumentos de cada canal são de fundamental importância para que o cantor possa identificar qual melodia e de qual canal, deseja cantar.

O usuário, novamente, pode optar por salvar os ranges em arquivo texto ou imprimir direto da janela.

6-Visualizar e mudar os instrumentos de cada canal MIDI do arquivo (cada arquivo pode possuir até 16 canais MIDI)

Em alguns programas como o Finale ou mesmo o Sibelius, mudar um instrumento de um determinado canal ou staff, em arquivos já criados, não é uma tarefa tão simples para a maioria dos usuários. No **Best Vocal** esta tarefa é bastante simples, bastando ao músico clicar em um botão, escolher, em um pop-up, o instrumento que quer mudar e

escolher o instrumento que vai substituí-lo. Apesar de não ser uma ferramenta fundamental para o que o programa se propõe: **Transpor músicas para uma região confortável e adequada a um cantor ou determinado instrumento solo,** a ferramenta é bem vinda por, novamente, ser simples de ser utilizada.

Para utilizar esta ferramenta, basta clicar no botão **Mudar Instrumentos**. Ao fazer isto, uma janela com dois pop-ups é aberta para que o usuário escolha o instrumento que quer modificar e o novo instrumento que ficará em seu lugar, conforme seqüência apresentada na figura 4.10.

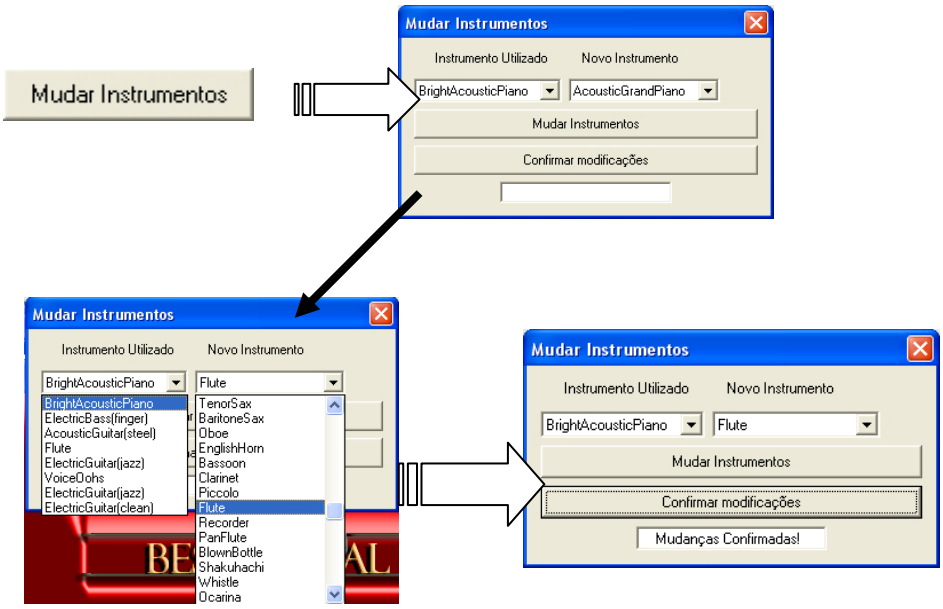


Figura 4.10 *- Mudança de instrumentos*

7-Transpor a música conforme tessitura vocal do músico ou instrumento que deseja utilizar com a música de playback, de acordo com a melodia de um determinado canal MIDI existente no arquivo

Geralmente um músico faz um teste de classificação vocal com profissionais da área, tais como: fonoaudiólogos e especialistas em canto. Este tipo de teste visa descobrir qual é o melhor range de conforto vocal (tessitura) para um determinado cantor cantar, sem desafinar, falhar ou forçar suas pregas vocais.

O que ocorre é que este range identificado pelo profissional nem sempre é o mesmo em qualquer hora do dia, principalmente se o músico já tiver cantado em excesso, se tiver se embriagado, conversado alto ou estiver, por exemplo, resfriado.

Assim, a ferramenta de transposição do **Best Vocal** tem como objetivo garantir que o músico sempre consiga cantar seu repertório sem forçar, falhar ou se preocupar em desafinar. Nos casos em que nenhuma tonalidade atenda tais requisitos, ter este conhecimento prévio é uma garantia de sucesso para o cantor, o qual poderá modificar seu repertório a tempo, sem comprometer seu nome devido a más interpretações.

Para ativar esta ferramenta, basta clicar no botão **Transposição.** Ao fazer isto, uma janela NDI é aberta para que o profissional possa transpor suas músicas na tonalidade que melhor se encaixe em seu range. O esquema de como a transposição é realizada é mostrado logo a seguir, sendo que a explicação mais detalhada é apresentada após o mesmo.

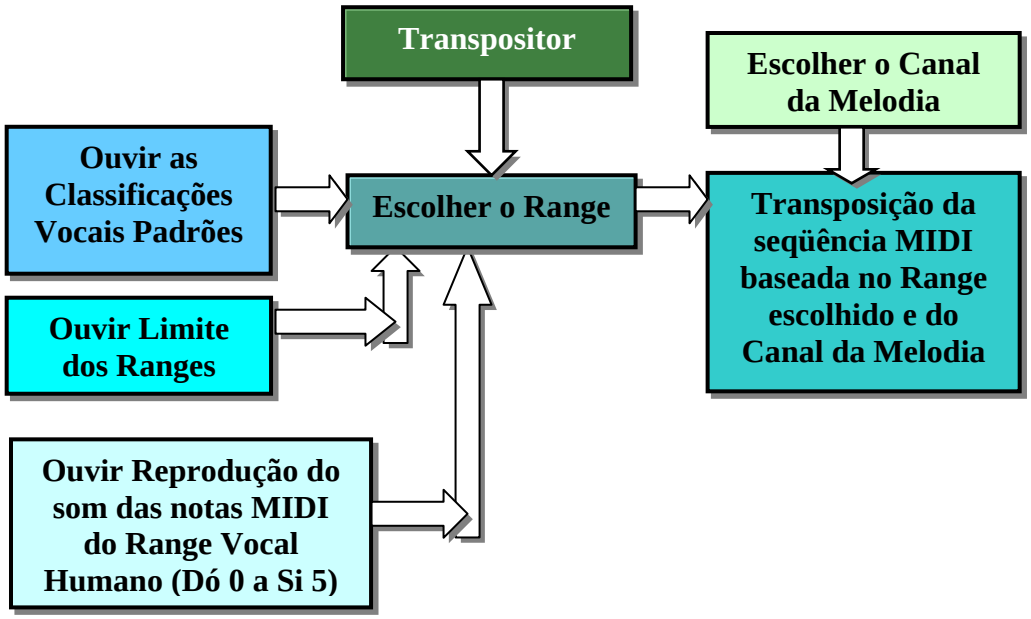


Diagrama 4.7 *– Transposição*

A janela **Transpositor** realiza duas ações distintas, mas destinadas a um mesmo objetivo, ou seja: a Transposição automática de todos os canais de uma seqüência MIDI baseada em um range vocal determinado pelo usuário e de uma melodia existente em um dos canais MIDI. Pode-se dividir a função desta janela, portanto, em duas partes:

- **Avaliação pessoal do range vocal:** Na metade inferior da janela **Transpositor**, mostrada na figura 4.11, tem-se alguns botões destinados para que o cantor

possa identificar um range vocal que julga ser o de melhor conforto e melhor qualidade timbral para ele. Para fazer isto, o sistema disponibiliza algumas classificações vocais (*vocal type*) consagradas na literatura, as quais podem ser ouvidas pelo usuário ao clicar nos respectivos botões.

- **Transposição da seqüência MIDI conforme range vocal declarado pelo usuário:**

Na metade superior da janela **Transpositor**, figura 4.11, a seguir, o profissional, após identificar um ou mais ranges de classificação vocal que julga alcançar com qualidade, pode fazer um ajuste fino, mais preciso, deste range. Este ajuste é feito através de dois pop-ups contendo notas musicais de **Dó 0** a **Si 5**. Nestes pop-ups, quando o usuário escolhe uma nota musical qualquer, o sistema emite o som da mesma, permitindo ao cantor tentar reproduzí-la com qualidade. Uma vez definido o range vocal definitivo (notas limites), pode-se efetuar a transposição da seqüência MIDI, de forma a centralizar o range da melodia que o cantor deseja cantar com o range de conforto escolhido pelo usuário. Para tanto, a janela disponibiliza um outro menu pop-up contendo a numeração dos 16 canais MIDI existentes, de forma que o usuário escolhe o canal onde está a melodia em questão, identificada na janela de **Tessitura da Música** (acionada pelo botão **Tessitura**). Ao clicar no botão **Transpor** desta janela, o sistema transpõe todos os canais MIDI (menos a bateria, canal 10) para o range de conforto do cantor.

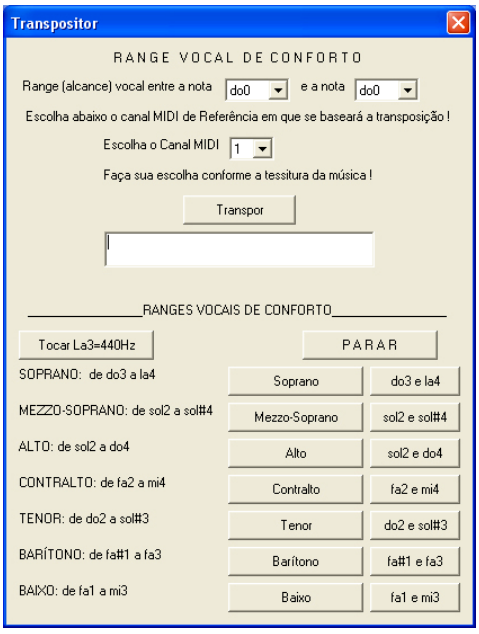


Figura 4.11 *- Janela de transposição*

A seguir, mostra-se com mais detalhes como utilizar esta ferramenta em conjunto com a ferramenta de tessitura já abordada anteriormente.

Na ferramenta de tessitura o cantor verifica o canal e instrumento da melodia que deseja cantar.

Ao mesmo tempo, o cantor seleciona no pop-up de range vocal a nota mais grave e a nota mais aguda que julga cantar sem esforço e com qualidade.

Feito isto, no pop-up **Canal MIDI** o profissional escolhe o canal da melodia identificada.

Pronto, agora é só clicar no botão **Transpor** e aguardar.

Se a transposição for possível, ou seja, se o range da melodia for menor que o do cantor, o sistema realizará a transposição da seqüência e emitirá uma mensagem de sucesso, caso contrário, emitirá uma mensagem de erro.

Figura 4.12 *- Seqüência de ações para a transposição*

Caso o cantor não conheça sua tessitura, o mesmo pode utilizar as dicas de classificação para determinar os limites de seu range, de seu alcance vocal de conforto. Para tanto existem três possibilidades de dica:

- Ouvir os ranges vocais mais usuais (em uma escala sonora)

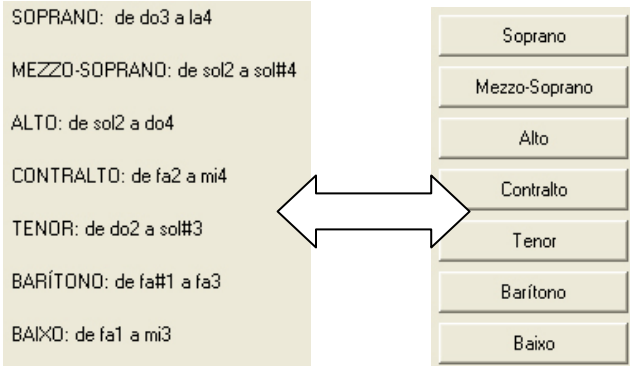


Figura 4.13 *- Classificações padrões*

- Ouvir apenas o som das notas musicais limites de cada classificação vocal

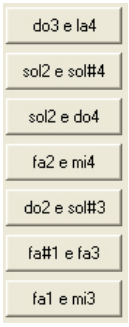


Figura 4.14 *– Botões de limites de range das classificações*

- Ouvir o som individual de cada nota musical entre a nota **Dó0** (32,7 Hz) e a nota **Si5** (1975,5Hz), observando que para o Best Vocal, a nota Lá diapasão é o Lá 3 (440Hz) (adotado por várias escolas brasileiras)

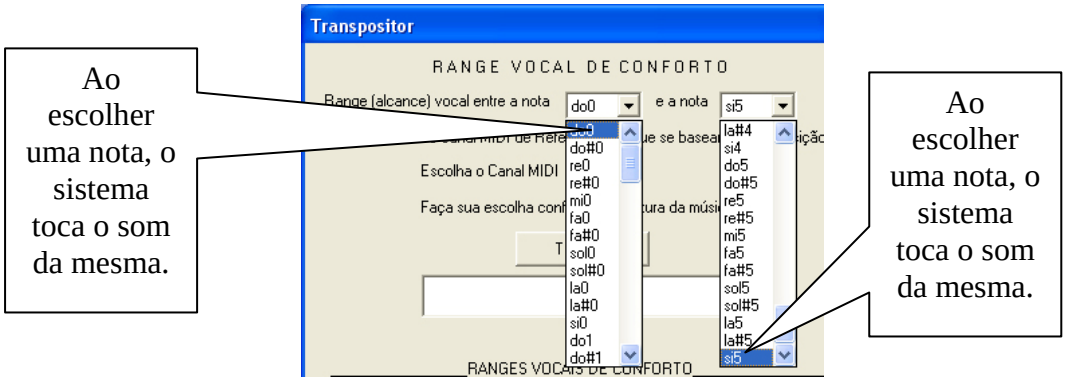


Figura 4.15 *– Pop-ups com limites do range do cantor*

Se o range do cantor não for adequado para a melodia escolhida, o **Best Vocal** emite uma mensagem avisando.

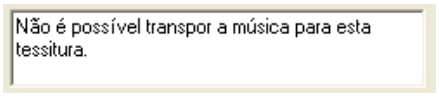


Figura 4.16 *– Mensagem de inviabilidade de transposição*

Realizadas as configurações, basta ao usuário clicar no botão de transposição da mesma janela para concluir a tarefa.

Figura 4.17 *– Botão de Transposição*

8-Salvar o arquivo mantendo as configurações do arquivo original acrescido das modificações realizadas

Esta ferramenta é realmente interessante. A princípio simples, mas, vista em profundidade, complexa de ser implementada. Realizadas as modificações desejadas pelas ferramentas anteriormente citadas, o **Best Vocal** salva o arquivo mantendo todas as características do arquivo inicial, tais como: título, copyright, lirismo, metrônomo e todos os eventos e mensagens MIDI exatamente iguais às existentes no arquivo original. Para utilizá-la, basta clicar no botão **Salvar**. Após realizado o salvamento, se o profissional clicar no botão **Tocar**, o mesmo executará a música com as alterações procedidas.

***Figura 4.18** - Salvando um arquivo modificado*

Observações finais

Pontos relevantes

Fazendo uma análise macroscópica, os pontos fortes e atrativos deste sistema estão principalmente:

- na ferramenta de eliminação de pausas iniciais dos arquivos MIDI que tanto incomodam os músicos, principalmente os que utilizam seqüências MIDI como playback;
- na ferramenta de transposição baseada na tessitura vocal do músico, permitindo ao mesmo sempre cantar na região de conforto vocal, evitando desafinar;
- por salvar o arquivo modificado no mesmo formato de abertura, mantendo as demais características, eventos, meta-eventos e mensagens existentes no arquivo original.

Capítulo 5

Estudo de casos empregando o Best Vocal 2005 e análise comparativa com outros programas semelhantes

Como primeiro estudo de caso, optou-se por mostrar passo a passo, detalhadamente, como proceder para transpor uma seqüência MIDI adequando-a para performance de um quarteto vocal.
Neste exemplo, além de tornar clara a utilização e potencialidades do Best Vocal 2005, será mostrado sua relevância em tarefas que requerem do músico um bom conhecimento e experiência em teoria musical, além de diminuir sensivelmente o tempo gasto na realização de tarefas afins.

5.1 Determinando o alcance vocal de conforto e adequando seqüências MIDI para acompanhar um grupo vocal

Aplicações:

- Se um cantor for participar de programas de seleção de cantores, tais como o programa da Sony “American Idol”, do “Fama” da Globo, do “Pop Star” do SBT, do “ Programa Raul Gil” ou de outro concurso qualquer de cantores, populares ou eruditos, onde uma das etapas é cantar solo a capela (sem

instrumentos), o maior problema é começar a cantar em uma tonalidade que lhe permita alcançar todas as notas da música sem desafinar (mais grave ainda é não conseguir cantar algumas das notas).

- No caso de um regente ou arranjador de um grupo vocal, um dos grandes problemas a enfrentar é determinar quem vai cantar qual voz em uma determinada música.
- Outro problema enfrentado por maestros e arranjadores é determinar a melhor tonalidade de uma música que permita a todos os cantores cantar agradavelmente a voz a eles destinada.
- O problema se agrava quando se tem poucas opções vocais, um pequeno número de cantores e uma porcentagem significativa de amadores, tanto cantores, quanto regentes e arranjadores.

Assim, determinar a tessitura vocal (o alcance vocal de conforto), para cada cantor e adaptar a música a ela é um trabalho que exige um profissional experiente e com uma relatividade dose de *feeling*.

Neste capítulo será mostrado como adequar uma seqüência MIDI para tais tipos de usuários (canto a capela e grupos vocais), culminando na geração de uma seqüência MIDI compatível com todos os cantores, estando os mesmos afinados e sem esforçar suas pregas vocais.

5.1.2 Identificando e classificando o range vocal de conforto para um cantor qualquer

O Best Vocal 2005 permite a qualquer tipo de músico, mesmo os sem experiência em classificação vocal e que tenham pouco conhecimento de teoria musical, manuseá-lo, conforme visto no capítulo 4 e repetido alguns pontos ainda neste capitulo.
Na realidade, para o músico utilizar o Best Vocal 2005 ele deverá saber pelo menos os nomes das notas musicais, ou, se nem isto souber, bastará ao mesmo ouvir o som de cada nota musical e identificar qual a nota mais grave e a mais aguda que consegue emitir; tarefa, até então, de pequena ou nenhuma complexidade.

5.1.2.1 Passos para utilização do sistema

1. Foi criado um instalador para o programa produzido, o qual está no CDROM em anexo. Assim, para utilizar o sistema, deve-se instalar o programa (Best Vocal 2005) no computador, bastando clicar no ícone do aplicativo Instalar.exe, conforme mostrado na figura 5.1.

Figura 5.1 *- Instalador do Best Vocal 2005*

2 A instalação é simples, bastando concordar com tudo o que for solicitado. Ao fazer isto, um grupo de programas é criado no computador. Para utilizar o sistema deve-se clicar no menu **Iniciar** do Windows, escolher o grupo **Best Vocal 2005** e clicar na opção **BestVocal2005.exe**, conforme figura 5.2.

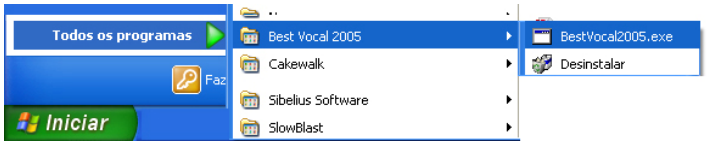


Figura 5.2 *- Ativando o Best Vocal 2005*

3 Ao fazer isto, a interface do Best Vocal 2005 é aberta, conforme figura 5.3.

Figura 5.3 *- Interface do Best Vocal 2005*

4 Para utilizar o Best Vocal, deve-se, conforme afirmado no capitulo 5, abrir um arquivo MIDI qualquer para ativar a interface. Se o problema for apenas classificar uma voz, qualquer arquivo MIDI serve. Para abrir o arquivo MIDI,

basta clicar no botão 

. Ao fazer isto, a janela explorar do Windows é aberta. Escolha o diretório e o arquivo que deseja abrir. No exemplo da figura 5.4, escolheu-se a música **PL088101.mid** da revista Playmusic 81, cedida pela mesma e colocada no CDROM em anexo.

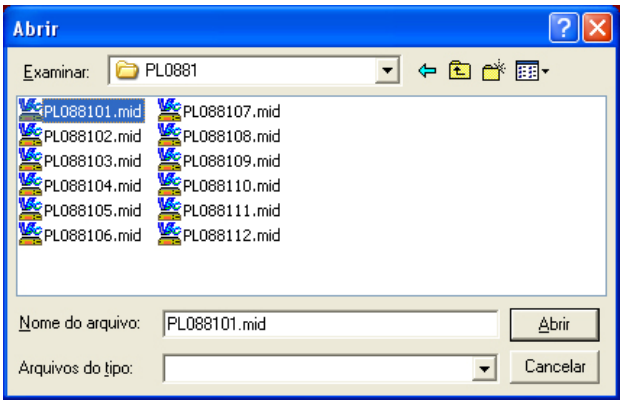


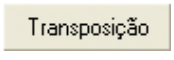
Figura 5.4 *- Abrindo um arquivo MIDI*

5 Se o arquivo realmente existir e for uma seqüência MIDI correta, a interface mostra uma mensagem informando que a abertura foi concluída com sucesso (Figura 5.5) ou uma mensagem que o arquivo não é um arquivo MIDI válido (Figura 5.6).

Figura 5.5 *- Mensagem de arquivo aberto com sucesso*

Figura 5.6 *- Mensagem de erro ao abrir um arquivo*

6 De posse do arquivo aberto, a interface está ativada e todas as ferramentas se tornam operacionais.

7 Deve-se clicar no botão 

para abrir a janela com as ferramentas que permitem a determinação do range vocal de cada cantor.

8 Ao fazer isto, a janela **Transpositor** é aberta apresentando a interface da mesma, conforme mostrado na figura 5.7.

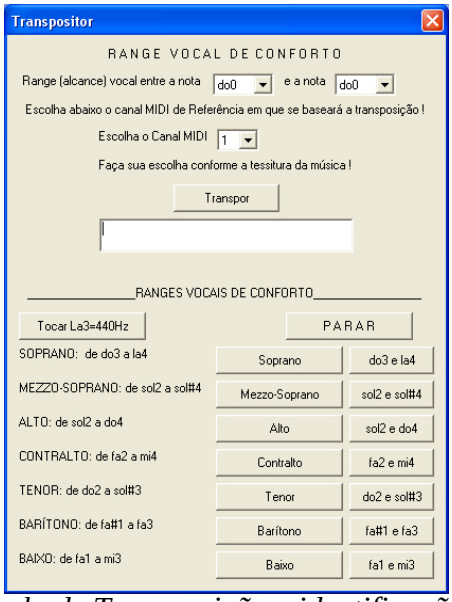


Figura 5.7 *- Janela de Transposição e identificação de range vocal*

9 Para que o cantor possa determinar o seu range (alcance vocal de conforto), existem três possibilidades:

9.1 Pode-se utilizar os botões que disparam (tocam) todas as notas dos ranges das classificações vocais mais usuais. Ao utilizar esta opção o músico poderá cantar junto com o som produzido pelo sistema e ver se realmente consegue reproduzir todas as notas com qualidade. Como geralmente a autocrítica não é um forte do ser humano, o ideal é que se peça a alguém para escutar junto para se ter uma segunda opinião. Esta ferramenta é bastante útil como assistência a regentes, fonoaudiólogos e profissionais do canto que normalmente fazem este tipo de análise em suas profissões.

As classificações vocais e seus ranges, disponibilizadas no sistema, estão registradas na figura 5.8 a seguir:

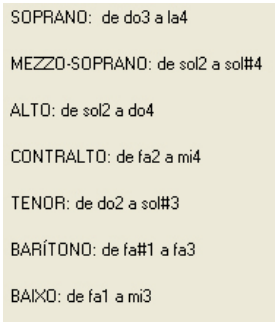


Figura 5.8 *- Classificação vocal padrão e seus ranges*

Para fazer soar todas as notas de cada uma das classificações, basta clicar nos botões correspondentes, mostrados na figura 5.9.

Figura 5.9 *- Botões de disparo das notas dos ranges*

9.2 Pode-se utilizar os botões que apenas tocam as notas limites (mais grave e mais aguda) das classificações. Esta opção é ideal para quem já reconhecidamente possui uma boa voz, bastando ouvir os limites dos ranges para se classificar, já que tem certeza que todas as demais notas soarão bem dentro dos limites identificados.

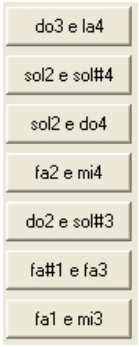


Figura 5.10 *- Botões de limites de range*

9.3 Pode-se utilizar os menus pop-up localizados no alto da janela **Transpositor**, o qual contém todas as notas musicais[1] entre a nota **Dó0** (32,7 Hz) e a nota **Si5** (1975,5Hz). Ao escolher clicar em uma nota qualquer, o Best Vocal produz o som da mesma para a apreciação do cantor. Utilizar os pop-ups para determinar os limites do range vocal é bastante útil para cantores que possuam um alcance vocal maior ou menor que o das classificações, ou seja, possuem um range vocal não padrão. A figura 5.11 mostra a utilização dos pop-ups.

[1] Cabe lembrar que, para a notação Brasileira e adotada pelo Best Vocal, o **Lá3** (diapasão) possui a frequência de 440Hz.

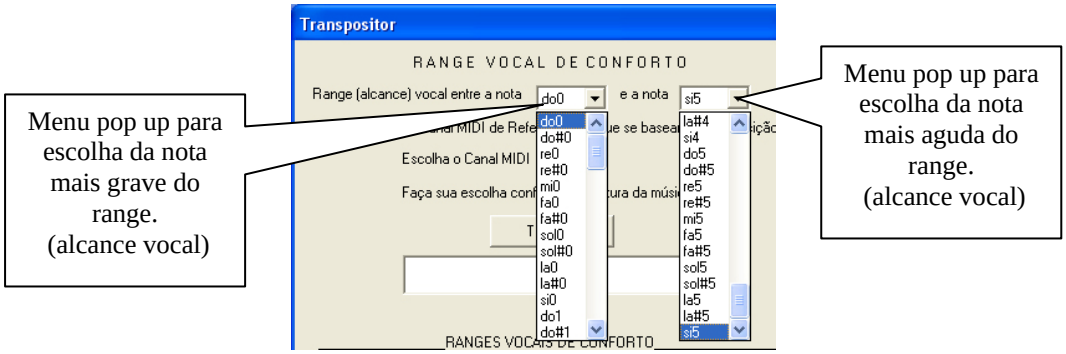


Figura 5.11 *- Tocando tons através dos menus pop-ups*

9.4 Assim, o usuário pode utilizar apenas uma das opções ou todas em conjunto. Realizados testes de campo, estatisticamente a preferênca de utilização se deu nos três caminhos, executados na seguinte seqüência:

- Deve-se escutar inicialmente todas as notas de cada classificação e seus limites, observando se é possível cantar a nota mais grave e mais aguda de cada classificação. Caso desejar interromper a execução do range completo, basta clicar no botão 

ou clicar em outro botão de classificação.
- Caso não consiga cantar todas as notas de uma das classificações, anote os nomes das notas musicais limites da classificação na qual você atingiu mais notas e vá para o menu pop-up para identificar as notas limites de seu alcance vocal de conforto (Figura 5.12).

Figura 5.12 *– Limites do range vocal de conforto*

- Partindo da nota limite inferior do range escolhido, clique no menu pop-up da esquerda, o qual permite a escolha da nota mais grave (Figura 5.13), e identifique o limite inferior de seu range vocal.

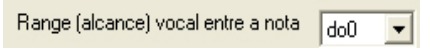


Figura 5.13 *– Limite inferior do range*

- Partindo da nota limite superior do range escolhido, clique no menu pop-up da direita, o qual permite a escolha da nota mais aguda (Figura 5.14), e identifique o limite superior de seu range vocal.

Figura 5.14 *- Limite superior do range*

- Anote estes limites. Futuramente precisará utilizá-los para transpor a música para um arranjo vocal completo para vários cantores.
- Caso a extensão vocal do cantor seja superior à das classificações, escolha a que lhe for mais confortável, anote os nomes das notas musicais limites desta classificação e vá para o menu pop-up identificar as notas limites de seu alcance vocal de conforto.
- Partindo da nota limite inferior do range escolhido, deve-se clicar no menu pop-up da esquerda, o qual permite a escolha da nota mais grave. Identifique o limite inferior do range vocal, escolhendo notas mais graves até que não consiga produzir a nota escolhida ou que a mesma saia "suja" ou desafinada. Neste caso, anote a nota anterior a esta última, a qual identificará o limite inferior real do range[2].
- Partindo da nota limite superior do range escolhido, clique no menu pop-up da direita, o qual permite a escolha da nota mais aguda (Figura 16). Identifique o limite superior do range vocal, escolhendo notas mais agudas até que não consiga produzir a nota escolhida ou que a mesma saia "suja" ou desafinada. Neste caso, anote a nota anterior, a qual identificará o limite superior real do alcance vocal de conforto (Tessitura).

[2] Existem casos em que cantores alcançam mais de um range de classificação. Isto é muito comum de acontecer.

5.2 Identificação do range vocal do cantor, manipulação de arquivos MIDI e transposição musical baseada no range do cantor

Este item é responsável por:

- apresentar as ferramentas desenvolvidas para auxiliar o cantor a identificar seu range vocal de conforto,
- mostrar como extrair do arquivo MIDI o range de cada instrumento, cada canal MIDI,
- mostrar como determinar em qual canal da seqüência MIDI está a melodia que se deseja cantar,
- mostrar como eliminar as pausas iniciais que possivelmente existam na seqüência MIDI,
- mostrar como transpor a música (seqüência MIDI) a partir do conhecimento do canal MIDI da melodia e do range vocal do cantor.

Assim, para transpor a seqüência MIDI (uma música) de tal forma que todos os cantores sejam agraciados com uma melodia adequada a seu range vocal de conforto, deve-se seguir as seguintes instruções:

5.2.1 Abrindo um novo arquivo MIDI

Escolha um arquivo **MIDI** contendo a música desejada. No exemplo dado, utilizar-se-á a seqüência MIDI SABIA.mid (disponibilizada no CDROM em anexo), a qual contém um arranjo coral a 4 vozes.

1- Abra o arquivo exemplo contendo a música **SABIÁ.** Para tanto, clique no botão Abrir. Feito isto, a janela explorar do Windows é aberta permitindo que o usuário escolha o diretório e o arquivo que deseja abrir (SABIA.mid).

Figura 5.15 *– Abrindo o arranjo vocal a 4 vozes*

2- Se o arquivo realmente existir e for uma seqüência MIDI correta, a interface mostra uma mensagem informando que o arquivo MIDI foi aberto com sucesso (Figura 5.16) ou uma mensagem que o arquivo não é um arquivo MIDI (Figura 5.17).

Figura 5.16 *– Mensagem de arquivo aberto com sucesso*

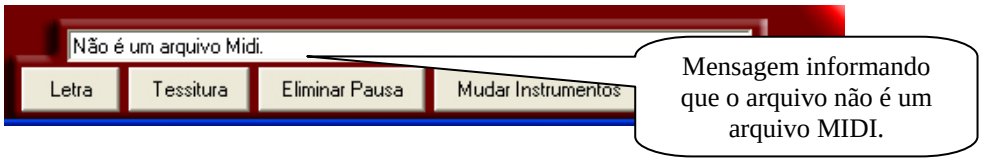


Figura 5.17 *- Mensagem de erro na abertura de um arquivo*

5.3 Verificando a tessitura dos instrumentos do arranjo midi

1. Clique no botão 

e a janela *Tessitura da Musica* será aberta com todos os canais de seu coral e o instrumento que estiver em cada canal.
2. Observe o item TESSITURA DE CADA CANAL MIDI. Nele estarão especificadas as notas mais graves e agudas que o instrumento alcança.
3. Observe o item INSTRUMENTOS DE CADA CANAL. No mesmo são apresentados os instrumentos de cada canal. Isto facilita a identificação do instrumento MIDI que está tocando uma voz específica. No exemplo da figura 5.18, todos os canais estão tocando o instrumento ChoirAahs (coral de Aahs).

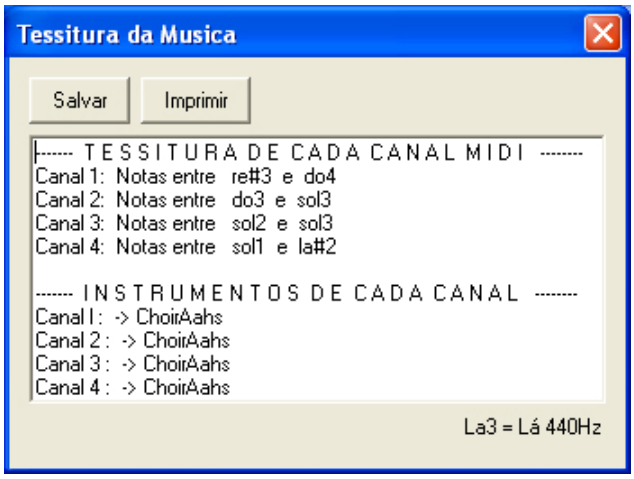


Figura 5.18 *- Janela de range e nome dos instrumentos*

4. As figuras 5.19 e 5.20 mostram um detalhe desta janela com informações sobre o Canal MIDI 1, de onde se conclui que a tessitura do Canal 1 está entre as notas musicais **re#3** e o **do4**. Não se pode esquecer que para o Best Vocal 2005, em português, o dó central do piano é o **Dó3**, e o **Lá 440Hz** é o **Lá3** (figura 5.21).

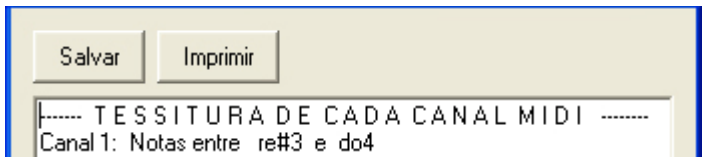


Figura 5.19 *– Tessitura do canal 1*

Figura 5.20 *- Instrumento do canal 1*

Figura 5.21 *– Dó central do piano e o Lá 440Hz*

5.4 Verificando se o alcance vocal dos cantores é adequado a uma determinada seqüência MIDI

1- Ao abrir a janela Tessitura da Música (Figura 5.22) obtém-se o conhecimento do range de notas abrangido por cada Canal MIDI. Na mesma observa-se que a música possui um arranjo vocal para quatro vozes. A tessitura de cada voz está especificada naquela janela. No item 5.1 foi mostrado com detalhes como determinar o alcance vocal de um cantor utilizando a janela **Transpositor** do Best Vocal 2005, o qual era acionado ao se clicar no botão Transposição .

2- Suponha que se tenha quatro cantores para cantar o arranjo, e que cada um já realizou a identificação de seu alcance vocal de conforto, conforme a seguir:

- **Cantor 1**: alcança da nota **sol2** até a **lá4**.
- **Cantor 2**: alcança da nota **fá3** até a **mi4**
- **Cantor 3**: alcança da nota **do2** até a **mi4**.
- **Cantor 4**: alcança da nota **fá1** até a **sol3**.

3- Para facilitar enxergar os ranges dos cantores e confrontá-los com os ranges dos instrumentos do arranjo da música Sabiá, a figura 5.22 mostra a superposição dos mesmos visualizando em um teclado de piano.

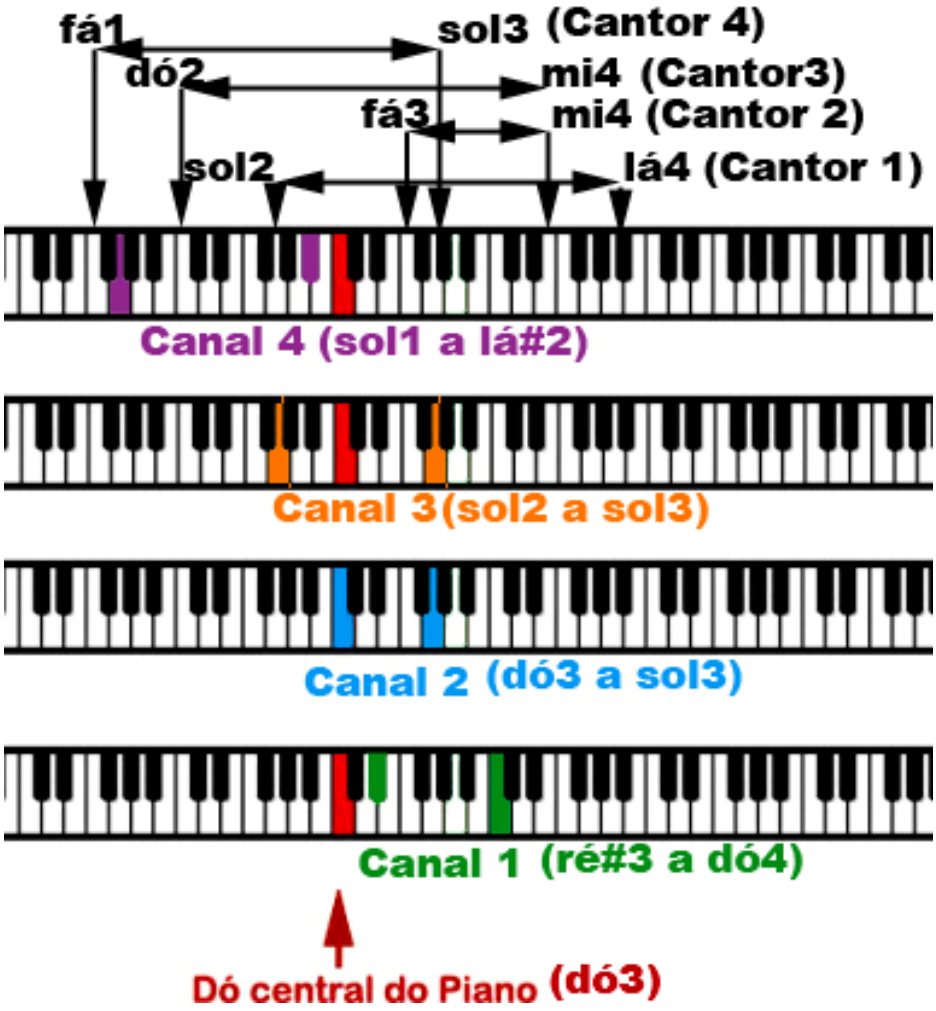


Figura 5.22 *– Visualização gráfica dos ranges dos canais do arranjo e dos cantores*

- **Cantor 1**

Através da figura 5.23, percebe-se que o Cantor 1 possui um alcance vocal (range) que abrange as vozes do Canal 1, do Canal 2 e do Canal 3, estando o mesmo apto a cantar qualquer uma desta vozes.

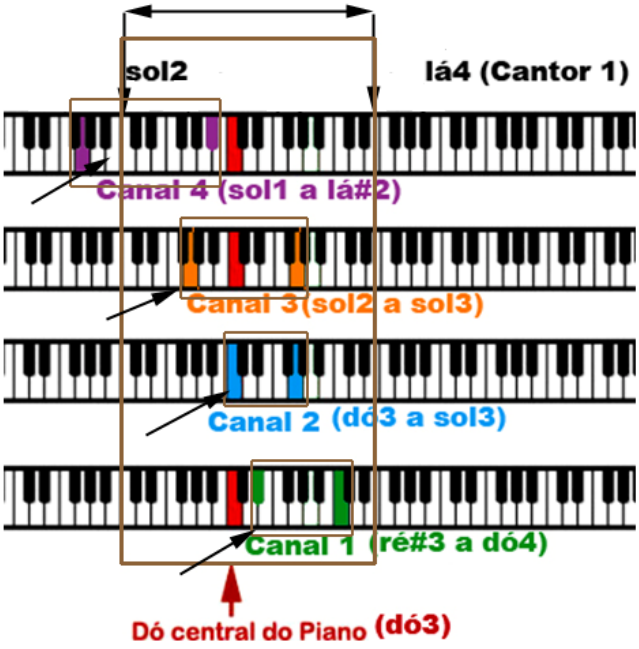


Figura 5.23 *- Range do cantor 1 x Tessitura da música*

- **Cantor 2**

Através da figura 5.24, percebe-se que o Cantor 2 não possui um alcance vocal (range) que abrange alguma das vozes do arranjo, e, desta forma, não está apto a cantar qualquer uma desta vozes na tonalidade atual da música.

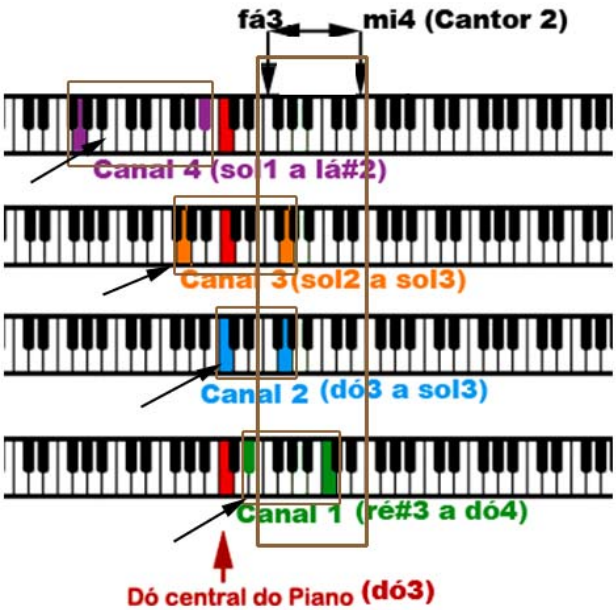


Figura 5.24 *- Range do cantor 2 x Tessitura da música*

- **Cantor 3**

Através da figura 5.25, percebe-se que o Cantor 3 também possui um alcance vocal (range) que abrange as vozes do Canal 1, do Canal 2 e do Canal 3, estando o mesmo apto a cantar qualquer uma desta vozes.

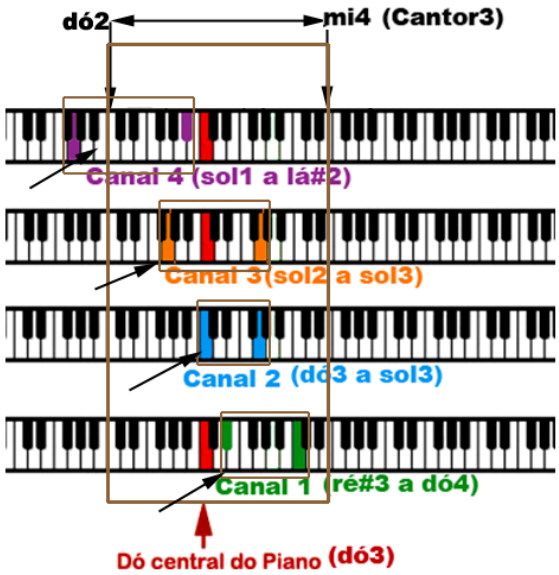


Figura 5.25 *- Range do cantor 3 x Tessitura da música*

- **Cantor 4**

Através da figura 5.26, percebe-se que o Cantor 4 possui um alcance vocal (range) que abrange as vozes do Canal 2, do Canal 3 e Canal 4, estando o mesmo apto a cantar qualquer uma desta vozes.

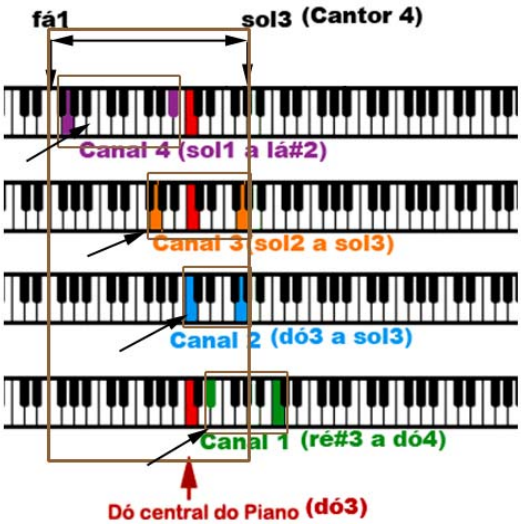


Figura 5.26 *- Range do cantor 4 x Tessitura da música*

Apenas o Cantor 2 não possui alcance vocal para cantar qualquer uma das vozes. A opção seria trocar este cantor ou transpor a música para que ele consiga cantar pelo menos uma das vozes. A primeira opção que deve ser testada é transpor a música para enquadrar todos os cantores ao arranjo, sem que seja necessário procurar um novo cantor com range suficiente para fazer parte deste grupo vocal.

5.5 Adequando o arranjo da seqüência MIDI para que todos os cantores, inclusive o Cantor 2, do presente quarteto vocal possam cantar uma determinada voz da música Sabiá

De posse do Best Vocal, esta tarefa já não pode ser considerada complexa ou trabalhosa. Basta, para tanto, seguir os passos mostrados a seguir:

1. No exemplo dado, o **Cantor 2** não consegue cantar nenhuma voz (melodia). Assim, todo o trabalho da transposição será ajustar o arranjo para que este cantor seja capaz de cantar pelo menos uma das melodias.
2. Para tanto, deve-se escolher uma das vozes da música que possua uma extensão vocal (número de notas musicais do range) igual ou menor que a do Cantor 2, já que esta voz deverá ficar enquadrada e centralizada no range vocal deste cantor.

3. Analisando o arranjo, verifica-se que o Canal 2 atende este requisito. O range do Canal 2 possui 9 (nove) notas, enquanto o range do Cantor 2 possui um range com 12 (doze) notas.
4. Assim, a solução agora é transpor a música conforme o alcance vocal deste cantor (Cantor 2) e a voz que ele deseja cantar (Canal 2). O **Best Vocal,** conforme já descrito neste capítulo e no capítulo 5, faz isto automaticamente, mediante a inserção correta do range do cantor e do canal MIDI onde reside a melodia que se deseja cantar.
5. Para tanto, nos menus pop-up de RANGE VOCAL DE CONFORTO (na janela **Transpositor**), coloque o range vocal do Cantor 2, conforme mostrado na figura 5.27.
6. Logo após, selecione o Canal MIDI 2 que será transposto, juntamente com todos os demais canais, para adequar o arranjo ao Cantor 2 (figura 5.27).
7. Feito isto, clique no botão **Transpor**, conforme figura 5.27.

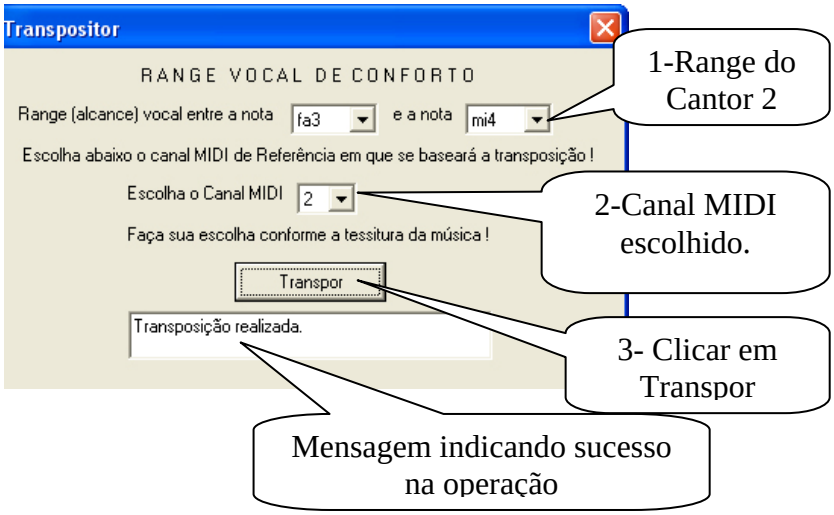


Figura 5.27 *- Transpondo a música automaticamente*

8. A próxima etapa é verificar a nova tessitura dos Canais do Arranjo, agora adaptado ao Cantor 2 para que possa cantar a melodia do Canal 2, para ver se os outros cantores ainda conseguem cantar pelo menos uma das vozes cada um.

9. Para tanto, basta clicar novamente no botão 

.

10. Ao fazer isto, uma nova janela de tessitura é aberta. Mantendo a anterior também ativa, você poderá comparar a música original com a transposta.
11. A vantagem de deixar a janela de tessitura anterior e posterior é facilitar verificar o que foi modificado. As duas janelas ficam sobrepostas. Assim, para ver as duas, desloque uma delas para sair uma de cima da outra. A figura 5.28 mostra estas duas janelas lado a lado. Logo abaixo desta figura é repetido o alcance dos cantores com a finalidade de verificar se a nova seqüência musical atende as solicitações, ou seja, se existe um cantor para executar cada uma das 4 (quatro) vozes..

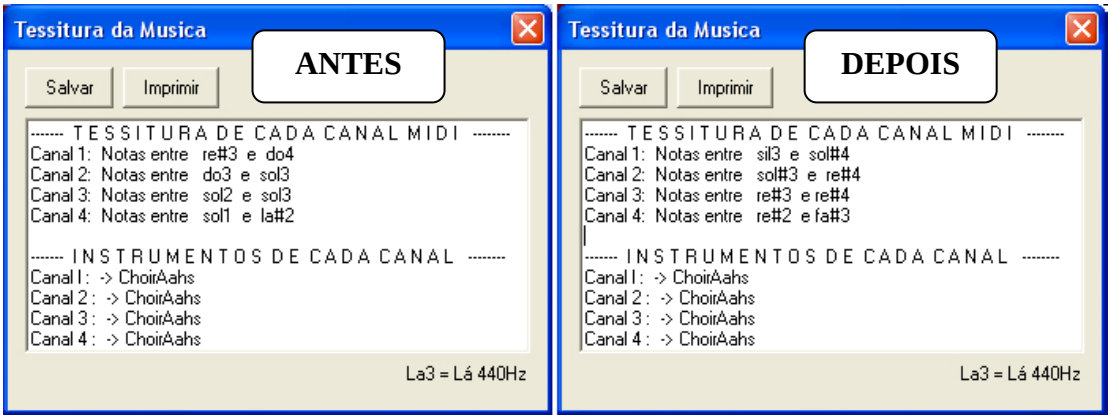


***Figura 5.28** – Tessitura antes e depois da transposição automática*

Range vocal do quarteto vocal.

- **Cantor 1**: alcança da nota **sol2** até a **lá4**.
- **Cantor 2**: alcança da nota **fá3** até a **mi4**.
- **Cantor 3**: alcança da nota **do2** até a **mi4**.
- **Cantor 4**: alcança da nota **fá1** até a **sol3**.

5.6 Nova análise dos Cantores e Canais

- **Cantor 1:** Com seu range (de **sol2** a **lá4)**, o mesmo consegue cantar o range das melodias (vozes) do Canal 1, do Canal 2 e do Canal 3.

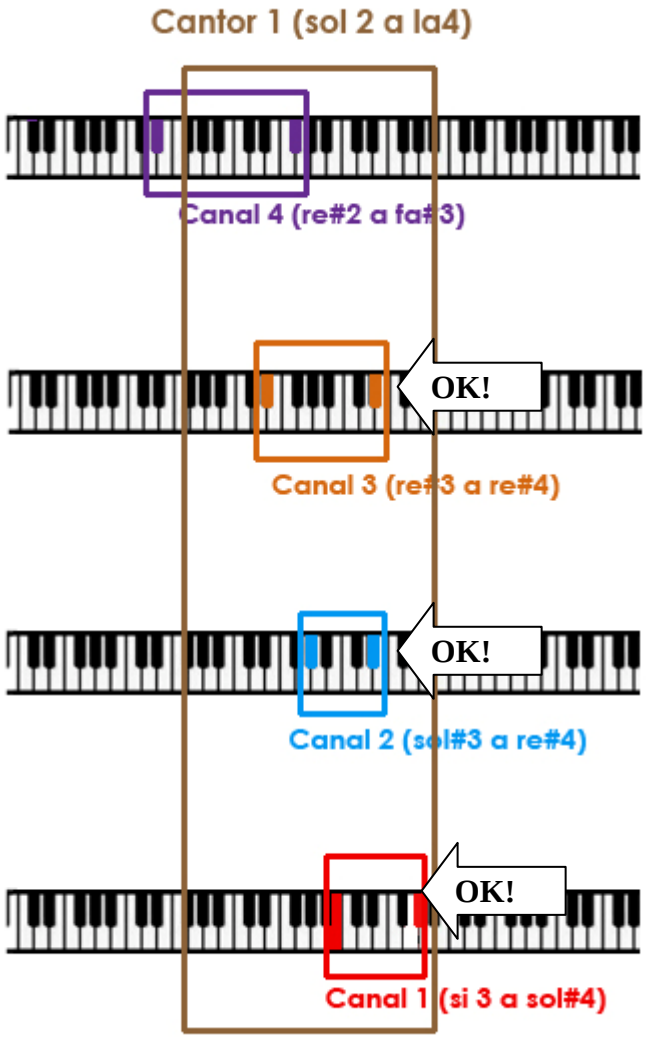


Figura 5.29 *– Comparação do range do cantor 1 com as melodias transpostas*

- **Cantor 2:** Com seu range (de **fá3** a **mi4)**, o mesmo consegue cantar apenas o range da melodia (voz) do Canal 2, conforme projetado.

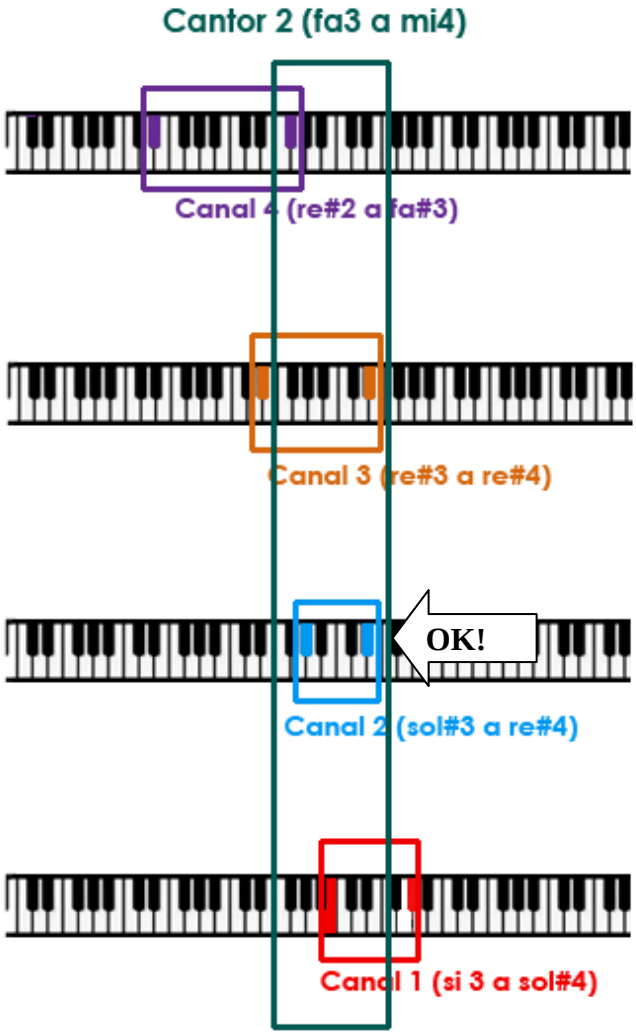


Figura 5.30 *– Comparação do range do cantor 2 com as melodias transpostas*

- **Cantor 3:** Com seu range (de **do2** a **mi4)**, o mesmo consegue cantar os ranges das melodias (vozes) do Canal 2, do Canal 3 e do Canal 4.

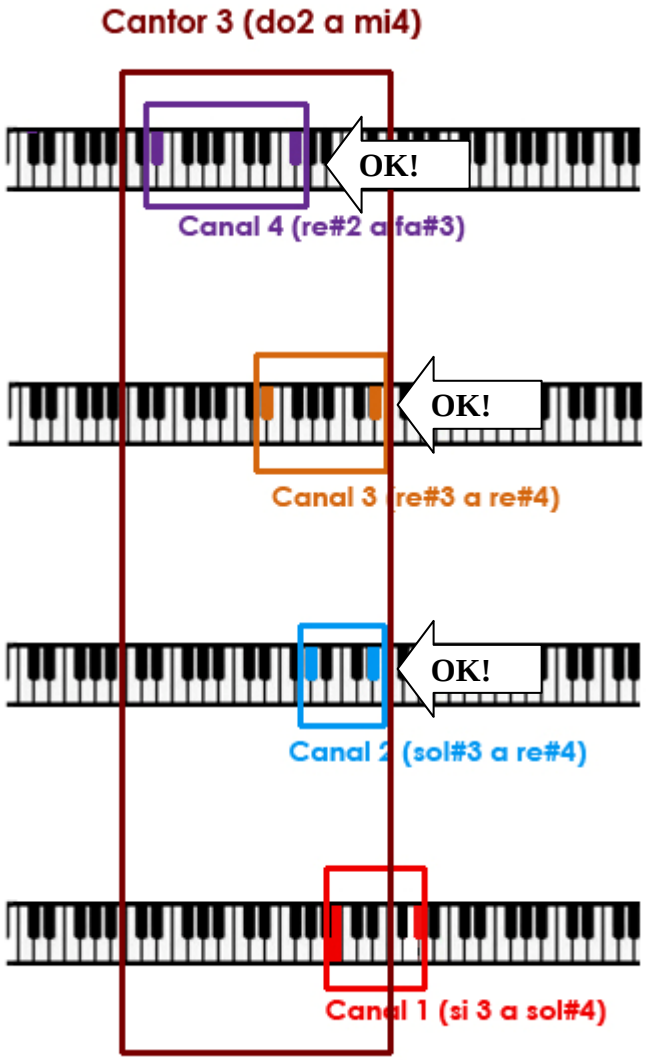


Figura 5.31 *– Comparação do range do cantor 3 com as melodias transpostas*

- **Cantor 4:** Com seu range (de **fá1** a **sol3)**, o mesmo consegue cantar o range da melodia (voz) do Canal 4.

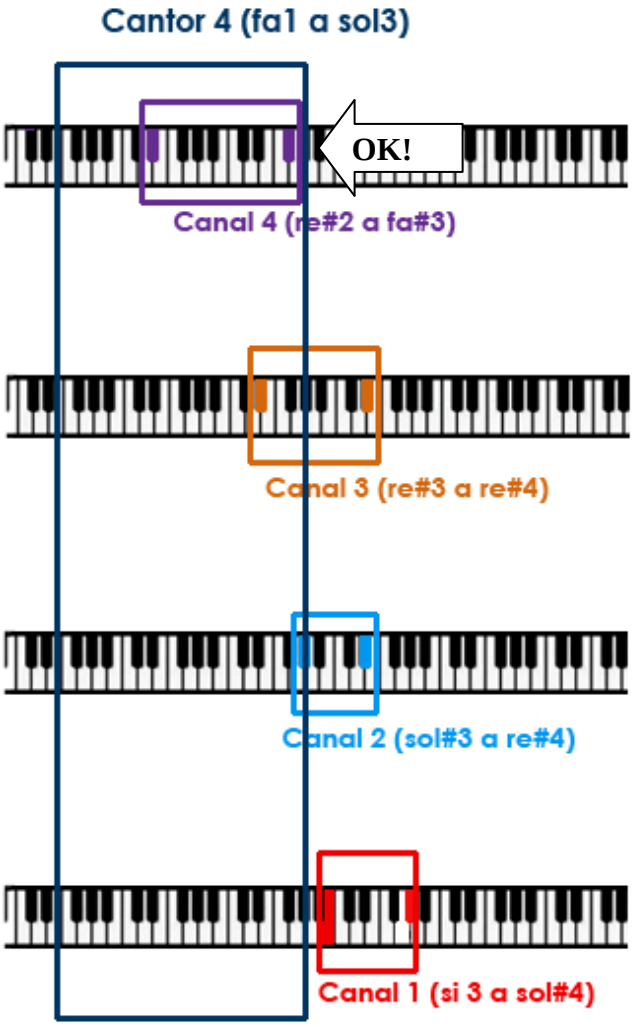


Figura 5.32 *– Comparação do range do cantor 4 com as melodias transpostas*

Deve-se observar que a transposição automática realizada pelo Best Vocal 2005 apresentou uma solução para o problema, onde:

- Para a voz do Canal 1 tem-se a opção do Cantor 1;
- Para a voz do Canal 2 tem-se a opção do Cantor 1, do Cantor 2 ou do Cantor 3;
- Para a voz do Canal 3 tem-se a opção do Cantor 1 ou do Cantor 3;
- Para a voz do Canal 4 tem-se a opção do Cantor 3 ou do Cantor 4.

A escolha dos cantores para o novo arranjo da música Sabiá é a seguinte:

- **Voz do Canal 1** – **Cantor 1** (única opção deste cantor).
- **Voz do Canal 2** – **Cantor 2** (única voz que o mesmo consegue cantar).
- **Voz do Canal 3** – **Cantor 3** (já que a voz 1 foi destinada ao cantor 1)
- **Voz do Canal 4** – **Cantor 4** (único que consegue cantar a voz 4).

Observações Relevantes:

O resultado final obtido, da mesma forma que o estudo de caso real realizado e que será apresentado ainda neste capítulo, item 5.10**,** dependerá da qualidade vocal dos cantores.

Se os mesmos tiverem boa qualidade no timbre em todo os ranges vocais identificados, personalizadamente, o resultado será bom, já que se garante com tal transposição que o cantor irá conseguir cantar todas as notas da melodia destinada a ele, bem como não desafinará.

Por outro lado, nem sempre o resultado final do conjunto será o melhor resultado possível, já que o timbre de alguns cantores são mais agradáveis na região mais aguda, outros, na região mais grave.

Partindo desta afirmativa, se existir para todos os cantores mais de uma opção de transposição, dever-se-á adotar aquela que permita ao cantor cantar em uma região onde seu timbre é mais agradável. Observando o caso do cantor 2, o mesmo possui um range de Fá3 a Mi4, e, o canal que vai executar possui um range de Sol#3 a Ré4. Assim, pode-se perceber que este cantor terá a opção de escolher entre cinco transposições, tonalidades, possíveis, ou seja:

10 De Lá3 a Mi 4

11 De Sol#3 a Ré# 4 (o sugerido pelo sistema)

12 De Sol 3 a Ré 4

13 De Fá#3 a Dó#4

14 De Fá 3 a Dó 4

Desta forma, pode ser que uma das outras quatro alternativas seja melhor, em termos de resultado sonoro, do que a centralizada no range.

Assim, deve-se prever como trabalho futuro, gerar esta informação ao cantor, permitindo que o mesmo escolha para qual tonalidade a música deverá ser transposta. Apesar de, na maioria das vezes, ser mais prático substituir o cantor que não possui uma

boa extensão vocal, com o sistema desenvolvido a tarefa de adaptar uma música para um ou mais cantores passa a ser mais agradável e efetiva.

Concluindo, com o Best Vocal pode-se adaptar um arranjo a um número determinado de cantores, com segurança, e de uma forma rápida, sem que, neste processo, seja necessário substituir um cantor nem correr o risco de destinar uma voz inadequada a um cantor que a executará com dificuldade, desafinadamente e com baixa qualidade.

5.7 Eliminando as pausas iniciais do arquivo

1- Antes de salvar as modificações realizadas, para que o usuário não tenha que esperar alguns segundos para o arquivo começar a tocar, devido à existência de pausas no início do arquivo, o mesmo pode eliminar estas pausas simplesmente clicando no botão de eliminar pausa existente na interface principal do Best Vocal 2005, conforme mostra a figura 5.33.

Figura 5.33 *– Eliminando pausas iniciais do arquivo*

2- Ao fazer isto, uma mensagem “Pausa retirada” é mostrada no campo de texto.

5.8 Salvando as modificações realizadas: transposição e eliminação de pausas

O Best Vocal salva a seqüência MIDI transposta, sem qualquer alteração adicional ao arquivo original que não seja as notas musicais dos canais, transpostas de acordo com o range vocal de um ou mais cantores. Para tanto, deve-se proceder os seguintes passos:

1- Clicar no botão 

da interface principal do Best Vocal 2005.

2- Ao fazer isto, em algumas máquinas e sistemas operacionais o botão de salvar fica em branco, e, em outros, toda a interface fica em branco. Quando isso acontece,

têm-se a impressão que algo errado ocorreu. Não é verdade**:** o que ocorreu é normal!

3- Dependendo da complexidade do arquivo MIDI, do grau de ocupação da máquina, ou seja, de quantos processos a mesma estiver rodando no momento, o salvamento gasta de 1 (um) a 7 (sete) minutos.

   Como exemplo: em um Duron de 900 MHz com 128 MBytes de RAM e taxa de ocupação menor que 30%, uma música de 30Kbytes, com 10 canais, gasta em média 3 minutos e meio para salvar.

Deste estudo de caso, pode-se perceber que o programa implementado atende os requisitos a ele solicitado.

5.9 Outros programas que fazem operações semelhantes às do Best Vocal

Existem vários programas comerciais que realizam transposição musical, tais como o Band-in-a-Box, o Finale, o Sibelius, o Sonar, e outros. O que ocorre é que estes programas fazem a transposição tradicional baseada em tonalidades ou por uma quantidade de semitons acima ou abaixo do pré-existente.

Apenas o Band-in-a-Box, partir da versão de 2005, é que começa a fazer transposições baseadas em tessitura, até mesmo por solicitação do grupo de Computação Musical da FEELT no ano de 2004, o qual já tinha escrito alguns livros e vários artigos sobre este programa com o subsídio dos programas cedidos pela PG Music Inc (www.pgmusic.com).

Após a solicitação, sem sucesso a curto prazo, resolveu-se implementar neste trabalho de dissertação um sistema que fosse simples, eficiente e aderente ao usuário do domínio musical, onde, já mesmo em 2004, apresentou-se a primeira versão do Best Vocal 2004, com publicação sobre análise do mesmo na revista Playmusic de fevereiro de 2005.

5.9.1 Band-in-a-Box 2005

O Band-in-a-Box 2005 incorporou uma ferramenta para transposição de arranjos baseado em *vocal range* (extensão vocal). Um dos problemas que esta ferramenta apresenta é que ela só aceita arranjos elaborados na sua plataforma e que tenha pelo menos uma trilha de melodia. Outra limitação é que a interface é complexa de ser utilizada por um usuário leigo em aplicativos computacionais aplicados à música, bem como demanda do usuário de conhecimentos mais profundos de teoria musical. A figura 5.34 mostra a interface inicial desta ferramenta.

Nas figuras 5.34 e 5.35 são mostradas as interfaces desta ferramenta.

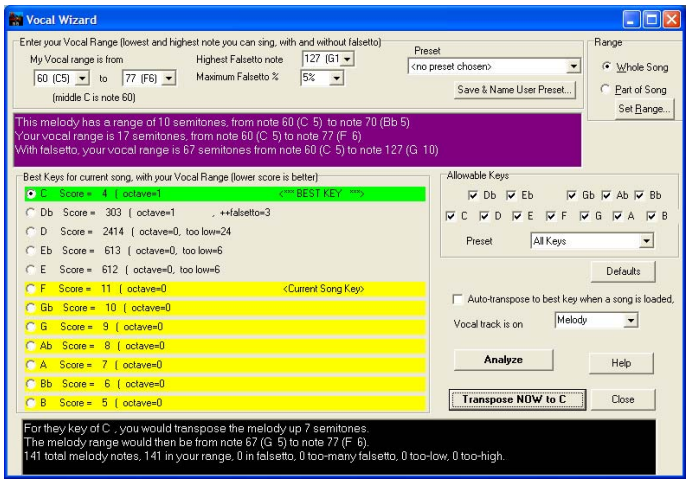


Figura 5.34 *– Interface principal do Vocal Wizard*

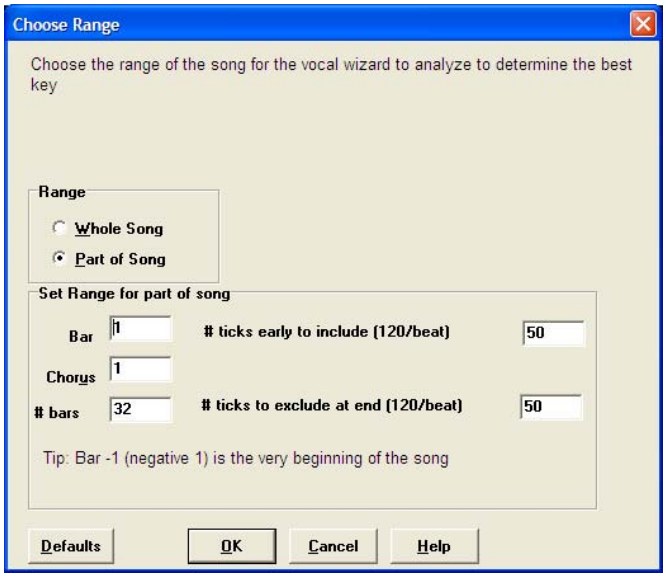


Figura 5.35 *– Interface de escolha de range*

5.9.2 Sibelius 2005

Este programa possui uma ferramenta para identificação de range por canal. O problema é que o mesmo faz isto apenas quando o arquivo MIDI é aberto em seu ambiente, e, gerada uma partitura de seu conteúdo. O mesmo marca (com uma cor diferente) a nota mais grave e a nota mais aguda de um determinado .

No caso da transposição, o Sibelius a realiza por tonalidade e por intervalos musicais (distância musical entre uma nota e outra – ver capítulo 3 - item 3.7.1), exigindo do usuário um bom conhecimento de teoria musical.

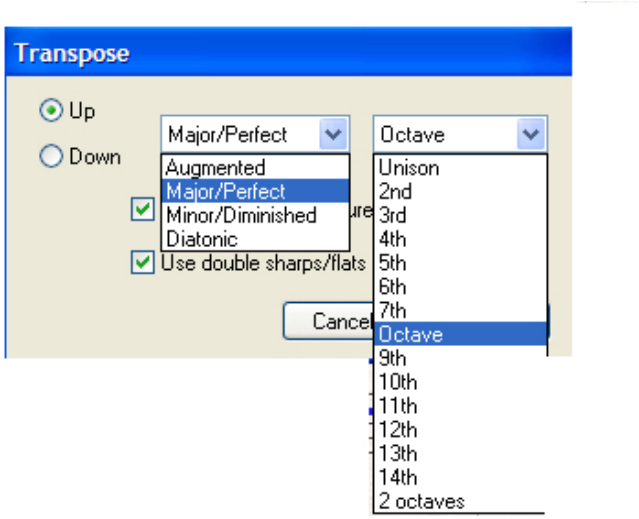


Figura 5.36 *– Transposição no Sibelius 2005*

5.9.3 Finale 2005

O Finale 2005 também apresenta apenas o método tradicional de transposição musical baseada em tonalidade. A janela da ferramenta de transposição deste programa é mostrada na figura 5.37.

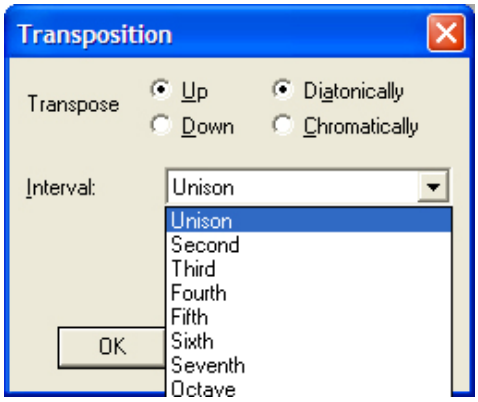


Figura 5.37 *– Transposição no Finale 2005*

5.9.4 Sonar 4 – 2005

O Sonar é um programa que atende a um público alvo formado de profissionais de estúdio e músicos seqüenciadores. Este público geralmente possui pouco conhecimento de teoria musical, de notação de partituras e de teoria musical. Dessa forma, este programa possui uma ferramenta simples de transposição que busca atender a esse tipo de usuário. A mesma é baseada em transposição por semitons[3], onde o músico escolhe quantos semitons acima ou abaixo, da tonalidade da música existente, deseja transpor. O Sonar não realiza transposições baseadas em tessitura vocal ou por tonalidade.

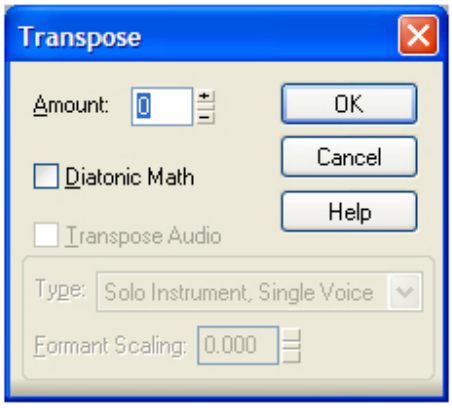


Figura 5.38 *–Transposição no Sonar 4*

[3] Um semitom é a distância em freqüência entre duas notas musicais quaisquer – ver Capítulo 3, item 3.6.1

Pontos relevantes

Analisados estes sistemas profissionais, nota-se uma carência de ferramentas que venham facilitar e viabilizar algumas tarefas do domínio musical mediada pelo computador, as quais, esta dissertação e seu produto gerado vêm contribuir, inserindo no universo da computação musical uma ferramenta para eliminar pausas de arquivos MIDI, integrada em um sistema automático de transposição baseada na tessitura vocal de um cantor ou grupo vocal qualquer.

5.10 Um estudo de caso realizado com um cantor popular usuário profissional da música seqüenciada

Este estudo foi feito com um cantor popular masculino. Ao mesmo foi apresentada a interface e sua operacionalidade. Seguindo as instruções necessárias, o mesmo procedeu da seguinte forma, acompanhado pela autora deste trabalho.

1. Em primeiro lugar, o mesmo instalou o programa, e, dentro do grupo de programa **Best Vocal 2005** criado, o mesmo clicou no executável **bestvocal2005.exe**, conforme a figura 5.39.

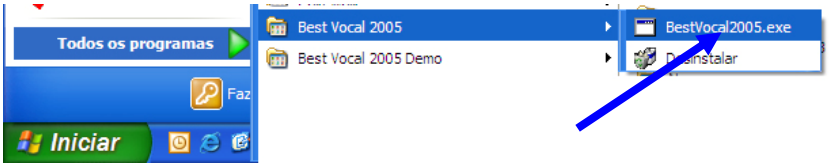


Figura 5.39 *–Abertura do programa Best Vocal 2005*

2. Ao fazer isto, o sistema abriu a interface inicial, conforme a figura 5.40.

Figura 5.40 *– Interface Inicial*

3. Seguindo as instruções, o cantor foi orientado a clicar no botão Abrir para escolher no gerenciador de arquivos do Windows um arquivo MIDI formato 0 ou formato 1, contendo a música com a respectiva melodia que desejava cantar, no caso, o arquivo PL088010.MIDI, conforme a figura 5.41.

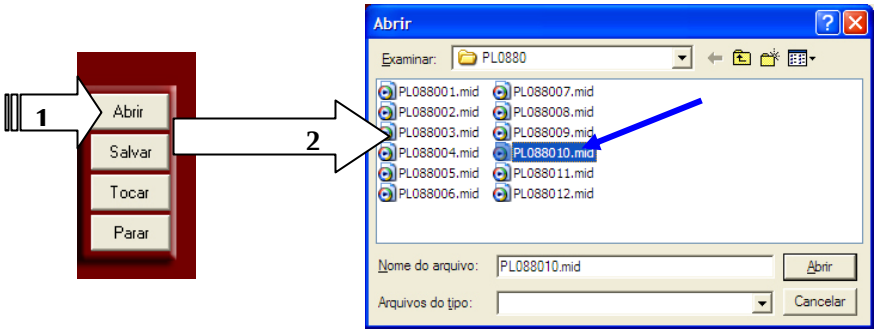


Figura 5.41 *– Abertura do arquivo MIDI*

4. Como o arquivo escolhido foi um arquivo MIDI correto, o sistema enviou uma mensagem de arquivo aberto com sucesso, conforme a figura 5.42. O cantor foi orientado para escolher um novo arquivo, caso a mensagem no campo de texto fosse a seguinte: "**Não é um arquivo MIDI**".

Figura 5.42 *– Mensagem de arquivo ABERTO COM SUCESSO*

5. Após a mensagem de arquivo aberto com sucesso, o mesmo, foi orientado a mandar executar (tocar) o arquivo MIDI, clicando no botão **Tocar,** para que identificasse qual seria o instrumento que estaria reproduzindo a melodia que desejava cantar. O mesmo foi orientado que poderia interromper a qualquer momento a execução clicando no botão **Parar**.

***Figura 5.43** – Botão Tocar*

6. Enquanto escutava a música, o cantor clicou no botão **Tessitura** para identificar qual seria o canal MIDI que estaria reproduzindo o instrumento da melodia que desejava cantar. O cantor identificou o instrumento **Violão Elétrico com cordas de aço,** ou **uma Guitarra** como sendo o instrumento que estava executando a melodia. Olhando os canais MIDI impressos na janela Tessitura, o que mais se aproximou foi o **ElectricGuitar[jazz]** no canal MIDI 4, cujo range de notas estava entre as notas **si2** e **re4** (inclusive).

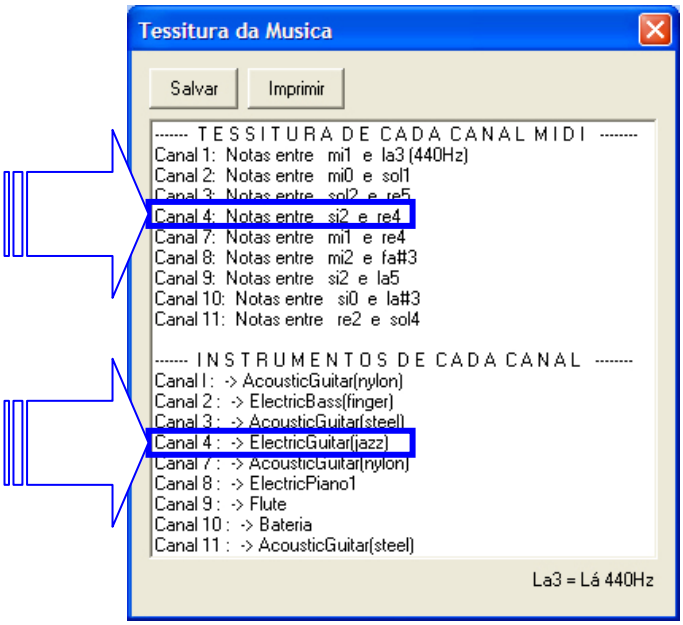


***Figura 5.44** – Janela Tessitura da Musica: escolha do canal da melodia*

Obs. O cantor, depois de proceder desta forma, chegou à conclusão que seria mais prático ver (visualizar na janela **Tessitura da Musica**) os instrumentos da música antes de mandar tocá-la, para facilitar identificar o instrumento desejado (instrumento que estava executando a melodia).

7. De posse do conhecimento do canal MIDI que contém a melodia (Canal 4), o cantor foi instruído a abrir a ferramenta de transposição para identificar seu

range vocal e transpor a música de forma a centralizar o range da melodia com seu range vocal. Para tanto, ele clicou no botão **Transpositor**, onde a seguinte janela foi aberta:

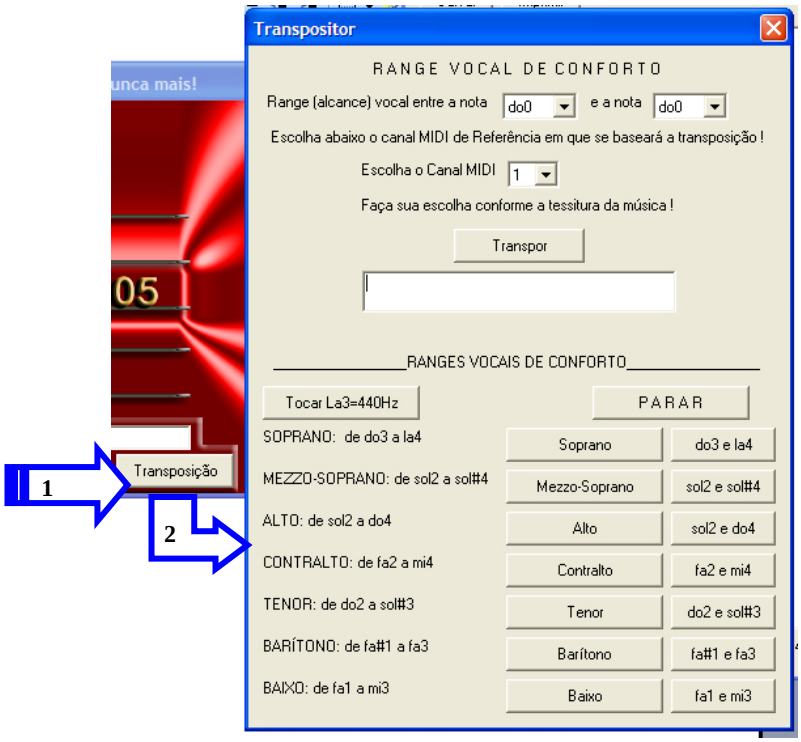


Figura 5.45 *– Janela Transpositor*

8. O cantor foi orientado a identificar os limites de seu range vocal, onde julgava que a voz dele estivesse agradável. Para tanto, ele poderia identificar as notas musicais limites de seu range vocal utilizando as três opções disponibilizadas pelo sistema:
   - escalas com classificações vocais padronizadas

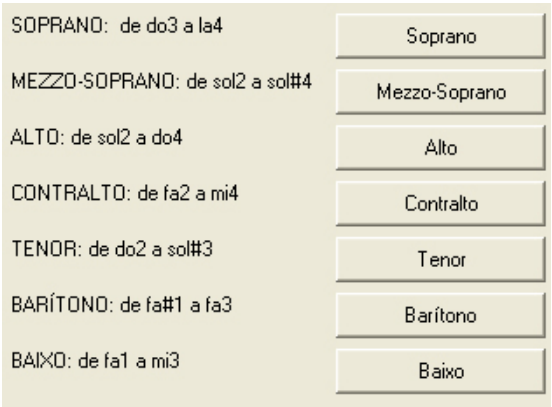


Figura 5.46 *– Escalas com classificações vocais padronizadas (de Soprano a Baixo)*

- notas limites destas escalas

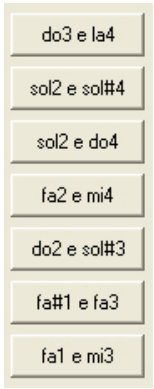


Figura 5.47 *– Notas limites das escalas com classificações vocais*

- um player (tocador) de sons de notas musicais, desde a nota **Dó 0** (32,7032 Hz) até a nota **Si 5** (1975,5332 Hz).

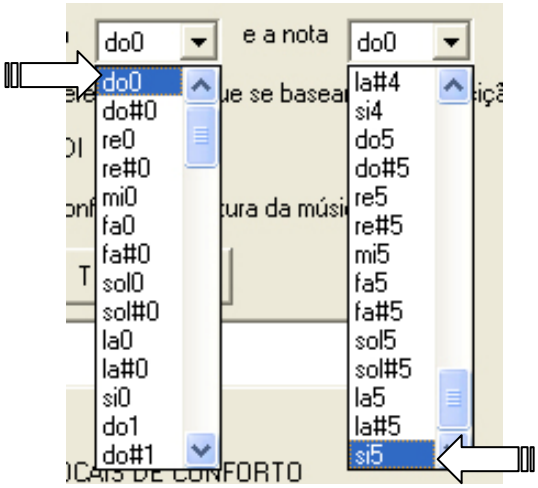


Figura 5.48 *- Sons das notas musicais: de **Dó 0** a **Si 5***

9. Realizadas as audições das notas, o cantor chegou à conclusão que seu range de conforto coincidia com o range de classificação de um Barítono, ou seja, de fa#1 a fa3 (inclusive). Assim, o mesmo foi orientado a entrar com as notas limites de seu range no menu pop-up do Range Vocal de Conforto, conforme mostra a figura 5.49.

Figura 5.49 *- Menu pop-up do Range Vocal de Conforto*

10. Ao se analisar a figura 5.44 pode-se perceber que o range da melodia, Canal MIDI 4, estava entre as notas **si2** e **re4** (inclusive), e desta forma, ele não conseguiria cantar (ou desafinaria) várias notas, o que, ao tentar, foi o que ocorreu.
11. Assim, o mesmo foi orientado a transpor a música para sua região de conforto, e, isto só ocorreria, se seu range vocal fosse superior (tivesse mais notas musicais) do que o range da melodia em questão. De si2 a re4 tem-se 16 notas musicais, e, de fa#1 a fa3 tem-se 24 notas, tornando possível a transposição. Foi explicado ao mesmo que não se preocupasse em fazer contas. Caso a transposição não fosse possível, o sistema enviaria uma mensagem, e desta forma, ele poderia tirar esta música do repertório, já que seu range era insuficiente para cantar no momento (poderia ser que em outra ocasião fosse possível).
12. Orientou-se ao cantor que escolhesse o canal MIDI desejado (o Canal 4), no menu pop-up correspondente, conforme mostra a figura 5.50.

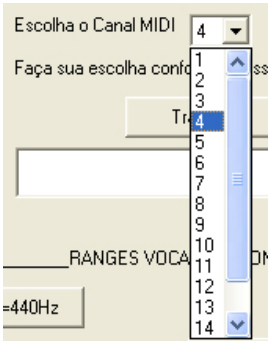


Figura 5.50 *- Escolha do Canal MIDI da melodia no menu pop-up*

13. Feito isto, o mesmo clicou no botão Transpor e aguardou a transposição ser realizada, com a mensagem no campo de texto indicando "**Transposição realizada**", conforme mostra a figura 5.51.

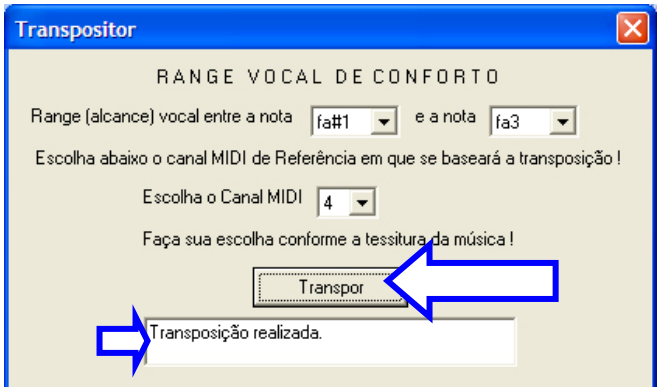


Figura 5.51 *– Transposição realizada*

14. O mesmo foi orientado a clicar novamente no botão tessitura para observar o resultado da transposição, onde observou que o range do Canal 4 foi alterado, ficando os limites entre **si1** e **re3**, limites dentro de seu range vocal.

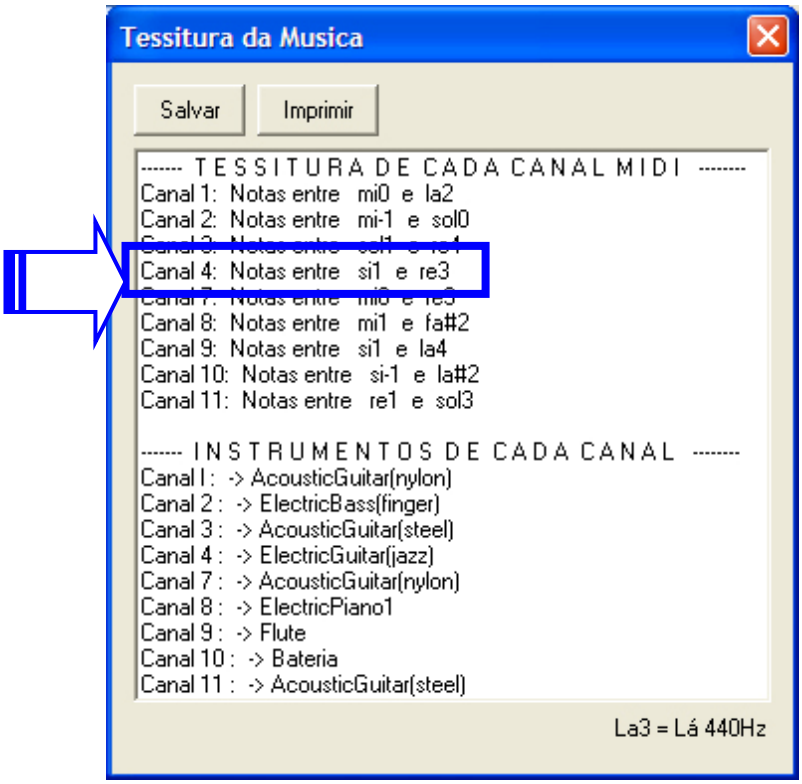


Figura 5.52 *– Resultado da transposição (notas limites entre **si1** e **re3**)*

15. O cantor desejou ouvir o resultado, mas, como o sistema não possui preview, o mesmo foi orientado a primeiro salvar o arquivo MIDI com um novo nome (para não perder o original), e, desta forma, poderia ouvir o resultado logo após o salvamento ser completado (o que levaria alguns minutos).

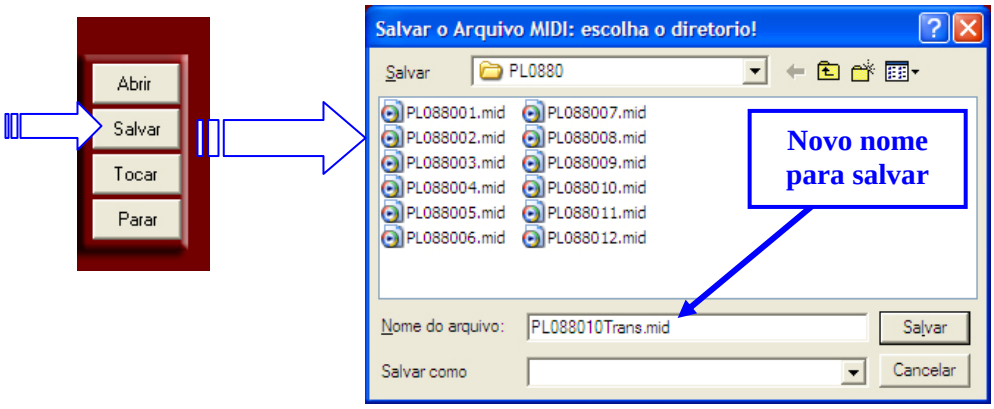


Figura 5.53 *– Arquivo MIDI salvo com um novo nome*

16. Para ouvir o resultado, bastava, novamente, clicar no botão **Tocar.**
17. Como o cantor ainda não havia memorizado a música, foi-lhe dito que poderia utilizar a ferramenta **letra** para extrair a mesma da música, caso o autor da seqüência tivesse registrado a letra no arquivo. Ele clicou no botão **Letra** e imprimiu a mesma para cantar na nova tonalidade transposta pelo sistema.

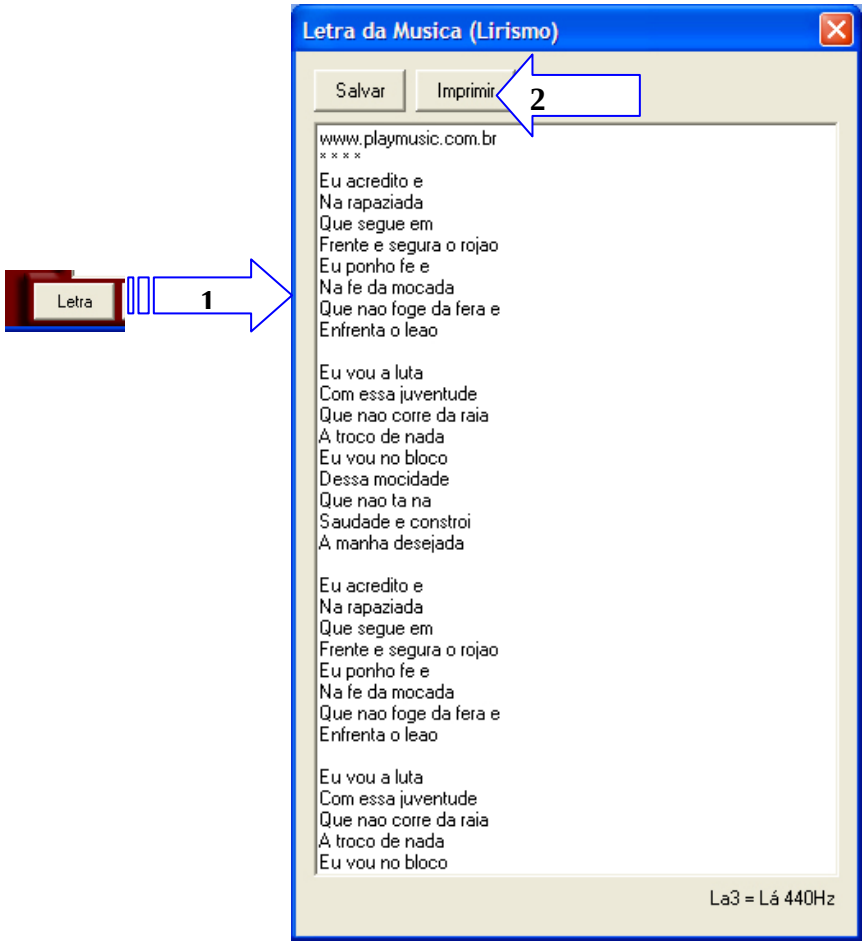


Figura 5.54 *– Abertura e Impressão da letra da música*

18. Foi dito ao cantor que ele poderia ainda mudar os instrumentos da música e eliminar as pausas iniciais que porventura existissem.

Conclusão do Cantor:

"O software Best Vocal 2005 realmente foi bastante útil para adequar a música à tonalidade, ao meu range vocal, evitando que, a qualquer momento, eu desafinasse, permitindo que eu conseguisse emitir todas as notas das melodias de meu repertório. Verifiquei que existiam outras possibilidades de transposição para as músicas, que não a definida pelo sistema , já que sobraram notas no meu range, tanto na região aguda quanto na grave. Desta forma, como a minha voz fica melhor quanto mais grave estiver dentro do meu range vocal, o ideal não seria centralizar um range com o outro e sim disponibilizar as possíveis opções de transposição pra que eu pudesse escolher a que melhor se adequasse ao meu timbre. A falta de um preview também foi algo incômodo, já que tive que esperar o arquivo ser salvo para depois ver como ficou. No caso de um número grande de músicas exigiria um tempo grande ocioso do músico. A criação de um batch, um sistema pra aplicar a transposição a um lote de música MIDI também minimizaria este problema."

Capítulo 6

Conclusões e trabalhos futuros

Esta dissertação teve como foco a implementação de soluções aplicadas ao domínio musical, em especial em computação sônica, bem como na contribuição profilática relativa à saúde e bom funcionamento do aparelho fonador humano em suas atividades de canto.

Os objetivos e metas firmados no Capítulo 1 foram cumpridos em sua plenitude, a saber:

1-Estudo da teoria e conceitos sobre sinais, ruídos e do processo de digitalização de sinais sonoros

Conforme pôde ser avaliado no capítulo 2, bem como nos demais capítulos, um longo estudo foi feito em torno deste tema e conceitos, os quais possibilitaram que se pudesse implementar soluções aderentes ao domínio musical e aos seus usuários.

2-Estudo dos formatos e protocolos de registro dos sinais sonoros

O estudo geral do processo de digitalização dos sinais analógicos, conceitos e parâmetros foram vistos no capítulo 2, apresentando como os sinais são armazenados em arquivos formato Wave (.wav).

3-Estudo do protocolo MIDI, hardware e software, bem como dos formatos de arquivos para registro de músicas seqüenciadas

Abordou-se, no capítulo 2, o Protocolo de comunicação MIDI e seus arquivos SMF formato 0 e formato 1, apresentando com detalhes os pontos mais relevantes e necessários para a implementação das ferramentas de leitura, análise e transformação dos arquivos MIDI em linguagem funcional CLEAN.

4-Projeto e implementação de um sistema de transposição automática de sequências musicais MIDI de acordo com a tonalidade de conforto de um determinado cantor.

Além de se projetar e implementar uma solução computacional em paradigma funcional (linguagem de programação CLEAN), para transposição automática de sequências musicais MIDI, foram projetadas outras ferramentas de interesse dos usuários alvo deste sistema para tornar o produto mais atrativo e relevante tanto comercialmente quanto em potencilidades e inedismos em potencial. Pode-se citar, entre estas ferramentas, a de eliminar pausas de arquivos tanto em formato 0 quanto em formato 1, ferramenta esta inexistente comercialmente no mercado atual de softwares para computer music. Foi também implementada uma ferramenta para extração do lirismo de um arquivo MIDI, permitindo ao usuários, salvar, editar e imprimir as letras das músicas já pré-formatadas pelo sistema.

5-Estudo de casos

No capítulo 5 foi apresentado um amplo estudo de caso (simulado) aplicando as potencialidades do sistema gerado (Best Vocal 2005), na adequação da tonalidade de uma música para os ranges de um quarteto vocal. Com tal experimento concluiu-se que o sistema atendeu os requisitos e objetivos esperados. Finalizada a simulação, foi também realizado um estudo de caso real com um cantor popular masculino, usuário profissional da música seqüenciada, onde o mesmo concluiu:

> *"O software Best Vocal 2005 realmente foi bastante útil para adequar a música à tonalidade, ao meu range vocal, evitando que, a qualquer momento, eu desafinasse, permitindo que eu conseguisse emitir todas as notas das melodias de meu repertório*."

Além disto, o programa foi colocado à venda em um site virtual no endereço http://paginas.terra.com.br/educacao/bestdream/, onde o mesmo teve uma boa aceitação, e, mesmo sem divulgação, vendeu-se aproximadamente 50 unidades para músicos de todo o país, os quais deram retorno em problemas encontrados na utilização e funcionabilidade do programa, permitindo que os erros encontrados fossem todos corrigidos, alcançando o software inicial sua oitava atualização com os corretores dos "bugs". Feitas as últimas correções, desde outubro de 2005 nenhum usuário relatou mais algum problema que carecesse de soluções.

6-Conclusão pela confrontação dos resultados e estudo de casos obtidos, com os objetivos propostos inicialmente

Os resultados obtidos através de estudo de casos, bem como de retorno de clientes que utilizam profissionalmente o sistema há mais de um ano, concluem por se ter alcançado com sucesso os objetivos e metas propostas inicialmente, podendo-se afirmar que as mesmas foram ultrapassadas, como é o caso do sistema permitir que qualquer tipo de arquivo MIDI possa ser aberto e manipulado pelo mesmo, sendo isto realizado de forma totalmente automática e transparente ao usuário. Apesar dos retrabalhos originados

devido a particularidades de equipamentos MIDI proprietários comercializados em todo o mundo, tais problemas fizeram com que a equipe de computer music da FEELT, da qual a presente mestranda faz parte, crescesse bastante em conhecimento e potencialidades na manipulação de arquivos MIDI, podendo afirmar que não existe praticamente mais nenhum desafio em se implementar sistemas de análise utilizando tal protocolo e seus formatos.

7-Levantamento e identificação de propostas para trabalhos futuros

Quanto à identificação de metas para trabalhos futuros, pode-se citar:

1- O programa atual não possui um preview que permita ao usuário testar as modificações feitas na música pelo Best Vocal, tais como: mudança de instrumento, eliminação de pausas e transposição, antes de salvar. Na versão atual, o mesmo só terá noção real dos resultados após o salvamento.
2- Seria interessante se implementar uma ferramenta que permitisse ao usuário cantar em um microfone e o sistema fizesse sua classificação vocal automaticamente. Para tanto dever-se-á implementar fonetogramas que traduzam a realidade da qualidade sonora do som produzido pelo cantor, e não apenas quantifique os limites das freqüências sonoras que o mesmo emitir. Além destes fonetogramas, dever-se-á implementar um sistema especialista, inteligente ou não, que modele o *feeling* de um profissional reconhecido no mercado que realize reconhecidamente bem a arte da classificação vocal.
3- O sistema gasta um tempo relativamente longo para remontar e salvar o arquivo MIDI mantendo as características iniciais da música original (é o único sistema que faz isto). Para torná-lo mais atrativo comercialmente, deve-se reimplementar o sistema de forma a trabalhar diretamente no arquivo, e não em uma imagem armazenada em uma lista de inteiros. Desta forma o sistema ficará sensivelmente mais rápido.

4- Implementar um arquivo de batch para aplicar a ferramenta de transposição a um lote de arquivos MIDI, dado o range do cantor, evitando que o mesmo tenha que ficar horas esperando o computador concluir cada transposição.
5- Implementar a opção do sistema mostrar todas as possíveis transposições que a música possa ter, sem que, com isto, a melodia da mesma saia do range do cantor. Isto permitirá que o cantor escolha a tonalidade que melhor se enquadre em seu timbre vocal (onde a melodia fica mais bonita para sua voz).
6- Implementar a opção para transpor, não só melodias para cantores, mas para adaptar um tipo de instrumento a uma determinada melodia, conhecida a tessitura do mesmo. Para tanto, deve-se construir um banco com os ranges de cada instrumento, de tal forma que o sistema possa automaticamente proceder a tal transposição, sem que o usuário tenha que conhecer os ranges de cada instrumento existente. Ferramenta bastante útil para quando se deseja um arranjo para um instrumento específico e não se encontra no mercado.
7- Dotar o sistema da potencialidade de avisar o músico quando um instrumento de um determinado canal ultrapassar a tessitura prevista para o mesmo, após realizada a transposição, já que, então, o som do mesmo soará estranho. De posse deste conhecimento, o músico poderá ter a opção de eliminar um instrumento da seqüência MIDI.
8- Implementar um sistema que permita a um arranjador entrar com o range de vários cantores, de tal modo que o sistema automaticamente escolha a melhor transposição da música para que todos os cantores consigam cantar pelo menos uma melodia da mesma.

Como o foco do trabalho é bastante centrado em uma aplicação, um problema específico, limitam-se as perspectivas de trabalhos futuros nestas três linhas, sendo uma delas, a classificação automática de tessitura vocal. Esta classificação possui um grau relativamente grande de dificuldade intrínseca, o qual geraria um interessante trabalho de doutorado, já que inexistem soluções automáticas e inteligentes para realizar tal tarefa, a qual possui um forte componente de intuição na classificação, além de fatores técnicos conhecidos e ferramentas matemáticas convencionais para análise de sinais pouco efetivas para uma solução apenas funcional do problema.

Referências

Capítulo 1

[1] LIMA, L.; C.A.L., et al. **Automatic Learning Composition Base don Polyphonic Wave Signals and Musical Histograms.** Tatra Mt Match. Publ. 23, 2001.

[2] LIMA, L. V.; DOVICCHI, J. C.; et al. **Music Based Coefficients for Wavelets Transform: A Contribuition to Musical Signal Análisis.** 108th AES Convention, Paris, 2000.

[3] HEIDEMAN, M. T.; JOHNSON, D. H.; BURRUS, C. S. **Gauss and the history of the fast Fourier transform.** IEEE Acoustics, Speech, and Signal Processing.

[4] BRIGHAM, E.O., **The Fast Fourier Transform.** Prentice Hall, Englewood Cliffs, NJ, 1975.

[5] COPE, D. **Experiments Musical in Intelligence (EMI): Non-linear Linguistic-Based Composition Interface**, vol., 18, pp. 117-139, 1989.

[6] COPE, D. **Computer Modelling of Musical Intelligence in EMI.** Computer Music Journal, vol. 16, no. 2, Summer, pp. 69-83, 1992.

[7] COPE, D. **Music & Lisp.** OH Expert, March, pp.26-33, 1988.

[8] LIMA, L. V. **Um Sistema de Composição Musical Dirigido por Estilo.** Tese de Doutorado, Universidade de São Paulo, São Paulo - Brasil, 1998.

[9] GARDNER, H., **Multiple Inteligences**: The theory em Practice, basic books inc., Traduzido por Maria Adriana Veríssimo Veronese, Artmed, Porto Alegre, 1993. 2000.257p.

[10] LIMA, Luciano Vieira. Real time high performance plataform for e-learning with high resolutions áudio, frames and vídeo file. In: **3ª. Conferência do PGL: Partnership in Global Learning "Consolidando Experiências em e-Learning",** FGV, São Paulo – Brasil, 2005.

[11] LIMA, Luciano Vieira; MACHADO, André Campos; PINTO, Marília Mazzaro. **Cakewalk sonar 2.0 : Seqüenciamento e Técnicas de Estúdio Audiodigital.** 2ª. ed. São Paulo: Editora Érica Ltda, 2003.

[12] LIMA, Luciano Vieira; LIMA; Sandra Fernandes de Oliveira; MACHADO, André Campos; PINTO, Marília Mazzaro. **Computação Musical : Arranjo Automático, Conversão de CD em MIDI e MIDI em Áudio utilizando o Band-in-a-Box, Virtual Sound Canvas 3.23 e Akoff Music Composer 2.0.** 1ª. ed. São Paulo: Editora Érica Ltda, 2004.

[13] LIMA, Luciano Vieira; LIMA; MACHADO, André Campos; PINTO, Marília Mazzaro. **Finale 2004: Editoração de Partituras, Composição e Arranjo.** 1ª. ed. São Paulo: Editora Érica Ltda, 2004.

[14] PASCHOAL, Fausto de. **Música no Computador com Sibelius 3 para Windows.** 1ª. ed. São Paulo: Editora Érica Ltda, 2004.

[15] LIMA, Luciano Vieira; MACHADO, André Campos; PINTO, Marília Mazzaro. **Encore 4.5.4 : Editoração de Partituras.** 1a. ed. São Paulo: Editora Érica Ltda, 2003.

[16] BEHLAU, Mara; Torres Maria Lúcia Graziano Magalhães. Medidas do Perfil de Extensão Vocal, Tempo Máximo de Fonação e Relação s/z em Cantores de um Coral Amador, Pré e Pós-Ensaio. In: **A Voz do Especialista.** Vol 1, p 109 – 131. Rio de Janeiro: Livraria e Editora Revinter Ltda, 2001

[17] MILLER, Donald G.; SCHUTTE, Harm K.; SULTER, Arend M.; WIT, Hero P. **A structured approach to voice range profile (phonetogram) analysis.** Disponível em: http://www.ub.rug.nl/eldoc/dis/medicine/a.m.sulter/c7.pdf.

Capítulo 2

[18] LIMA, Luciano Vieira; LIMA; Sandra Fernandes de Oliveira; MACHADO,André Campos; PINTO, Marília Mazzaro.**Sound Forge 8.0: Gravação ao Vivo, Restauração de Sons de LPs e Masterização áudio Digital.** 1ª. ed. São Paulo: Editora Érica Ltda, 2005.

[19] MACHADO, André Campos. Tradutor de Arquivos MIDI para Texto Utilizando Linguagem Funcional CLEAN. Dissertação de Mestrado em Engenharia Elétrica, Uberlândia: UFU, 2001.

[20] ROLAND-MIESZKOWSKI,Marek. **Consequences of Nyquist Theorem for Acoustic signals Stored in Digital Format.** Disponível em: http://www.digital-recordings.com/pubneq.html.

[21] **MIDI Detailed Specification 1.0.** International MIDI Association, 1983, 1991, 1994.

[22] LOPES, Guilherme Francisco Lopes. **Desenvolvimento de uma biblioteca e uma calculadora Midi, em linguagem funcional, para análise e manipulação de Arquivos padrão Midi (SMF) formato 0 e 1.** Dissertação de Mestrado em Engenharia Elétrica, Uberlândia: UFU, 2004.

[23] RATTON, Miguel. **Criação de Música e Sons no Computador.** Rio de Janeiro: Editora Campus, 1995.

Capítulo 3

[24] GROUT, D.; PALISCA, C. **História da Música Ocidental.** Lisboa: Gradiva, 1994.

[25] LIMA, Florêncio de Almeida Lima. **Elementos Fundamentais da Música.** 4ª.ed. Rio de Janeiro,1958.

[26] CIFRAS, MV Portal de.**Mini-Curso Básico de Técnica Vocal: Classificação Vocal .***Cap.4.* Disponível em: http://www.mvhp.com.br.

[27] BAS, **Trattato di Forma Musicale.** Milão: Ricordi, 1964.

[28] ZAMACOIS, Joaquín. **Curso de Formas Musicales.** Cuarta edición. Barcelona: Editorial Labor S.A.,1960.

Capítulo 4

[29] LIMA, Luciano Vieira; RUFINO, Hugo Leonardo Pereira; GOULART, Reane Franco; CURY FILHO, Reny. **Linguagem Funcional CLEAN: uma solução moderna, eficiente e aderente à modelagem de funções e ao ensino da Matemática.** In: CONGRESSO NACIONAL DE MATEMÁTICA APLICADA E COMPUTACIONAL, São José do Rio preto, 2003, v.1, p.1-7.

[30] WINSTON & HORN. **LISP.** 3rd Edition. Addison-Wesley Publishing Company, 1980.

[31] LIMA, L. V. et al. **Editing Standard MIDI File in Functional Language CLEAN.** Diderot 99, Vienna, 1999.

[32] WELLESLEY,Alessandro Barros. **Programação Funcional com Clean 2.1 para Windows.** Uberlândia: Editora Pietro Ltda, 2004.

[33] BIRD, Richard. **Introduction to Functional Programming using Haskell, 2nd edition,** Prentice Hall press,1998.

[34] THOMPSON, Simon. **Haskell: The Craft of Functional Programming,** Second Edition, Addison-Wesley, 507 pages, paperback, 1999.

Capítulo 5

[35] LIMA, Sandra Fernandes de Oliveira; RUFINO, Hugo Leonardo Pereira; LOPES, Guilherme Francisco; GOULART, Reane Franco; LIMA, Luciano Vieira. **Matemática aplicada à detecção de tessitura vocal e à transposição automática de músicas em arquivos MIDI padrão.** In: CONGRESSO NACIONAL DE MATEMÁTICA APLICADA E COMPUTACIONAL, São José do Rio preto, 2003, v.1, p.1-7.

[36] MARTINS, Gláucia Macedo Mendes. **Projeto e Implementação de uma Interface Visual Interativa para Montagens de Conteúdo de Autoria Multimídia.** Dissertação de Mestrado em Engenharia Elétrica, Uberlândia: UFU, 2005.

ANEXO 1 - ASCII

A Tabela ASCII (American Standard Code for Information Interchange) é usada pela maior parte da industria de computadores para a troca de informações. Cada caractere é representado por um código de 8 bits (um byte). Abaixo mostramos a tabela ASCII de 7 bits. Existe uma tabela extendida para 8 bits que inclue os caracteres acentuados.

Tabela ASCII – 127 códigos (7 BITS)

Caractere	Decimal	Hexadecimal	Binário	Comentário
NUL	00	00	0000 0000	Caracter Nulo
SOH	01	01	0000 0001	Começo de cabeçalho de transmissão
STX	02	02	0000 0010	Começo de texto
ETX	03	03	0000 0011	Fim de texto
EOT	04	04	0000 0100	Fim de transmissão
ENQ	05	05	0000 0101	Interroga
ACK	06	06	0000 0110	Confirmação
BEL	07	07	0000 0111	Sinal sonoro
BS	08	08	0000 0100	Volta um caracter
HT	09	09	0000 1001	Tabulação Horizontal
LF	10	0A	0000 1010	Próxima linha
VT	11	0B	0000 1011	Tabulação Vertical
FF	12	0C	0000 1100	Próxima Página
CR	13	0D	0000 1101	Início da Linha
SO	14	0E	0000 1110	Shift-out
SI	15	0F	0000 1111	Shift-in
DLE	16	10	0001 0000	Data link escape
D1	17	11	0001 0001	Controle de dispositivo
D2	18	12	0001 0010	Controle de dispositivo
D3	19	13	0001 0011	Controle de dispositivo
D4	20	14	0001 0100	Controle de dispositivo
NAK	21	15	0001 0101	Negativa de Confirmação
SYN	22	16	0001 0110	Synchronous idle
ETB	23	17	0001 0111	Fim de transmissão de bloco
CAN	24	18	0001 1000	Cancela
EM	25	19	0001 1001	Fim de meio de transmissão
SUB	26	1A	0001 1010	Substitui

ESC	**27**	**1B**	**0001 1011**	**Escape**
FS	**28**	**1C**	**0001 1100**	**Separador de Arquivo**
GS	**29**	**1D**	**0001 1101**	**Separador de Grupo**
RS	**30**	**1E**	**0001 1110**	**Separador de registro**
US	**31**	**1F**	**0001 1111**	**Separador de Unidade**
Espaço	**32**	**20**	**0010 0000**	
!	**33**	**21**	**0010 0001**	
"	**34**	**22**	**0010 0010**	
#	**35**	**23**	**0010 0011**	
$	**36**	**24**	**0010 0100**	
%	**37**	**25**	**0010 0101**	
&	**38**	**26**	**0010 0110**	
'	**39**	**27**	**0010 0111**	
(	**40**	**28**	**0010 1000**	
)	**41**	**29**	**0010 1001**	
*****	**42**	**2A**	**0010 1010**	
+	**43**	**2B**	**0010 1011**	
,	**44**	**2C**	**0010 1100**	
-	**45**	**2D**	**0010 1101**	
.	**46**	**2E**	**0010 1110**	
/	**47**	**2F**	**0010 FFFF**	
0	**48**	**30**	**0011 0000**	
1	**49**	**31**	**0011 0001**	
2	**50**	**32**	**0011 0010**	
3	**51**	**33**	**0011 0011**	
4	**52**	**34**	**0011 0100**	
5	**53**	**35**	**0011 0101**	
6	**54**	**36**	**0011 0110**	
7	**55**	**37**	**0011 0111**	
8	**56**	**38**	**0011 1000**	
9	**57**	**39**	**0011 1001**	
:	**58**	**3A**	**0011 1010**	
;	**59**	**3B**	**0011 1011**	
<	**60**	**3C**	**0011 1100**	
=	**61**	**3D**	**0011 1101**	
>	**62**	**3E**	**0011 1110**	
?	**63**	**3F**	**0011 1111**	
@	**64**	**40**	**0100 0000**	
A	**65**	**41**	**0100 0001**	
B	**66**	**42**	**0100 0010**	
C	**67**	**43**	**0100 0011**	
D	**68**	**44**	**0100 0100**	
E	**69**	**45**	**0100 0101**	
F	**70**	**46**	**0100 0110**	

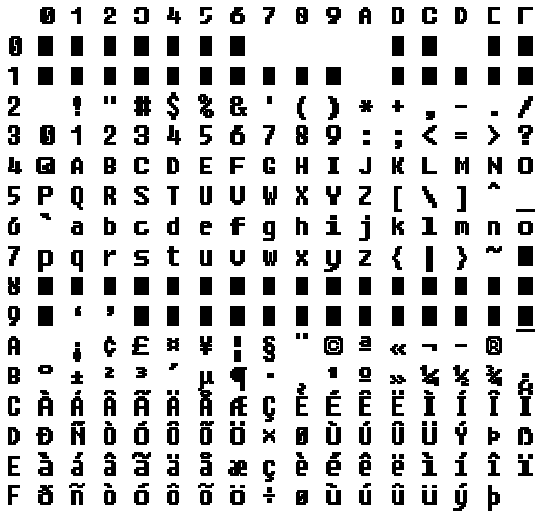
G	**71**	**47**	**0100 0111**
H	**72**	**48**	**0100 1000**
I	**73**	**49**	**0100 1001**
J	**74**	**4A**	**0100 1010**
K	**75**	**4B**	**0100 1011**
L	**76**	**4C**	**0100 1100**
M	**77**	**4D**	**0100 1101**
N	**78**	**4E**	**0100 1110**
O	**79**	**4F**	**0100 1111**
P	**80**	**50**	**0101 0000**
Q	**81**	**51**	**0101 0001**
R	**82**	**52**	**0101 0010**
S	**83**	**53**	**0101 0011**
T	**84**	**54**	**0101 0100**
U	**85**	**55**	**0101 0101**
V	**86**	**56**	**0101 0110**
W	**87**	**57**	**0101 0111**
X	**88**	**58**	**0101 1000**
Y	**89**	**59**	**0101 1001**
Z	**90**	**5A**	**0101 1010**
[	**91**	**5B**	**0101 1011**
\	**92**	**5C**	**0101 1100**
]	**93**	**5D**	**0101 1101**
^	**94**	**5E**	**0101 1110**
_	**95**	**5F**	**0101 1111**
`	**96**	**60**	**0110 0000**
a	**97**	**61**	**0110 0001**
b	**98**	**62**	**0110 0010**
c	**99**	**63**	**0110 0011**
d	**100**	**64**	**0110 0100**
e	**101**	**65**	**0110 0101**
f	**102**	**66**	**0110 0110**
g	**103**	**67**	**0110 0111**
h	**104**	**68**	**0110 1000**
i	**105**	**69**	**0110 1001**
j	**106**	**6A**	**0110 1010**
k	**107**	**6B**	**0110 1011**
l	**108**	**6C**	**0110 1100**
m	**109**	**6D**	**0110 1101**
n	**110**	**6E**	**0110 1110**
o	**111**	**6F**	**0110 1111**
p	**112**	**70**	**0111 0000**
q	**113**	**71**	**0111 0001**
r	**114**	**72**	**0111 0010**
s	**115**	**73**	**0111 0011**

t	**116**	**74**	**0111 0100**
u	**117**	**75**	**0111 0101**
v	**118**	**76**	**0111 0110**
w	**119**	**77**	**0111 0111**
x	**120**	**78**	**0111 1000**
y	**121**	**79**	**0111 1001**
z	**122**	**7A**	**0111 1010**
{	**123**	**7B**	**0111 1011**
|	**124**	**7C**	**0111 1100**
}	**125**	**7D**	**0111 1101**
~	**126**	**7E**	**0111 1110**
DELETE	**127**	**7F**	**0111 1111**

TABELA ASCII 256 CÓDIGOS (8 BITS)

Código ASCII estendido para IBM PC

De uma forma geral, os caracteres ASCII possuem valores de 0 a 127 (decimal) ou 7F (hexadecimal). Contudo, quando o IBM PC foi desenvolvido a placa de vídeo possuía apenas um byte para definir cada caractere em um monitor de 80x25 caracteres. Com o avanço da tecnologia os monitores passaram a possuir muito mais definição, superando e muito o número máximo de caracteres por tela desde sua concepção. Sendo assim porque não se adicionar mais um bit por caractere? Porque não inventar, representar, outros 128 caracteres novos? Para isto foi criado o **Código ASCII Estendido**, o qual é mostrado conforme utilizado pelo Windows:



Para quem quiser converter para decimal e não sabe, segue uma tabela prática. (você pode utilizar a calculadora do Windows.

Convertendo Hexadecimal para Decimal

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F.
0	000	001	002	003	004	005	006	007	008	009	010	011	012	013	014	015
1	016	017	018	019	020	021	022	023	024	025	026	027	028	029	030	031
2	032	033	034	035	036	037	038	039	040	041	042	043	044	045	046	047
3	048	049	050	051	052	053	054	055	056	057	058	059	060	061	062	063
4	064	065	066	067	068	069	070	071	072	073	074	075	076	077	078	079
5	080	081	082	083	084	085	086	087	088	089	090	091	092	093	094	095
6	096	097	098	099	100	101	102	103	104	105	106	107	108	109	110	111
7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
8	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
9	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
A	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
B	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
C	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
D	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
E	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
F	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

ANEXO 2 –

INSTRUMENTOS DO CANAL 1-9, 11-16

AcousticGrandPiano	0
BrightAcousticPiano	1
ElectricGrandPiano	2
Honky-tonkPiano	3
ElectricPiano1	4
ElectricPiano2	5
Harpsichord	6
Clavi	7
Celesta	8
Glockenspiel	9
MusicBox	10
Vibraphone	11
Marimba	12
Xylophone	13
TubularBells	14
Dulcimer	15
DrawbarOrgan	16
PercussiveOrgan	17
RockOrgan	18
ChurchOrgan	19
ReedOrgan	20
Accordion	21
Harmonica	22
TangoAccordion	23
AcousticGuitar(nylon)	24
AcousticGuitar(steel)	25
ElectricGuitar(jazz)	26
ElectricGuitar(clean)	27
ElectricGuitar(muted)	28
OverdrivenGuitar	29
DistortionGuitar	30
Guitarharmonics	31
AcousticBass	32
ElectricBass(finger)	33
ElectricBass(pick)	34

FretlessBass	35
SlapBass1	36
SlapBass2	37
SynthBass1	38
SynthBass2	39
Violin	40
Viola	41
Cello	42
Contrabass	43
TremoloStrings	44
PizzicatoStrings	45
OrchestralHarp	46
Timpani	47
StringEnsemble1	48
StringEnsemble2	49
SynthStrings1	50
SynthStrings2	51
ChoirAahs	52
VoiceOohs	53
SynthVoice	54
OrchestraHit	55
Trumpet	56
Trombone	57
Tuba	58
MutedTrumpet	59
FrenchHorn	60
BrassSection	61
SynthBrass1	62
SynthBrass	63
SopranoSax	64
AltoSax	65
TenorSax	66
BaritoneSax	67
Oboe	68
EnglishHorn	69
Bassoon	70
Clarinet	71
Piccolo	72
Flute	73
Recorder	74
PanFlute	75
BlownBottle	76
Shakuhachi	77
Whistle	78
Ocarina	79
Lead1(square)	80
Lead2(sawtooth)	81
Lead3(calliope)	82

Lead4(chiff)	83
Lead5(charang)	84
Lead6(voice)	85
Lead7(fifths)	86
Lead8(bass+lead)	87
Pad1(newage)	88
Pad2(warm)	89
Pad3(polysynth)	90
Pad4(choir)	91
Pad5(bowed)	92
Pad6(metallic)	93
Pad7(halo)	94
Pad8(sweep)	95
FX1(rain)	96
FX2(soundtrack)	97
FX3(crystal)	98
FX4(atmosphere)	99
FX5(brightness)	100
FX6(goblins)	101
FX7(echoes)	102
FX8(sci-fi)	103
Sitar	104
Banjo	105
Shamisen	106
Koto	107
Kalimba	108
Bagpipe	109
Fiddle	110
Shanai	111
TinkleBell	112
Agogo	113
SteelDrums	114
Woodblock	115
TaikoDrum	116
MelodicTom	117
SynthDrum	118
ReverseCymbal	119
GuitarFretNoise	120
BreathNoise	121
Seashore	122
BirdTweet	123
TelephoneRing	124
Helicopter	125
Applause	126
Gunshot	127

INSTRUMENTOS DO CANAL 10 ->Percussão e Bateria

Código 0 = Standard Set
Código 8 = Room Set
Código 16 = Power Set
Código 32 = Jazz Set
Código 48 = Orchestral Set
Código 40 = Brush Set
Código 24 = PElectronic Set
Código 56 = SFX Set
Código 25 = TR-808 Set
Código 16 = Power Set

ANEXO 3 -

Tabela de notas, Freqüência e Código Midi equivalente dentro da faixa de freqüências audíveis pelo ser humano

Notas musicais	Freqüências	Código MIDI
re#-1	19.4454 Hz	15
mi-1	20.6017 Hz	16
fa-1	21.8267 Hz	17
fa#-1	23.1246 Hz	18
sol1	24.4997 Hz	19
sol#-1	25.9565 Hz	20
la-1	27.50000 Hz	21
la#-1	29.1352 Hz	22
si-1	30.8677 Hz	23
do0	32,7032 Hz	24
do#0(reb0)	34,6478 Hz	25
re0	36,708 Hz	26
re#0(mib0)	38,8908 Hz	27
mi0	41,2034 Hz	28
fa0	43,6536 Hz	29
fa#0(solb0)	46,2494 Hz	30

sol0	48,9994 Hz	31
sol#0(lab0)	51,913 Hz	32
la0	55 Hz	33
la#0(sib0)	58,2704 Hz	34
si0	61,7354 Hz	35
do1	65,4064 Hz	36
do#1(reb1)	69,2956 Hz	37
re1	73,4162 Hz	38
re#1(mib1)	77,7818 Hz	39
mi1	82,4068 Hz	40
fa1	87,307 Hz	41
fa#1(solb1)	92,4986 Hz	42
sol1	97,9988 Hz	43
sol#1(lab1)	103,8262 Hz	44
la1	110 Hz	45
la#1(sib0)	116,541 Hz	46
si1	123,4708 Hz	47
do2	130,8128 Hz	48
do#2(reb2)	138,5914 Hz	49
re2	146,8324 Hz	50
re#2(mib2)	155,5634 Hz	51
mi2	164,8138 Hz	52
fa2	174,6142 Hz	53
fa#2(solb2)	184,9972 Hz	54
sol2	195,9978 Hz	55
sol#2(lab2)	207,6524 Hz	56
la2	220 Hz	57
la#2(sib0)	233,0818 Hz	58

si2	246,9416 Hz	59
do3 (dó central do piano)	**261,6256 Hz**	60
do#3(reb3)	277,1826 Hz	61
re3	293,6648 Hz	62
re#3(mib3)	311,127 Hz	63
mi3	329,6276 Hz	64
fa3	349,2282 Hz	65
fa#3(solb3)	369,9944 Hz	66
sol3	391,9954 Hz	67
sol#3(lab3)	415,3046 Hz	68
la3 (diapasão)	**440 Hz**	69
la#3(sib0)	466,1638 Hz	70
si3	493,8834 Hz	71
do4	523,2512 Hz	72
do#4(reb4)	554,3652 Hz	73
re4	587,3296 Hz	74
re#4(mib4)	622,254 Hz	75
mi4	659,2552 Hz	76
fa4	698,4564 Hz	77
fa#4(solb4)	739,9888 Hz	78
sol4	783,9908 Hz	79
sol#4(lab4)	830,6094 Hz	80
la4	880 Hz	81
la#4(sib0)	932,3276 Hz	82
si4	987,7666 Hz	83
do5	1046,5022 Hz	84
do#5(reb5)	1108,7306 Hz	85
re5	1174,659 Hz	86

re#5(mib5)	1244,508 Hz	87
mi5	1318,5102 Hz	88
fa5	1396,913 Hz	89
fa#5(solb5)	1479,9776 Hz	90
sol5	1567,9818 Hz	91
sol#5(lab5)	1661,2188 Hz	92
la5	1760 Hz	93
la#5(sib0)	1864,655 Hz	94
si5	1975,5332 Hz	95
do6	2093,0046 Hz	96
do#6(reb6)	2217,461 Hz	97
re6	2349,3182 Hz	98
re#6(mib6)	2489,0158 Hz	99
mi6	2637,0204 Hz	100
fa6	2793,8258 Hz	101
fa#6(solb6)	2959,9554 Hz	102
sol6	3135,9634 Hz	103
sol#6(lab6)	3322,4376 Hz	104
la6	3520 Hz	105
la#6(sib0)	3729,31 Hz	106
si6	3951,0664 Hz	107
do7	4186,009 Hz	108
do#7(reb7)	4434,922 Hz	109
re7	4698,6362 Hz	110
re#7(mib7)	4978,0318 Hz	111
mi7	41,2034 Hz	112
fa7	5587,6518 Hz	113
fa#7(solb7)	5919,9108 Hz	114

sol7	6271,927 Hz	115
sol#7(lab7)	6644,8752 Hz	116
la7	7040 Hz	117
la#7(sib0)	7458,6202 Hz	118
si7	7902,1328 Hz	119
do8	8372,018 Hz	120
do#8(reb8)	8869,8442 Hz	121
re8	9397,2726 Hz	122
re#8(mib8)	9956,0634 Hz	123
mi8	10548,0818 Hz	124
fa8	11175,3034 Hz	125
fa#8(solb8)	11839,8216 Hz	126
sol8	12543,854 Hz	127
sol#8(lab8)	13289,7504 Hz	
la8	14080 Hz	
la#8(sib0)	14917,2404 Hz	
si8	15804,2656 Hz	
do9	16744,0362 Hz	
do#9(reb9)	17739,6884 Hz	
re9	18794,5452 Hz	
re#9(mib9)	19912,127 Hz	
mi9	21096,1636 Hz	
fa9	22350,6068 Hz	

ANEXO 4 -

Linguagem funcional CLEAN

Este capítulo visa justificar o paradigma escolhido para o desenvolvimento e implementação deste trabalho de dissertação, fornecedor de um subsidio conceitual e prático da utilização da linguagem de programação escolhida, CLEAN, de forma que este texto atenda não somente a leitores com bases sólidas em programação, mas, também, aos músicos que posteriormente venham buscar neste trabalho uma fonte de informações que os permitam desenvolver aplicativos para manipulação e análise de ações no domínio musical.

Alguns tópicos são apresentados com mais detalhes devido à carência de documentação sobre a linguagem CLEAN, por ser a mesma ainda relativamente nova.

A escolha do paradigma e da linguagem de programação

A escolha correta do melhor paradigma e respectiva linguagem de programação para se implementar uma classe de problemas é fundamental para se atingir os resultados esperados, bem como para facilitar, tornar mais aderente, a modelagem do problema e tratamento dos erros que forem surgindo durante o projeto.

Conforme citado na introdução, a matemática e a música sempre andaram juntas destes os primórdios da humanidade. Isto se deu e se dá devido à música possuir técnicas e raciocínio extremamente matemáticos e lógicos, inclusive pela mesma trabalhar com um sistema de grafia (notação musical) em base binária.

Desta forma, a escolha de um paradigma que trabalhe bem com funções matemáticas e que facilite implementações afins, é o ideal para se trabalhar e gerar ferramentas também no domínio musical.

Além de se escolher um paradigma adequado, deve-se também escolher uma linguagem que seja a mais aderente possível aos fenômenos que se deseja modelar.

A escolha natural, feitas tais análises, se dá pelo paradigma funcional, o qual possui várias vantagens em relação ao paradigma procedimental, descritas com detalhes e simplicidade por RUFINO e LIMA [29].

Até uma década atrás, a linguagem funcional LISP [30] foi a linguagem mais utilizada em computação musical, principalmente na construção de sistemas especialistas e sistemas inteligentes. Atualmente existem linguagens funcionais mais poderosas e mais fiéis ao paradigma matemático, podendo-se citar, entre elas: Haskell e Clean.

Dentre estas duas linguagens, fortemente tipadas, sem problemas de transparência referencial, avaliação destrutiva e efeitos colaterais, optou-se por utilizar a linguagem Clean [31] por possuir, também, o atrativo de se poder implementar interfaces gráficas de uma forma simples e por rodar em diversos sistemas operacionais sem que seja necessário a instalação de dlls.

A seguir, apresenta-se alguns conceitos relevantes para que se possa entender a escolha do referido paradigma, bem como princípios básicos de utilização da linguagem de programação CLEAN, suas potencialidades e aderência ao paradigma matemático.

Por que mais uma linguagem?

Fica sempre a seguinte pergunta no ar:

- Com uma linguagem apenas um bom programador não seria capaz de resolver todos os problemas de modelamento e implementação computacionais?

A resposta é: praticamente sim!

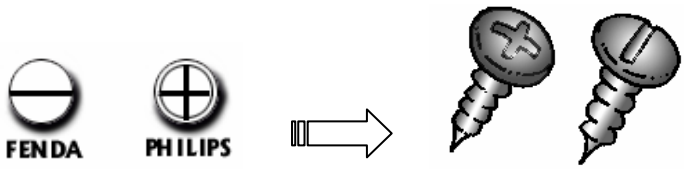
Uma pergunta conseqüente seria:

- Para que então tantos tipos de linguagens?

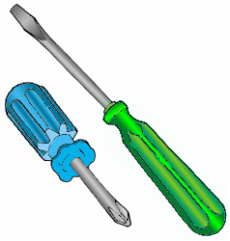
Pode-se responder esta pergunta utilizando uma analogia muito simples e aderente ao conceito. Já que as linguagens de programação, tais como CLEAN, são ferramentas computacionais, associe-as ao conceito e à utilização das ferramentas que normalmente são utilizadas no dia a dia.
Suponha que tenha que realizar uma tarefa bem simples:

Apertar um parafuso

Suponha que possua um parafuso do tipo Philips, ou seja, em vez de ter um chanfro em linha reta em sua cabeça ele tem um chanfro em X (ou em cruz se preferir).



Ao utilizar tal parafuso, pode-se tentar apertá-lo com uma chave de fenda comum ou pode-se fazer a mesma tarefa com uma chave própria (a chave Philips).



A chave Philips foi desenvolvida para apertar parafusos Philips. Assim, a tarefa será mais simples, mais rápida e adequada aos propósitos propostos quando utiliza-se a ferramenta certa (o parafuso Philips permite um aperto melhor).

Aproveite este exemplo e se questione:

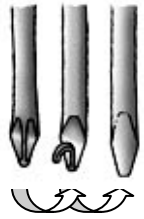
- **Será possível apertar um parafuso Philips utilizando uma chave de fenda comum?**

A resposta é sim, apesar de não conseguir obter um aperto tão firme quanto o que seria efetuado com a chave Philips.

- **E se for apertar um parafuso comum, tipo fenda, será que conseguiria apertá-lo com uma chave tipo Philips?**

A resposta a princípio seria não.

Diz-se a princípio, devido ao fato de que, se fizer algumas modificações na chave Philips, tal como limar a cabeça da mesma até que ela fique chata, tal como uma chave de fenda comum, seria possível realizar a tarefa proposta.



O que ocorre é que tal modificação radical descaracterizará por completa a ferramenta utilizada, chegando a tal ponto de não permitir que a mesma realize adequadamente as tarefas que anteriormente fazia, ou seja, apertar parafusos Philips com mais força, mais torque.

Ao mesmo tempo, a modificação desta ferramenta demandará do usuário uma destreza e um conhecimento especializado que o permita modificar a ferramenta adequadamente.

Assim, apertar um parafuso comum fica mais fácil e adequado utilizando uma chave de fenda comum, sendo uma tarefa difícil apertá-lo com uma chave de fenda Philips.

Desta forma, para cada tarefa do dia a dia tem-se vários tamanhos e tipos de chaves de fenda, alicates, etc. Cada ferramenta atendendo tipos de tarefas específicas, **paradigmas** diferentes.

O mesmo acontece com as linguagens de programação. Existem tarefas computacionais que são mais facilmente resolvidas se forem utilizadas as ferramentas computacionais adequadas, ou seja, com **paradigmas aderentes** à aplicação.

Paradigma

Cada tarefa pode ser implementada e solucionada de várias formas diferentes. A forma com que se soluciona um determinado problema é chamada de **paradigma**.
Como exemplo, antes de Santos Dumont, os relógios eram sempre carregados no bolso e presos a uma correntinha. Esta solução parecia perfeita, ou seja, atendia todas as expectativas dos usuários.

- Então, para que modificá-la ?

Esta pergunta nem chegou a ser questionada, já que era perfeita. Eis que um belo dia, Santos Dumont resolveu inventar sua máquina voadora, a qual exigia que suas mãos estivessem firmemente fixas aos controles delas, já que se as retirassem poderiam ocorrer erros que possivelmente acarretariam a queda das mesmas. Como poderia, portanto ele, o piloto, olhar as horas para, por exemplo, verificar o tempo de combustível, retirando o relógio do bolso, já que o painel ainda não havia sido inventado (um novo paradigma)?

Neste momento surgiu a necessidade de se modificar o paradigma de como utilizar e carregar um relógio. Assim, Santos Dumont implementou uma nova solução, um novo paradigma, que deu origem aos relógios de pulso. Este é um exemplo clássico de mudança de paradigma. A mudança dos relógios movidos à corda e os movidos à pilha (bateria) é um outro exemplo de quebra de paradigma. É totalmente diferente trabalhar, consertar ou apenas utilizar um relógio com corda e um relógio com pilha (bateria). O surgimento dos relógios digitais também foi uma nova mudança de paradigma, a princípio desacreditada pelos Suíços, mas popularizada com sucesso pelos japoneses.

A chave Philips e a chave Allen são, também, exemplos de mudanças de paradigma.

As linguagens de programação também possuem paradigmas de programação diferentes. Assim, pode-se citar quatro paradigmas muito conhecidos, a saber:

1- O paradigma imperativo, procedural ou procedimental.
2- O paradigma lógico
3- O paradigma funcional
4- O paradigma de Orientação a Objeto

As linguagens desenvolvidas em cada um destes paradigmas poderão realizar basicamente as mesmas tarefas, só que de formas totalmente diversas. Assim, algumas tarefas, alguns aplicativos, serão mais facilmente implementadas em uma linguagem do que em outra.

Como escolher o paradigma mais adequado a um determinado projeto?

A resposta é simples: - escolha o paradigma mais **aderente** à sua aplicação. Para isto, deve-se conhecer os conceitos, particularidades, potencialidades e restrições de cada um.

Aderência

Um outro ponto relevante, que deve ser destacado, é que nem sempre um paradigma que possua todas as ferramentas necessárias à efetivação da tarefa é aderente ao usuário.

Entende-se por aderência a forma com que se apresenta uma determinada tarefa ou ferramenta ao programador ou a um usuário final do produto.

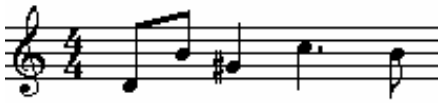
Observe o seguinte questionamento e exemplo:

- Suponha que seja desejado apresentar o resultado de um programa musical a um músico. Qual seria a melhor maneira de apresentar tais resultados?

Pode-se refazer esta pergunta da seguinte forma:

- Qual seria a forma mais aderente de apresentar uma música gerada por um programa de computador, por um aplicativo, para um músico?
- Por outro lado, qual seria a forma mais aderente para apresentar o resultado a uma pessoa que não domine a sintaxe musical?

A resposta é que para um músico a forma de apresentação mais aderente de uma música ao mesmo seria uma partitura convencional do tipo:



Se um profissional não for um músico, fatalmente não entenderá nada do que este gráfico, este desenho da partitura, está querendo informar. Muito menos poderá aquilatar se existem ou não erros nesta apresentação. Neste caso, pode-se dizer que a partitura musical não é uma forma aderente de apresentar os resultados do programa para uma pessoa que não seja músico.

Desta forma deve-se criar um outro tipo de notação que possa apresentar os resultados de um programa musical de forma mais aderente para um usuário não músico. Por exemplo:

Compasso = [(Ré5,c), (Si5,c), (G#5,Sm), (Dó6, Smp),(Si5, c)]

Um usuário com conhecimentos de computação ou matemática percebe rapidamente que a informação é uma lista com 5 tuplas de dois elementos cada, cujo nome é Compasso. Se for informado ao mesmo que o primeiro elemento de cada tupla é uma nota musical e que o segundo é o tempo, a duração desta nota, a informação será facilmente absorvida por ele. Por outro lado, se tentássemos explicar, ensinar ao mesmo como ler a simbologia da notação musical, esta seria uma tarefa bem mais complicada e complexa.

Assim, aderência é tornar sua ferramenta ou os resultados de seu programa fáceis de serem lidos ou utilizados pelo usuário alvo de seu sistema.

Aderência não tem nada a ver com o grau de complexidade ou de recursos de um sistema e sim, com a forma mais simples de apresentar o que se queira a um determinado usuário, de forma que o mesmo não tenha a mínima dificuldade em utilizar os conhecimentos e ferramentas de seu sistema.

Para melhor fixar o que foi afirmado, veja como uma linguagem com paradigma funcional, no caso CLEAN, é aderente ao paradigma matemático:

Para tanto, observe a definição de um número de Fibonacci:

Fibonacci (n) = n, se n < = 1
Fibonacci (n) = Fibonacci (n-2) + Fibonacci (n-1), se n >1

A implementação dessa função em paradigma funcional, linguagem CLEAN, é apresentada a seguir:

```
Fibonacci  n |  n <= 1 =  n
Fibonacci  n |  n > 1   =  Fibonacci (n-2) + Fibonacci (n-1)
```

Como se pode verificar, o programa escrito em CLEAN é praticamente a própria definição matemática do problema proposto, ou seja, CLEAN é uma linguagem em paradigma funcional, extremamente aderente ao paradigma matemático.

Se fosse utilizado o paradigma procedural ou imperativo, Linguagem C, o mesmo programa ficaria da seguinte forma:

Sequência de Fibonacci escrita em C.

```
#include <stdio.h>
#include <stdlib.h>

int fib(int n){
     int x,lofib,hifib,i;

if (n <= 1) return(n);
  lofib = 0;
  hifib = 1;
  for (i = 2;i <= n; i++){
     x = lofib;
     lofib = hifib;
     hifib = x + lofib;}
return (hifib);}

void main(){
int fibo;
char string[25];

fibo = fib(45);
itoa(fibo, string, 15);
printf("fib(45)=%d \n",fibo,string);
return;}
```

Pode-se perceber que a implementação procedural da série de Fibonacci muito se difere da definição matemática da mesma. Se você não for um bom programador em Linguagem **C**, fatalmente não entenderia o que tal programa faz. Neste caso, diz-se que a linguagem procedural, no caso **C**, não é aderente ao paradigma matemático. Isto não significa, entretanto, que não se pode implementar funções matemáticas em **C**, nem que tal paradigma não seja eficiente nestes casos. O que se pode afirmar é que o mesmo não é aderente, ou seja, não possui a clareza desejada na implementação e leitura de programas que realizem tarefas do paradigma matemático.

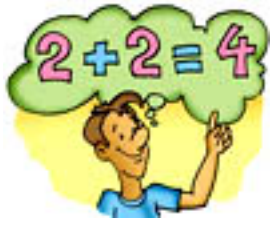
Erro de tipo

Algumas linguagens podem acarretar efeitos colaterais indesejáveis e imprevisíveis nos programas implementados. As linguagens procedurais são aquelas que mais efeitos colaterais produzem. CLEAN foi criado de tal forma a procurar evitá-los.

A definição dos tipos de dados numa linguagem de programação é algo importante, os mesmos evitam em parte o que denominamos de efeitos colaterais, os quais são comuns em sistemas gerados por linguagens com paradigma procedural ou imperativo. Para que se entenda melhor o conceito e a necessidade da tipagem de dados, veja a seguinte analogia, dada por alguns questionamentos:

-Quanto você acha que é dois mais dois?

A resposta é simples, ou seja: quatro.



-E quanto é um tijolo mais dois tijolos?

A resposta também é simples: três tijolos.

-E quanto é uma banana mais um abacaxi?

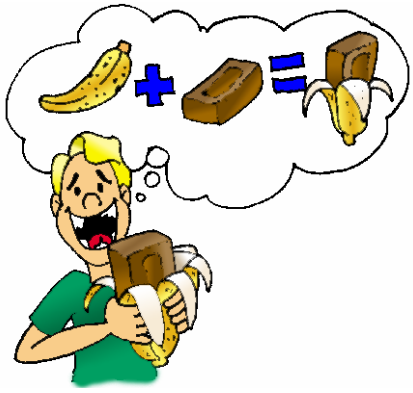
Agora o problema ficou um pouco mais complicado de ser solucionado, ou seja:

Como somar banana com abacaxi?

Algumas pessoas poderiam tentar uma resposta dizendo que o resultado é uma salada de frutas. Mesmo se fosse, onde esta resposta seria armazenada, já que não se tem inicialmente lugar para guardar o tipo salada de frutas? Apenas se possui um lugar para guardar banana e outro para abacaxi. Assim, o que seria mais correto: guardar a salada de frutas no local das bananas ou no dos abacaxis?

Em qualquer um dos lugares que for armazenado o resultado, ele não teria nenhum significado, estaria fora do contexto.

-Bom, e quanto seria uma banana mais um tijolo?



Agora realmente o problema perde todo o significado, nem a salada de frutas serve como resposta.

Neste momento, sucinta o seguinte questionamento:

- O que somar tijolo com banana tem a ver com linguagens tipadas?

Serve para alertar que não se pode somar tijolo com banana. O correto, quando for solicitado tal absurdo, seria informar à pessoa que pediu que se somasse tijolo com banana que a mesma cometeu um erro de tipo e que infelizmente não se pode efetuar a operação desejada sem causar problemas imprevisíveis em futuras operações.

Misturar bananas com tijolos é semelhante, em computação a se misturar inteiros com reais, operar tipos de dados diferentes.

Infelizmente, as escolas não se preocupam em manipular corretamente estes tipos de dados, e, assim, comete-se o absurdo de realizar a seguinte soma:

2 + 3,5 = 5,5

Isto pode parecer, a princípio, estar correto. O que ocorre é que esta é uma operação envolvendo elementos de dois conjuntos distintos, onde, no conjunto dos inteiros não existem elementos reais e vice versa. Realizando tal soma estar-se-ia ferindo no mínimo a propriedade de fechamento dos conjuntos, ou seja, em um conjunto, ao somarmos dois de seus elementos sempre teremos como resultado um elemento deste conjunto.

Na computação isto se agrava. Primeiro porque os conjuntos não são infinitos, existe uma quantidade máxima de elementos que se pode armazenar na memória de um computador. Por outro lado, a forma de se armazenar os inteiros é completamente diferente da forma de se armazenar os reais. Desta forma, por absurdo, se fosse possível somar inteiros com reais, como o resultado seria armazenado: como real ou como inteiro?

A linguagem Funcional CLEAN, assim como as linguagens tipadas modernas, não permite que, mesmo por distração, o programador opere com tipos de dados diferentes.

Observando a soma **2 + 3,5**, fica o questionamento: o que será que a pessoa que propôs esta conta queria fazer? E se a mesma tivesse digitado por engano a vírgula e o resultado fosse a quantidade de adrenalina a ser injetada em um paciente com parada cardíaca?

Além de não permitir tais erros de tipo, CLEAN [32] é completamente aderente ao paradigma matemático. Desta forma, o mesmo não permite absurdos cometidos por linguagens procedurais, tal como a avaliação destrutiva de variáveis.

Ex: **x = x + 1**

Entender como tais erros ocorrem exigiria do leitor um conhecimento mais profundo de programação, o que não é o interesse no momento, mas é importante saber que os mesmos existem e tentar evitar que eles ocorram quando utilizarmos linguagens ou aplicativos que utilizam o paradigma procedural, ou, uma solução melhor, escolher uma linguagem, um paradigma, que por si só não permita que tais fatos ocorram, assim como o funcional.

A Linguagem CLEAN

Um dos pontos que torna a Linguagem Funcional CLEAN atrativa a desenvolvimentos no domínio musical está na facilidade de se implementar qualquer sistema computacional com a mesma. CLEAN possui uma ferramenta matemática que permite que se defina o conjunto imagem, o conjunto de todas as possíveis soluções para um problema, de uma forma simples, formal e que apenas exige do programador ter o conhecimento do domínio de onde pertencem seus dados e das regras que devem ser aplicadas aos mesmos para gerar o conjunto solução (Imagem). Esta ferramenta matemática é denominada por Notação Zermelo-Frankel [29] (Compreensão de listas e vetores).

Tipos de dados:

• Caractere

- Um **caractere** é um símbolo de um alfabeto.
- O alfabeto utilizado pelo computador é o ASCII:

  ASCII ->American **S**tandard **C**ode for **I**nformation **I**nterchange [**ANEXO 1**]
- No CLEAN, um caractere é representado por um símbolo colocado entre apóstrofos.

  Exemplos: ‘a’= caractere letra **a** minúscula, ‘1’ = caractere dígito **1**.
- Não se pode confundir o **caractere ‘2’** com o **inteiro 2**, um é um símbolo e o outro um valor a ser manipulado, e assim por diante.

- Na tabela com este alfabeto, os caracteres que representam os dígitos iniciam com o código 48. As letras maiúsculas começam com o código.

  Assim:

  - Dígitos => ‘**0’ = 48** , ‘**1’ = 49,** ...’**9’ = 57**.
  - Letras Maiúsculas => **‘A’ = 65, ‘B’ = 66**, ...’**Z’ = 90.**
  - Letras Minúsculas => **‘a’ = 97**, **‘b’ = 98**, ... **‘z ‘ = 122.**

- O tipo de dado CARACTERE é referenciado no CLEAN como: **Char**

• String

- Uma String é uma cadeia de caracteres colocados entre aspas. Exemplo: **“Casa”**, **“casa”**, **“1234”**. Onde **“Casa”** e **“casa”** são Strings diferentes devido a ambas começarem com caracteres ASCII diferentes. O caractere **‘c’** possui um código igual a **99** e o caractere **‘C’** igual a **67** [ANEXO 1].
- Uma String é uma cadeia (vetor) de caracteres que devem ser “lidos” de uma só vez, e não caractere por caractere.

- O tipo de uma String é : **{ {#Char} }**
- Um vetor de caracteres são elementos de mesmo tipo de dados entre chaves e separados por vírgula. Exemplo: **{1,2,3} = vetor de inteiros** ou **{‘a’,’r’}= vetor de caracteres**.
- Uma String é um vetor de caracteres que só possui sentido se lido de uma vez só.
- Em CLEAN, a representação de uma cadeia de caracteres pode ser feita de duas formas:
    1. **Estrita = String = {#Char}**
    2. **Normal = {Char}**

Veja a diferença:

Estrita (String):

Start :: {#Char}

Start = {'a','g','o','r','a'}

Devolve: “agora”

Normal:

Start :: {Char}

Start = {'a','g','o','r','a'}

Devolve: {'a','g','o','r','a'}

Ou seja: Ao declararmos o tipo de dados de saída da função **Start** como String (um vetor de caracteres estrito), ao executarmos o vetor de caracteres, o CLEAN devolveu uma String. No caso de declararmos como um vetor de caracteres, o mesmo devolveu um vetor de caracteres. Pode parecer a princípio ser a mesma coisa, mas, na realidade, são “entidades” diferentes.

Obs. Você pode acrescentar caracteres em uma string, mas, em uma cadeia (vetor) de caracteres não estritos não.

Exemplo:

Obs. +++ é o operador de concatenação de duas cadeias estritas de caracteres (**Strings**). O mesmo não se aplica a uma cadeia (vetor) de caracteres não estrita.

1- operação com duas cadeias de caracteres não estritas.

funde ::{Char} {Char} -> {Char}

funde x y = x +++ y

Start = funde {'a','g','o','r','a'} {' ','s','o','u','.'}

Devolve: Uma mensagem de erro dizendo não ser possível realizar a operação.

2- operação com duas cadeias de caracteres estritas (String).

funde ::{#Char} {#Char} -> {#Char}

funde x y = x +++ y

Start = funde {'a','g','o','r','a'} {' ','s','o','u','.'}

Devolve: “agora sou.”

ou

funde x y = x +++ y

Start = funde "agora" " sou."

Devolve: “agora sou.”

Obs. Isto mostra que os dois tipos de dados {#Char} e {Char} são diferentes, apesar de representarem uma cadeia de caracteres.

Para normalizar a nomenclatura, diz-se que:

- Uma cadeia de caracteres estrita (String) é um vetor de caracteres eager.
- Uma cadeia de caracteres não estrita é um vetor de caracteres laze.

Assim:

- Em um vetor eager, todos os elementos do mesmo são acessados, lidos, de uma vez só.
- Em um vetor laze, cada elemento do mesmo pode ser acessado, lido, separadamente.

• Vetores

- Um vetor possui elementos de mesmo tipo de dados delimitados por chaves e separados por vírgula. Exemplo: **{1,2,3} = vetor de inteiros** ou **{‘a’,’r’}= vetor de caracteres**.
- A declaração de um vetor é feita colocando-se o tipo do dado entre chaves. Por exemplo: **{1,2,3}** é do tipo: **{Int}** , **{ “casa”, “porta”}** é do tipo: **{ {#Char} }, {2.4}** é do tipo **{Real}**, **{ {1,2}, {4,5,6}, {3} }** é do tipo: **{ {Int} }**.
- Já foi visto anteriormente um caso particular de vetores: a cadeia de caracteres estrita: **String**.
- Um vetor possui número fixo de elementos, com a exceção da String.

• Funções de Manipulação de Vetores

- Pegando um elemento de um vetor

Praticamente, para os propósitos iniciais, esta é a função que mais será utilizada, e, portanto, somente ela será apresentada.

Para pegar um elemento do vetor, basta acrescentar no fim do mesmo o sinal de ponto seguido de um valor inteiro entre colchetes. Este valor fará com que o elemento do vetor cujo índice for o inteiro colocado entre colchetes seja retornado.

Exemplo:

Start = {‘a’,’r’,‘o’}.[2]

Devolve: ‘o’, o qual é o caractere de índice 2. Lembre-se que o primeiro elemento possui índice 0.

Start = “aro”.[2]

Devolve: ‘o’, o que já era de se esperar, já que uma String é um vetor.

- # Matrizes

- Uma matriz é um vetor de vetores com elementos de mesmo tipo de dados.
- Cada elemento de uma matriz é um vetor.
- Uma matriz possui vetores como elementos, separados por vírgula e delimitados por uma chave. Exemplo: **{{1, 34}, {22, 33}, {4}}**. O tipo de dado desta matriz exemplo, é declarado da seguinte forma: **{{Int}}**

- ## Funções de Manipulação de Matrizes

Novamente só vamos abordar neste documento como pegar um elemento de uma matriz

- Pegando um elemento de uma matriz

O procedimento é semelhante ao utilizado para vetores, com a diferença que entre colchetes ficarão dois inteiros: o primeiro deles indica qual é o vetor da matriz que está sendo acessado, e, o segundo valor inteiro indica qual elemento deste vetor será devolvido.

Exemplo:

matriz :: {{Int}}

matriz = { {1,2},{3},{5,6,7} }

Start = matriz.[2,0]

Devolve: 5, ou seja, o primeiro elemento do vetor de índice 2 (o terceiro vetor)

Observe que para manipular uma matriz, deve-se declarar o tipo dela. Para tanto, teve-se de criá-la antes de manipulá-la.

- # Listas

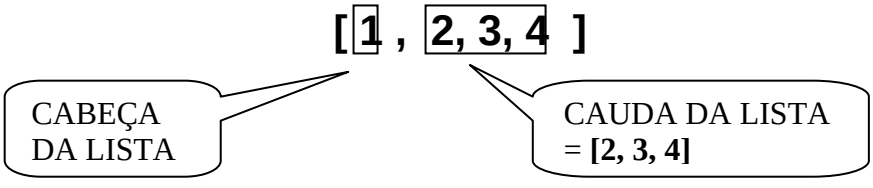
- Uma lista possui elementos de mesmo tipo de dados delimitados por colchetes e separados por vírgula. Exemplo: **[1,2,3] = lista de inteiros** ou **[‘a’, ‘r’] = lista de caracteres**.
- A declaração de tipo de uma lista é dada pelo tipo de seus elementos colocado entre colchetes. Exemplo:
  1. a lista **[1,2,3]** possui o tipo **[Int]**,
  2. a lista **[‘a’, ‘r’]** possui o tipo **[Char]**,
  3. a lista **[“casa”, “eu”]** possui o tipo **[{#Char}]**,
  4. a lista **[[1,2],[3],[44,5]]** possui o tipo **[[Int]].**
- Uma lista possui um número variado de elementos, ou seja, você pode eliminar um elemento de uma lista ou mesmo acrescentar.

- Uma Lista em CLEAN possui a seguinte estrutura: **[c:r],** onde **c** é a cabeça da lista (o primeiro elemento dela) e **r** o resto da lista (sua calda, ou seja, a lista sem a cabeça).

- **Lista vazia**->Uma lista vazia é representada por **[ ]**, ou seja, o abrir e fechar de colchetes com tantos espaços em branco internamente quanto se deseje.

- **Exemplos de listas finitas:**

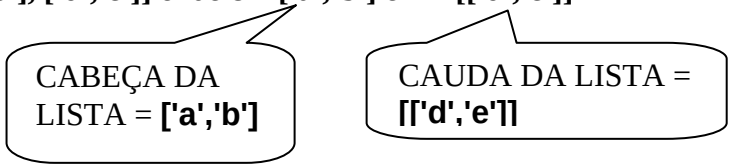
- lista de inteiros: **[1, 2, 3, 4]**



Comparando a lista **[1, 2,3,4]** com **[c:r]** temos que: **c = 1** e **r = [2,3,4]**

- lista de reais: **[1.0, 2.0, 3.0, 4.0]** onde **c = 1.0** e **r = [2.0, 3.0, 4.0]**
- lista de strings: **["a","b","c"]** onde **c = "a"** e **r = ["b","c"]**
- lista de caracteres: **['a','b','c']** onde **c = 'a'** e **r = ['b','c']**

- lista de listas: **[['a','b'], ['d','e']]** onde **c =['a','b']** e **r = [['d','e']]**



- **Tuplas**

Uma tupla é uma coleção de elementos separados por vírgula. Tal coleção é delimitada por parênteses. Alguns exemplos de tuplas incluem:

- (‘a`,1 ,“casa”) – uma tupla formada por elementos de três diferentes tipos de dados.
- ([1, 2, 3], [4, 5, 6]) – uma tupla formada por duas listas.

Algumas vezes você poderá sentir necessidade de agrupar informações de diferentes tipos. Por exemplo, podemos pensar em abstrair uma pessoa pelo seu nome e sua identificação. O nome pode ser representado por um valor do tipo String enquanto que a identificação pode ser representada por um inteiro. A representação de uma pessoa por uma lista não seria possível pois a lista deveria ter dois elementos de tipos diferentes (nome e identificação) e isto violaria a propriedade de que todos os elementos da lista devem ter um mesmo tipo. O uso de uma tupla formada por dois elementos de tipos diferentes (nome e identificação) pode ser usado neste caso.

As tuplas, como mostram os exemplos descritos anteriormente, podem ter elementos que apresentam diferentes tipos de dados. Podemos ter tuplas formadas por listas assim como listas formadas por tuplas. Os exemplos que seguem mostram listas cujos elementos correspondem a tuplas.

- **Exemplos de listas de tuplas:**

- **Lista de tuplas de dois elementos inteiros:**

[(1, 2), (3, 4), (5, 6)]

Neste caso se **[c:r]** é a lista **[(1, 2), (3, 4), (5, 6)]**, então:

c = (1, 2) e **r = [(3, 4), (5, 6)]**

- **Lista de tuplas de três elementos inteiros:**

  [(1, 2, 3), (4, 5, 6)]

  Neste caso se **[c:r]** é a lista **[(1, 2, 3), (4, 5, 6)]**, então:

  c = (1, 2, 3) e **r = [(4, 5, 6)]**

- **Lista de tuplas de tuplas de três elementos inteiros:**

  [((1,2,3),(4,5,6)),((7,8,9),(10,11,12))]

  Neste caso se **[c:r]** é a lista **[((1,2,3),(4,5,6)),((7,8,9),(10,11,12))]** então:

  c = ((1,2,3),(4,5,6)) e **r = [((7,8,9),(10,11,12))]**

A linguagem CLEAN possui algumas funções básicas que permitem a manipulação deste tipo de dado. Se estas não forem suficientes você poderá criar as suas próprias funções.

• Declaração de tipo

Quando for declarar o tipo de uma função cujos argumentos utilizem o tipo tupla, proceda da seguinte forma:

Exemplo 1: tuplas de inteiros

A seguir tem-se a definição do tipo de uma função ***f*** cujo argumento de entrada é uma tupla de dois elementos inteiros e devolve como resposta uma tupla de dois elementos inteiros.

```
f :: (Int, Int) -> (Int, Int)
```

Exemplo 2: tuplas de inteiros e reais

A função ***f*** cujo tipo é definido a seguir apresenta como argumento de entrada uma tupla de dois elementos, sendo um inteiro e o outro um real. O valor resultante do cálculo desta função é um valor do tipo **Int**.

```
f :: (Int, Real) -> Int
```

Exemplo 3: tuplas de inteiros, reais e caracteres

Neste exemplo tem-se a definição do tipo de uma função ***f*** cujo argumento de entrada é uma tupla de três elementos, sendo um inteiro, o outro um real e o último um caractere. A saída é uma lista de inteiros.

```
f :: (Int, Real, Char) -> [Int]
```

Pelos exemplos descritos observa-se que os tipos das tuplas são especificados pela enumeração dos tipos dos elementos que a constituem delimitados por parênteses. Assim temos que:

- (“Carlos”, 2312) :: (String, Int) (leia como: a tupla (“Carlos”, 2312) é de tipo (String, Int)) .
- (‘a’, False, 2) :: (Char, Bool, Int) (leia como: a tupla (‘a’, False, 2) é de tipo (Char, Bool, Int)).

• Notação Zermelo-Frankel

Este trabalho de dissertação trabalha com conjuntos de notas musicais e regras de pertinências das mesmas a estes conjuntos. A este conjunto de notas são aplicadas várias regras de inferência para formar os conjuntos soluções para que se possa transpor uma música para tonalidades, as quais possuem suas regras específicas. Este tipo de problema é simples de ser modelado e implementado utilizando a notação Zermelo-Frankel, disponibilizada na linguagem CLEAN com completa aderência e fidelidade matemática. Desta forma, é interessante que sejam aqui apresentados os detalhes e conceitos de implementação e utilização desta poderosa ferramenta matemática.

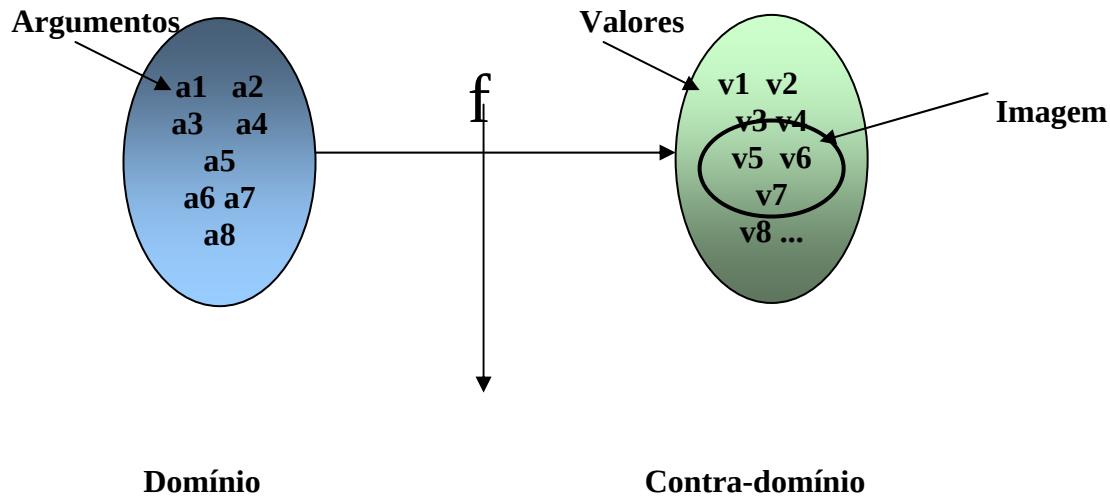
Esta notação, proposta por Ernest Zermelo, um matemático alemão, e por Adolf Frankel, um lógico Israelita, é um recurso, uma ferramenta matemática poderosa, veloz e eficiente na arte de computar valores de uma função [29]. A notação Zermelo-Frankel

formaliza, explicita a lei de formação de um conjunto imagem, dado o domínio da função e as regras de inferência da mesma. Para entendê-la melhor, deve-se ater a alguns conceitos fundamentais.

Computando o valor de uma função

No começo do século, matemáticos e filósofos se preocupavam com métodos que, a partir do argumento, pudessem calcular o valor de uma função. Assim, dado um elemento do domínio, eles buscavam descobrir, desenvolver um processo mecânico para se avaliar, encontrar o correspondente elemento do contradomínio. Este processo mecânico deveria ser constituído por uma sucessão de aplicações de regras de inferências, as quais determinariam o valor da função.

Os filósofos da matemática deram, a este processo, o nome de **computação do valor da função**. Foi Alonzo Church (Church, 1941) quem propôs o primeiro sistema formal de computação.
Observe a figura, a seguir:



- Onde: argumento é um elemento do domínio e valor um elemento do conjunto imagem.

Conceitos Fundamentais

1. Computar o valor de uma função é aplicar a função e suas regras de inferência a um elemento do domínio (denominado de argumento da função). Esta aplicação determina, infere, qual elemento do contradomínio é o valor (elemento do contradomínio) resultante desta aplicação.

2. Os elementos de um domínio devem possuir a mesma lei de formação, o mesmo tipo de dado.

3. O conjunto imagem é o conjunto de todos os valores computados por uma determinada função quando a mesma é aplicada a todos os elementos (argumentos) do domínio.

4. Os valores do conjunto imagem e do contradomínio devem possuir a mesma lei de formação, o mesmo tipo de dado.

5. O conjunto imagem, portanto, é formado pelos valores do contradomínio identificados pela aplicação da(s) regra(s) de inferência da função em todos os argumentos do domínio.

6. Assim, o conjunto imagem possuirá menos elementos que o contradomínio, ou, no máximo, a mesma quantidade. O conjunto imagem só terá menos elementos que o domínio se existir restrições aos elementos do domínio (veremos a seguir como isto é feito).

Declarando o tipo de uma função (domínio e imagem)

É fundamental para tornar um programa mais legível e modular que se possa explicitar a declaração do tipo de dado de cada função. Matematicamente isto é similar a se declarar o domínio e a imagem da mesma.

Em CLEAN, fazer isto é uma tarefa simples, bastando ao programador digitar o **nome_da_função** seguido dos sinais **::**, feito isto, após tais sinais deve-se colocar o tipo de dado de cada argumento da função (domínio) seguido dos sinais -> seguido do tipo de dado do valor (resultado) da função (imagem).

Nome_da_função::tipo_de_dados_do_domínio->tipo_de_dados_do_contraDomínio

Observações Gerais da Notação

Zermelo, e depois finalizado por Frankel, definiram, formalizaram o conjunto imagem da seguinte forma:

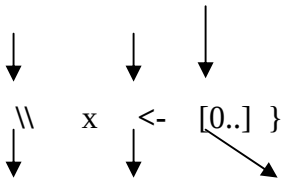
Domínio: {x ∈ **N**}

Função : f(x) = x^3

Notação Matemática da formalização do conjunto imagem:

S = { x^3 | x ∈ **N** }

Implementação em CLEAN:



tal que pertence (naturais = intervalo de 0 a infinito)

Obs. O símbolo ^ é o operador de potenciação utilizado pelo CLEAN.

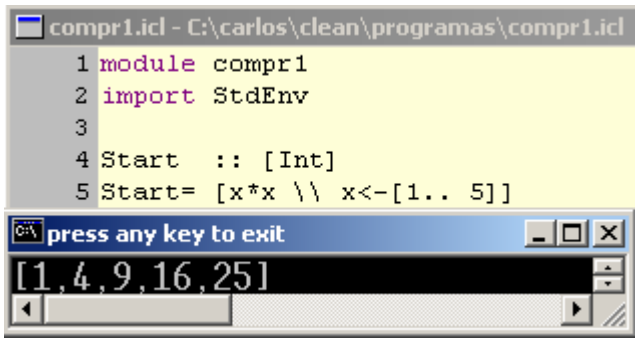
Para exemplificar esta notação, observe o seguinte exemplo:

- Mostrar o conjunto dos quadrados dos inteiros entre 1 e 5.

Da matemática tem-se : S = { x^2 | 1 ≤ x ≤ 5 }

Em CLEAN tem-se : S = [x^x \\ x <- [1..5]]

A implementação no CLEAN é mostrada na figura a seguir.



Obs. :: [Int] declara o tipo do dado devolvido pela função Start, ou seja: uma **lista de inteiros**.

Generalizando a Notação Zermelo-Frankel

Em CLEAN, a notação pode ser implementada:

1- Para listas: **[NOTAÇÃO]**

2- Para vetores: **{ NOTAÇÃO }**

A única diferença é que para vetores a notação se delimita com chaves e para listas com colchetes.

- Exemplificando para listas, a generalização fica:

[função ou regra da função \\ domínio e suas restrições]

IMAGEM

- Exemplificando para vetores, a generalização fica:

{ função ou regra da função \\ domínio e suas restrições }

IMAGEM

Utilizando a Notação Zermelo-Frankel

Para tornar mais clara a implementação desta notação, segue-se um exemplo comentado:

Exemplo:

Implementar a Notação Zermelo-Frankel em CLEAN de tal forma que a mesma gere o conjunto imagem formado pela lista de todos os números naturais pares de 0 a 9.

Domínio: $\{x \in \mathbf{N} \mid 0 \leq x \leq 9 \wedge x \text{ é par}\}$

ou seja-> **{2,4,6,8}**

Função: f(x) = x

ou regra -> o valor de x será igual ao do argumento, sem qualquer modificação.

A pergunta que deveria ser suscitada agora é:

- Como montar corretamente a notação para que tal lista seja gerada?

Observe como fazer isto passo a passo.

1- Primeiro, o que se deseja é uma lista, assim, inicia-se colocando os colchetes que delimitam uma lista:

[ ]

2- O próximo passo é colocar o separador entre a função e o domínio utilizados por tal notação:

[\\ ]

Função ⇦ ⇨ **Domínio e restrições**

3- Quando a função possuir apenas uma regra, coloca-se a regra do lado esquerdo do separador \\.

- Se a mesma possuir mais de uma regra, coloca-se do lado esquerdo do separador a função com seu(s) argumento(s).

- No caso, a função é **f(x)=x**, possuindo apenas uma regra: **x**, ou seja, o valor inferido é igual ao argumento da função.
- Neste caso, coloca-se uma letra ou palavra qualquer como regra, desde que a mesma comece com uma letra minúscula (exigência da linguagem CLEAN).
- Feito isto, obtém-se:

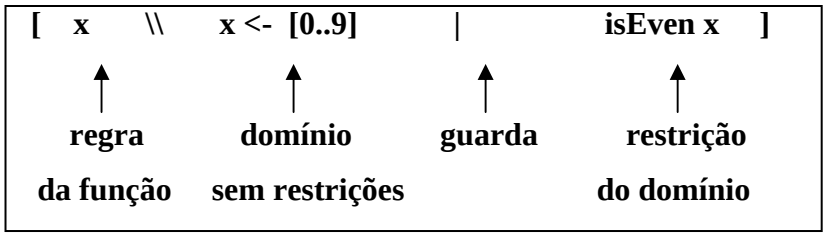
[x \\ ]

4- O próximo passo, é colocar o domínio sem suas restrições (mesmo que elas existam).
   - O domínio é: **x pertence ao conjunto dos números naturais, tal que x é maior ou igual a 0 e menor ou igual a 9** (**{x ∈ ℕ , 0 ≤ x ≤ 9}**).
   - Assim, como o teclado do computador não possui o símbolo de pertence (∈) nem o do conjunto dos naturais (ℕ), os mesmos são substituídos no CLEAN por <- e **[0..9]** respectivamente, onde [0..9] corresponde à lista com todos os naturais de 0 a 9.
   - Esta lista ([0..9]) é equivalente a **[0,1,2,3,4,5,6,7,8,9]**.
   - A notação em CLEAN fica:

[x \\ x<-[0..9]]

5- Se o que se deseja fosse apenas isto, a lista produzida seria uma cópia do domínio, ou seja, uma lista de elementos x, tal que cada elemento da imagem é igual a um elemento do domínio.
   - No exercício proposto, o domínio possui uma regra de restrição:
     - O elemento do domínio **x**, argumento da função, tem que **ser par.**
   - Para iniciar uma regra no CLEAN, deve-se colocar uma **guarda**, uma barra vertical, após a descrição geral do domínio.
   - Logo após esta guarda (|) coloca-se a(s) restrição(ões).
   - A restrição, neste caso, é que só devem ser considerados pela função os argumentos pares do domínio.
   - Uma restrição é reconhecida quando a mesma necessita de aplicação de alguma (ou mais de uma) função aos elementos do domínio.
   - Neste presente caso, deve-se aplicar uma função que verifica cada argumento do domínio se o mesmo é **par**.

- Se for, a regra da função é aplicada ao mesmo e o valor é inferido. Se não for, o elemento do domínio é recusado e nele não é aplicada a regra de inferência da função em questão que inferirá os valores do conjunto imagem.
- Em CLEAN, existe uma função que testa se um inteiro é par: **isEven**.
- A notação Zermelo-Frankel para listas fica, finalmente, da seguinte forma:



ou seja:

[x \\ x <- [0..9] | isEven x]

a lista gerada por esta notação é:

[0, 2, 4, 6, 8]

Resumindo:

- Uma compreensão de listas, portanto, consiste de duas partes delimitadas pelo abrir e fechar de colchetes, separadas por \\ (duas barras invertidas).
- A parte esquerda contém uma função ou uma regra.
- **O domínio x <– xs**, onde xs é uma lista ou um vetor, que aparem no lado direito dos sinais divisórios \\. O sinal <- significa: pertence.
- Uma expressão **x <– xs** é denominada de ***gerador***.

Sedimentando os Conceitos Fundamentais

Foi visto anteriormente a generalização e exemplos de uso da notação Zermelo-Frankel. Pôde-se perceber ser a mesma uma solução aderente ao paradigma matemático e à forma com que nosso cérebro manipula a mesma informação.

É interessante rever tais conceitos com uma abordagem diferente e com alguns detalhes a mais, utilizando um exemplo já referenciado anteriormente.

O Exemplo

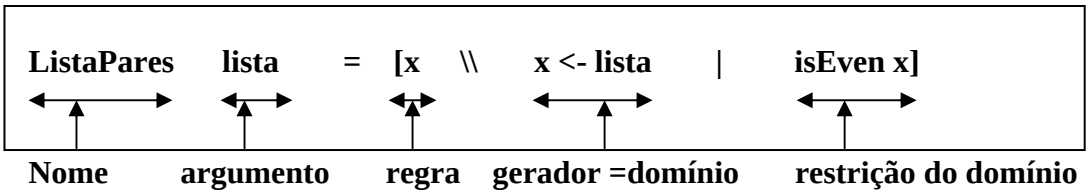
Assim, se deseja-se obter uma lista com os elementos pares inferidos de uma outra lista com uma quantidade qualquer de elementos, basta implementar um programa de forma análoga que seria a sua descrição matemática, ou seja:

ListaPares :: [Int] -> [Int]	**Declaração de tipo da função**
ListaPares lista = [x \\ x <- lista | isEven x]	**Declaração da função**

Revisando conceitos básicos

Função, argumento, domínio e regra:

- Uma função possui um nome, argumento(s) e regra(s). O sinal de igualdade (=) separa a regra da função com seu argumento.

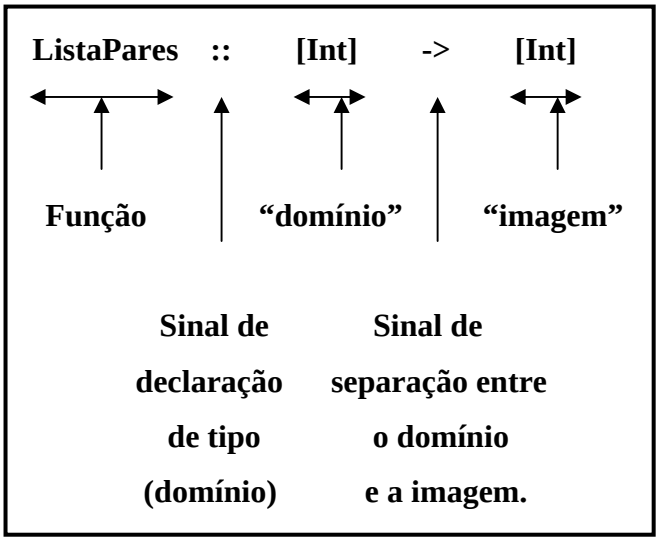


- No exemplo, o nome da função é: **ListaPares** e o seu argumento é **lista**.
- O conjunto de todos os argumentos de **lista** representa o domínio da função, o qual é novamente retratado na notação Zermelo-Frankel no gerador (**x<-lista**) que gera elemento por elemento do domínio para a inferência (determinação) dos elementos do conjunto imagem.
- Dentro da notação Zermelo-Frankel, a regra é o que está antes das duas barras invertidas, ou seja: **x**. A regra deste exemplo, **x**, diz que o elemento da imagem extraído do domínio não sofrerá modificações quando inferido por ela. Assim, se a função fosse:

ListaPares lista = [x \\ x <- lista], o resultado seria uma lista idêntica à lista dos argumentos, ou seja, o domínio e a imagem seriam iguais devido a regra discriminar que o valor dos elementos na imagem serão iguais ao do domínio, ou seja**, f(x) = x**, ou, simplesmente: **x**. .

- Assim, o domínio é o conjunto de todos os elementos representados pelo argumento da função, o que, pela declaração de tipo da função utilizada como exemplo resulta: o "domínio" é uma lista de inteiros **[Int]**.
- Na notação Zermelo-Frankel, o domínio é descrito pelo gerador da mesma, o que, no exemplo dado é: **x <- lista**.
- A imagem é o conjunto de todos os elementos resultantes da aplicação das regras de inferência da função a todos os elementos do domínio (com respectivas restrições). Pela declaração de tipo da função, a imagem, assim como o domínio, é uma lista de inteiros.

Declaração do tipo de dados do argumento e do valor da função:



- A imagem é, portanto, conforme afirmado, o conjunto formado pela aplicação da(s) regra(s) de inferência da função a todos os elementos do domínio, resguardadas as restrições feitas ao domínio na declaração da função. Neste caso, a restrição imposta é que a função somente deverá ser aplicada aos elementos pares do domínio (da lista de inteiros fornecida), ou seja: **isEven x**.

- Assim, tem-se:
  - **Regra =** devolve uma lista de elementos do tipo **x (Int)** sem alterá-los
  - **Domínio =** Os elementos de **x** pertencem à lista dada: **x <- lista**
  - **Restrição =** Só deverão ser pegos os elementos pares da lista: **isEven x**
  - **Imagem =** uma lista dos elementos pares do domínio: **[x \\ x <- lista | isEven x]**

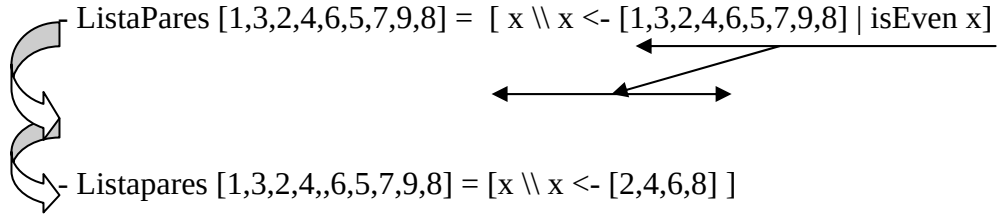
Exemplo:

Start = ListaPares [1,3,2,4,6,5,7,9,8]

Devolve:

[2,4,6,8]

ou seja:



- ListaPares [1,3,2,4,,6,5,7,9,8] = [2,4,6,8]

Obs. Perceba que o domínio que inicialmente era a lista [1,3,2,4,6,5,7,9,8], aplicada a restrição imposta ao mesmo, é como ele se tornasse: [2,4,6,8], sob o qual a regra (x) finalmente seria aplicada.

Funções com uma ou mais regras

Observe, a seguir, como implementar funções matemáticas que possuem uma ou mais regras de inferência, com respectivas condições de aplicabilidade de cada uma delas.

1- Função com uma regra de inferência:

Quando uma função possui apenas uma regra, a mesma vem logo após ao sinal de igualdade (=) .

- **1o. Exemplo**

Formalização Matemática

f(x) = **x**

A regra de inferência desta função é:

->dado um argumento do domínio, o valor no contra-domínio é igual ao valor do argumento (elemento), sem qualquer modificação. Neste caso, o conjunto imagem será igual ao conjunto domínio.

Implementação em CLEAN

```
f x = x
```

- **2o. Exemplo**

Formalização Matemática

f(x) = $\mathbf{x^2}$

A regra de inferência desta função é:

->dado um argumento do domínio, o valor no contra-domínio é igual ao valor do argumento (elemento) elevado ao quadrado. Neste caso, novamente a quantidade de elementos da imagem é igual à do domínio.

Implementação em CLEAN

```
f x = x^2
```

Função com duas regras de inferência

1º. Exemplo (uma função hipotética qualquer)

Formalização Matemática

$$\begin{cases} f(x) = \mathbf{sen(x)}, & \text{se} \quad \mathbf{0.0 < x < 3,1416} \\ f(x) = \mathbf{cos(x)}, & \text{se} \quad \mathbf{3,1416 \leq x \leq 4,7124} \end{cases}$$

regras **condições das regras**

onde: **3,1416 = Π(pi)**

e **4,7124 = 3Π/2**

Implementação em CLEAN

Funções com mais de uma regra e condições de aplicação de cada uma devem utilizar guadas (|) para iniciar cada regra. Desta forma, após o último argumento da função, deve-se iniciar uma guarda com a primeira condição. A cada condição, a cada regra, uma nova guarda deve ser iniciada. Veja como isto é feito através do seguinte exemplo:

```
f x
| x > 0.0       && x <  3.1416   = sin x
| x >= 3.1416 && x <= 4.7124  = cos x
```

Guardas **condições** **regras da função**

Obs1. Em CLEAN, em vez de vírgula para separar a parte inteira da parte decimal de um número é utilizado o sinal de ponto (**.**).

Obs2. A regra da condição que obter sucesso é a que será executada

Obs3. Observe que neste exemplo as condições de aplicabilidade das regras não estão completas, ou seja: o que aconteceria se o valor do argumento **x** fosse instanciado, assumisse um valor entre com um valor entre **3Π/2 e 2Π** (ou valores múltiplos deste e dos demais)? O que ocorre é que o programa retornaria uma mensagem de erro dizendo que não encontrou uma regra que satisfizesse. Assim, existe uma função em CLEAN

que permite se aplicar uma outra regra a qualquer condição atendida. Esta função é: **otherwise**, e deve ser colocada depois da última guarda existente, ou seja:

```
f x
| x > 0.0       &&  x <  3.1416   = sin x
| x >= 3.1416 && x <= 4.7124  = cos x
| otherwise =  abort  "argumento nao previsto pelas regras da função"
```

- Veja que nesta guarda, utiliza-se como regra, como resultado, uma informação no formato de uma String.
- Os sinais **&&** corresponde ao operador lógico **E**.
- Anterior à mensagem (a string) está a palavra reservada **abort** . A mesma é uma função que faz com que o CLEAN interrompa a execução da função e envie a mensagem, a String, logo após ela, ao usuário.
- Se entrarmos com o argumento 5.0 a esta função, **f 5.0**, a mensagem descrita **"argumento nao previsto pelas regras da função"** será apresentada e a tentativa de aplicação da função ao argumento **5.0** será abortada pela função **abort**.

2º. Exemplo (fatorial de um número)

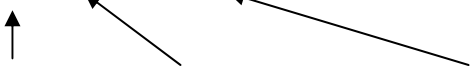
Formalização Matemática

$$\begin{cases} f(x) = \mathbf{1}, & \text{se } \mathbf{x = 0} \quad \textbf{... Obs.} \text{ Por definição, } \mathbf{0!=1}. \\ f(x) = \mathbf{x \cdot (x-1)!}, & \text{se } \mathbf{x > 0} \end{cases}$$

regras **condições das regras**

Implementação em CLEAN

```
f x
|  x == 0  = 1
|  x > 0    =  x * f (x-1)
```



Guardas condições regras da função

Utilizando a condição **otherwise** e a função regra **abort** para eventuais valores não previsto na função fatorial, tal como: -5, **f** fica:

```
f x
|  x == 0  = 1
|  x > 0    =  x * f (x-1)
|otherwise = abort ("não existe fatorial de  " +++ (toString x) )
```

Veja que neste caso a mensagem devolve uma frase que: "**nao existe fatorial de x**", onde **x** é o valor do argumento. A função **toString** transforma o argumento da função em uma string, para que a função +++ [LIMA e RUFINO] faça a concatenação de duas strings: a mensagem e o argumento convertido em string.

Observe que quando uma função possui mais de uma regra, uma condição deve ser atendida para que a presente regra seja aplicada ao argumento em questão.

1- Caso a condição da primeira regra falhe (não seja verdadeira), o CLEAN passa a verificar a próxima condição.

2- Se a condição da próxima regra falhar, novamente o CLEAN passa a verificar a próxima condição.

3- O item 2 (anterior) é executado até que uma condição seja atendida. Quando isto ocorrer, a regra desta condição é aplicada ao argumento em questão (sob teste da condição) e o valor no contradomínio é inferido.

Mais detalhes sobre a notação Zermelo-Frankel

Uma expressão x <– xs é denominado de *gerador*.

Tal como na Matemática, CLEAN permite que se utilizem condições para filtrar (eliminar) elementos do contradomínio. Veja o seguinte exemplo ilustrativo:

Exemplo: Gerar uma lista cujos elementos sejam o quadrado dos números pares que estão no intervalo de 1 a 10.

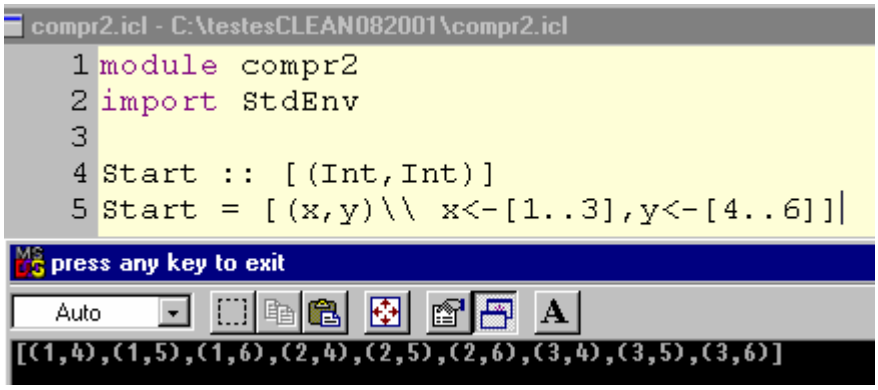
Da matemática temos: **S = { x^2 | 1 =< x >= 10 ^ x é par }**

Em CLEAN[1] temos: **S = [x*x \\ x <-[1..10] | isEven x]**

Obs. A função **isEven** é uma primitiva do CLEAN que testa se um valor é ou não par.

Utilizando mais de um *gerador* ao mesmo tempo

Na notação Zermelo-Frankel, após as duas barras invertidas (\\) pode aparecer mais de um *gerador* separados por uma vírgula. Quando isto ocorre temos uma *combinação ortogonal de geradores.* Ao se utilizar uma combinação ortogonal de geradores, a expressão colocada na frente das duas barras invertidas é calculada para toda possível combinação das variáveis correspondentes. O exemplo, a seguir, torna mais claro o que foi dito:



[1] Observe o quanto aderente, o quão parecido, é a notação em CLEAN à definição matemática.

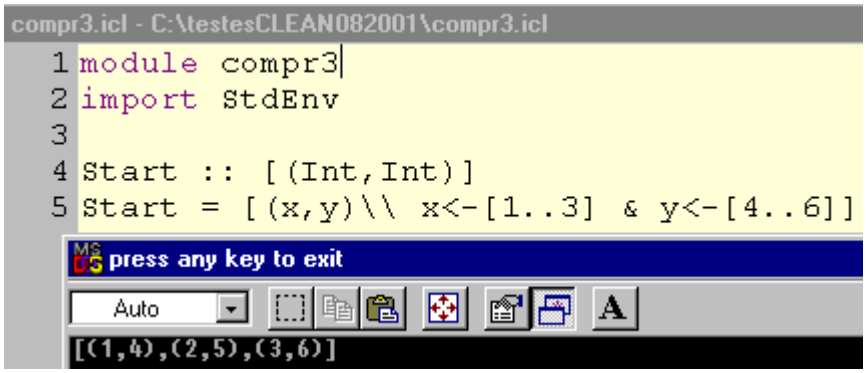
Nela, a expressão colocada na frente das duas barras invertidas é calculada para toda possível combinação das variáveis correspondentes.

Este programa devolve a lista **[(1,4),(1,5),(1,6),(2,4),(2,5),(2,6),(3,4),(3,5),(3,6)]**, ou seja, ele pega o primeiro elemento do primeiro gerrador (**x<-[1..3]**) e devolve todas as combinações com os elementos do segundo gerador (**y<-[4..6]**), ou seja: **(1,4),(1,5) e (1,6)**, depois pega o segundo elemento do primeiro gerador e devolve todas as combinações com os elementos do segundo gerador, ou seja: **(2,4),(2,5) e (2,6)**, o processo continua até que o programa tenha obtido todas as combinações possíveis de elementos do primeiro gerador comos elementos do segundo gerador. Finalmente o sistema devolve a lista **[(1,4),(1,5),(1,6),(2,4),(2,5),(2,6),(3,4),(3,5),(3,6)]**, ou seja: todas as combinações de (x,y) para os geradores dados.

Combinação paralela de geradores

Existe ainda uma outra forma de combinar geradores denominada de: *combinação paralela de geradores.* Para diferenciar da forma anterior, nesta combinação os geradores são separados pelo símbolo **&** em vez de vírgula (**,**).

A diferença entre as duas formas de utilizar geradores é que a segunda, a combinação paralela de geradores, produza como resultado apenas a combinação de elementos de mesmo índice de cada gerador, ou seja: o primeiro elemento do primeiro gerador como primeiro elemento do segundo gerador, o segundo elemento do primeiro gerador com o segundo elemento do segundo gerador, e assim sucessivamente.
Veja o exemplo a seguir:



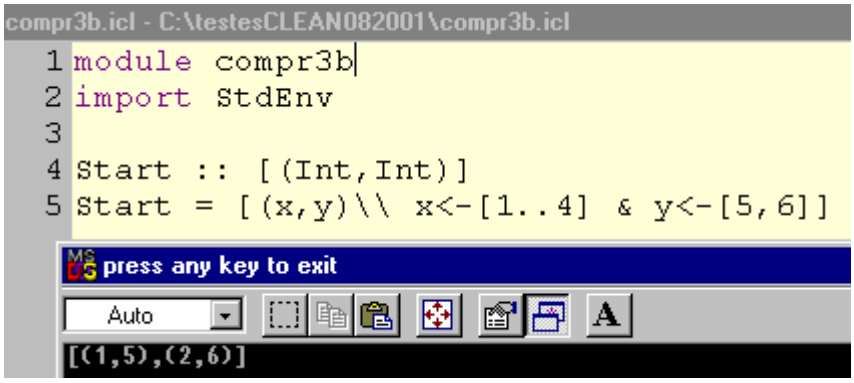
os geradores são separados pelo símbolo **&** em vez de vírgula (**,**)

O programa acima gera como resultado a lista **[(1,4),(2,5),(3,6)]** , ou seja, a combinação de elementos de mesmo índice:

Primeiro gerador – [1 , 2, 3]
Segundo gerador – [4 , 5, 6]

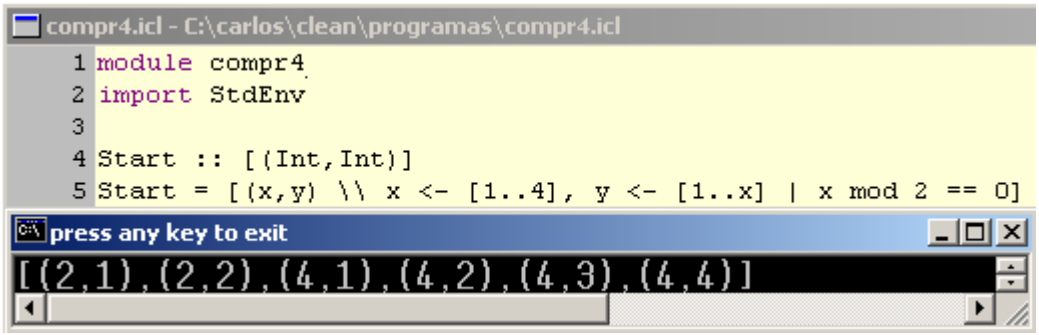
Se o primeiro gerador tiver uma quantidade de elementos diferente da do segundo gerador, o programa encerra a geração da lista quando todos os elementos do gerador que possuir a menor lista for percorrido. Veja o exemplo a seguir:

Primeiro gerador – [1 , 2, 3 , 4]
Segundo gerador – [5 , 6]



Assim, quando a menor lista é totalmente percorrida, todos os geradores combinados com **&** param de gerar elementos.

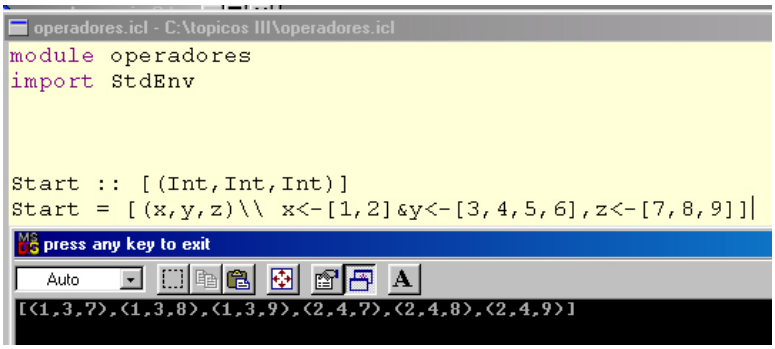
Pode-se estabelecer condições para os valores a serem gerados por uma combinação de geradores. A condição, mais uma vez, deve ser separada dos geradores por uma barra vertical. O programa devolve a lista **[(2,1),(2,2),(4,1),(4,2),(4,3),(4,4)].** Observe que a visibilidade da variável **x** (denominada de *escopo* de **x**) não se restringe somente ao lado esquerdo da compreensão. Ela também pode ser manipulada no lado direito do gerador introduzindo **x**. Entretanto **y** não pode ser utilizada nos geradores precedendo-a, isto é, **y** pode somente ser usada em **(x,y)** e na condição. O símbolo **<–** que define a pertinência é especialmente utilizado na compreensão de listas e não constitui um operador.



- Prioridade dos operadores dos geradores (& e ,).

O operador "**,**" possui prioridade em relação ao operador "**&**".

Vamos ver alguns exemplos para tornar claro esta afirmativa.



Interface Visual

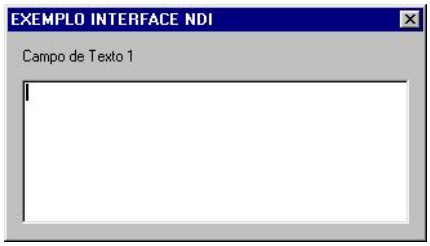
Toda interface visual e gráfica necessita da utilização de avaliações destrutivas e de ferir o princípio da transparência referencial matemática para sua viabilização. Fazer isto é matematicamente inaceitável, mas, sem tais permissões, é impossível se implementar tais interfaces. Para tanto, as linguagens funcionais modernas criaram princípios, filosofia, métodos, de como se implementar tais interfaces sem que, com isto, tais absurdos matremáticos venham acarretar efeitos colaterais graves nos produtos gerados por elas.

Assim, a linguagem Haskell [33] [34] criou **Mônadas** e O CLEAN o princípio dos **Tipos únicos**. Este princípio garante que cada variável do sistema, cada dado utilizado, seja referenciado apenas uma vez em um processo, e, como cada processo só possui variáveis e funções locais, os efeitos destrutivos causados nos processos são controlados localmente e não afetam globalmente os demais processos. A seguir, é apresentado progressivamente o projeto de interfaces com campos de texto, botões e outros recursos.

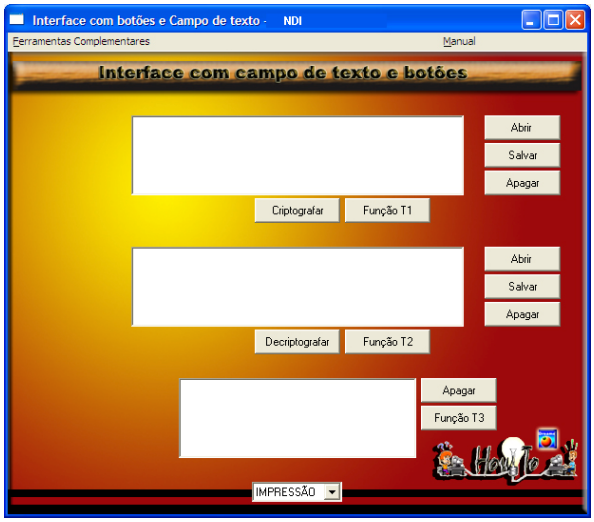
- **Interface com botões e caixa de texto**

Existem 3 tipos básicos de interface: NDI, SDI e MDI.

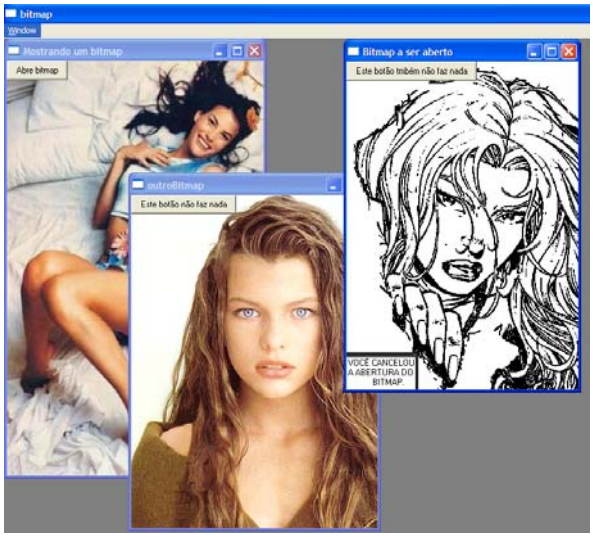
- NDI - **N**o **D**ocument **I**nterface . Esta é uma interface que possui um arcabouço, campos de textos, botões, pop-ups, mas que não permite inclusão de imagens. É uma interface denominada de diálogo. Exemplo: Janelas de informação de erros do sistema.



- SDI - **S**ingle **D**ocument **I**nterface – é uma interface semelhante ã NDI, com a diferença que pode-se colocar imagens de fundo e mudar o layout e conteúdo da janela. Possui a restrição de apenas permitir a visualização de uma janela de cada vez. Exemplo: Paint Brush, Bloco de notas, …



- MDI - **M**ultiple **D**ocument **I**nterface – Interface com todos os recursos de imagem, permitindo que se abram várias janelas ao mesmo tempo. Exemplo: Word, Photoshop, ...



No sistema proposto nesta dissertação, utilizar-se-á estes três tipos de interface. A interface MDI para permitir que vários processos sejam abertos em paralelo, a SDI para cada processo (aplicativo) e a NDI para mensagens de erro e informações gerais e, também, para aplicativos.

A diferença básica de uma NDI para uma SDI e uma MDI está apenas no cabeçalho onde se permitirá abrir imagens de fundo e criar janelas dentro de janelas. Como todos os conceitos de uma interface NDI são válidos para as outras duas, neste capítulo serão abordados detalhes relevantes à construção de NDIs.

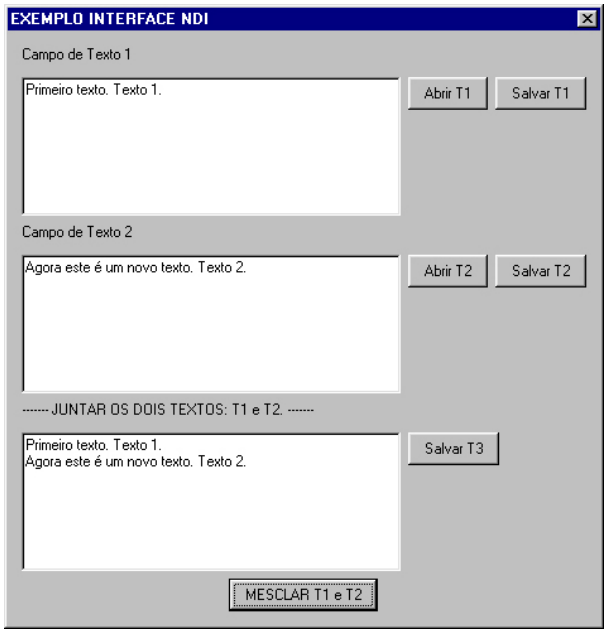
• Construção de interfaces NDIs

Uma janela é delimitada por um arcabouço, ou seja, uma barra mais grossa em azul contendo o nome da janela e linhas laterais e inferior que a delimita.

Para exemplificar, serão apresentados os passos de construção de uma janela NDI que posua os seguintes componentes:

- Três campos de texto.
- O primeiro e o segundo para colocar um texto escrito ou lido de um arquivo texto qualquer.
- O terceiro texto pode ser também editado normalmente, mas, se você clicar no botão logo abaixo dele, no mesmo será colocado os textos dos dois primeiros campos.

- Cada um dos dois primeiros textos possui um botão de abrir um arquivo texto de até 300.000 caracteres e um botão de salvar o que estiver no campo.
- No terceiro campo tem um botão de salvar e um de mesclar os dois primeiros textos. Veja na figura ao lado a interface que acabamos de especificar:



Para que se compile um programa com interface e ferramentas de IO (entrada e saída de dados), não se deve, antes, esquecer de compilar (rodar o programa). Você deve configurar o ambiente (Environment) para **Objet IO**, conforme figura a seguir:

- **Implementação do Arcabouço**

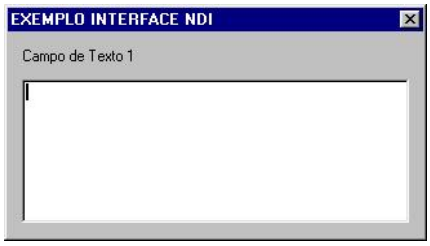
O programa a seguir possui apenas o cabeçalho principal que será utilizado nos outros exemplos. A interface abre uma moldura (arcabouço) com um lable: **MOUDURA.** Código do programa **arcabouço.icl.**

```
module arcabouço
import StdIO,StdEnv
// linha padrão que inicia uma interface NDI
Start comp  = startIO NDI 0  objeto[] comp
// função que abre  um processo para criação de
//uma janela de diálogo
objeto proc
# (_,proc)=openDialog "" dialogo proc
=proc
// função que abre a janela de diálogo, dá
// nome ao arcabouço e cria uma linha de texto
// através do comando TextControl
dialogo = Dialog "EXEMPLO INTERFACE NDI"
                (TextControl "MOUDURA" [])
                [WindowClose (noLS closeProcess)]
```

- **Colocando um novo lable e um campo de texto.**

Código do programa **um.icl**

```
module um
import StdIO,StdEnv

Start comp
= startIO NDI  0  objeto   []   comp
objeto proc
// linha que cria identificadores para os objetos (campo de texto,
// botões, ...
#(ids,proc)=openIds 2 proc
#(_,proc)=openDialog "" (dialogo ids) proc
=proc

dialogo ids = Dialog "EXEMPLO INTERFACE NDI"
// TextControl cria uma linha de texto
// ControlPos indic o alinhamento, no caso, à esquerda da janela(Left)

                (TextControl "Campo de Texto 1"
                [ControlPos (Left,zero)]
// :+:   -> cola de controles, acrescenta mais um controle
// na interface
                :+:
//EditControl cria um campo de edição, no caso, com 300 pixels de
largura
// e com 8 linhas
                EditControl "" (PixelWidth 300) 8
                [ControlId t1,ControlPos (Left,zero)]
                )
                [WindowClose (noLS closeProcess)
                ,WindowId wd1
// nomeando os identificadores (t1 é o campo de texto, wd1 é o
arcabouço)
                ]
where
                t1  =ids!!0
                wd1 =ids!!1
```



Observe, também, que na função **objeto** foram criados dois identificadores: um para a janela de diálogo e outro para o campo de texto.

A seguinte linha faz isto:

#(ids,proc)=**openIds 2** proc, em, em consequência disto, a função dialogo é agora chamada com um argumento, este argumento são os identificadores. A linha que faz isto é: #(_,proc)=openDialog "" **(dialogo ids)** proc.

A linha:

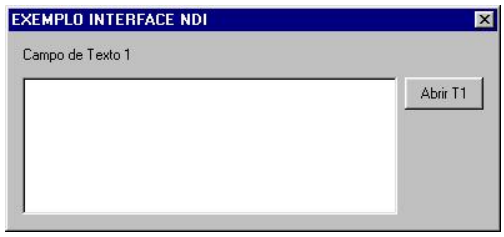
EditControl "" (PixelWidth **300**) **8** [ControlId t1,ControlPos (**Left**,zero)] cria um campo de texto de **300** pixels de largura por **8** linhas colocado no lado esquerdo (**Left**) da janela de diálogo.

O símbolo **:+:** que significa control glue (cola de controle), adiciona um controle a mais na janela de diálogo.

- **Colocando um lable, um campo de texto e um botão de abrir texto.**

O programa dois.icl

```
module dois
import StdIO,StdEnv

Start comp
= startIO NDI
        0
        objeto
        []
        comp
objeto proc
#(ids,proc)=openIds 2 proc
#(_,proc)=openDialog "" (dialogo ids) proc
=proc
dialogo ids = Dialog "EXEMPLO INTERFACE NDI"
                (TextControl "Campo de Texto 1" [ControlPos
                (Left,zero)]
                 :+:
                EditControl "" (PixelWidth 300) 8 [ControlId t1,
                ControlPos (Left,zero)]
                :+:
//ButtonControl cria um botão
//ControlFunction faz com que uma função seja disparada
//ao se clicar no respectivo botão, no caso, a função ler1
                ButtonControl "Abrir T1" [ControlFunction ler1]
                )
                [WindowClose (noLS closeProcess)
                ,WindowId wd1
                ]
where
                t1  =ids!!0
                wd1 =ids!!1
//Fun'ção para abrir um arquivo de texto através do explorer
    ler1 (el,proc)
//selectInputFile abre o explorer para selleção de um
arquivo
    #(maybeFile, proc)= selectInputFile proc
    |       isNothing maybeFile= (el,proc)
    #       (Just nome)= maybeFile
//fopen abre o arquivo selecionado
//FReadText incida que o arquivo aberto é para leitura
    # (ok, file, proc)= fopen nome FReadText proc
//freads faz a leitura de n bytes do arquivo, no caso:
// 3.000.000 Bytes
    # (conteudo, file1)= freads file 300000
// setControlText escreve em um canpo de texto,
// no caso, o conteúdo de até 3.00.000 Bytes do arquivo lido
```



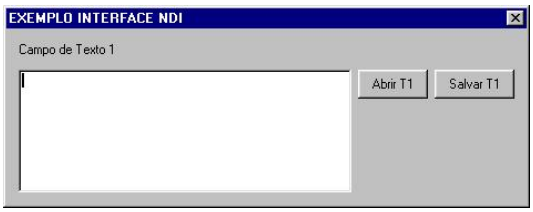
# proc= appPIO (setControlText t1 conteudo) proc //fclose fech o arquivo aberto para liberá-lo para outros programas ou processos utilizarem. # (ok, proc)= fclose file1 proc =(el,proc)	

OBS.

Veja que, como um botão foi criado, o mesmo deverá fazer alguma coisa, assim, uma função foi associada ao mesmo: **ler1**

- **Mantendo a interface criada e colocando um botão para salvar o que estiver no campo de texto**

Daqui para frente, como os programas aumentam sensivelmente, primeiro será apresentada a interface gerada, e, logo após o código do programa. O programa da próxima interface é **tres.icl**




```
module tres
import StdIO,StdEnv

Start comp
= startIO NDI
        0
        objeto
        []
        comp
objeto proc
#(ids,proc)=openIds 2 proc
#(_,proc)=openDialog "" (dialogo ids) proc
=proc
dialogo ids = Dialog "EXEMPLO INTERFACE NDI"
              (TextControl "Campo de Texto 1" [ControlPos (Left,zero)]
              :+:
              EditControl "" (PixelWidth 300) 8 [ControlId t1,ControlPos (Left,zero)]
              :+:
              ButtonControl "Abrir T1" [ControlFunction ler1]
              :+:
              ButtonControl "Salvar T1" [ControlFunction salvar1]
              )
              [WindowClose (noLS closeProcess)
              ,WindowId wd1
              ]
where
              t1  =ids!!0
              wd1 =ids!!1
//ABRIR ARQUIVO ATRAVES DO EXPLORER NO CAMPO DE TEXTO t1
              ler1 (el,proc)
              #           (maybeFile, proc)= selectInputFile proc
              |           isNothing maybeFile= (el,proc)
              #           (Just nome)= maybeFile
              # (ok, file, proc)= fopen nome FReadText proc
              # (conteudo, file1)= freads file 300000
               # proc= appPIO (setControlText t1 conteudo) proc
               # (ok, proc)= fclose file1 proc
```

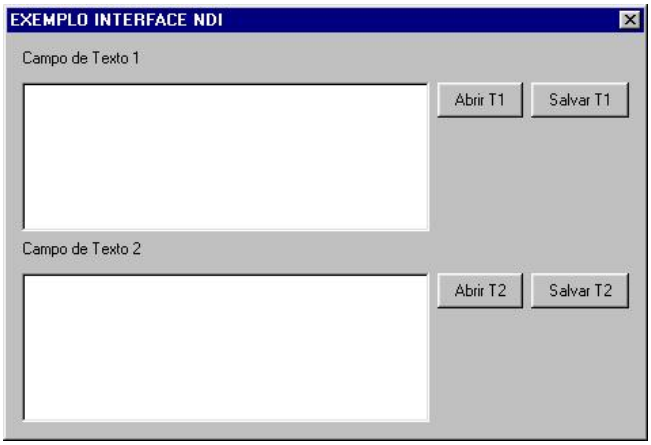
```
            =(el,proc)
//SALVAR ARQUIVO texto1  ATRAVES DO EXPLORER
            salvar1(el,proc)
//getWindow  lê os identificadores da janela
              # (Just dial,proc) = accPIO(getWindow wd1) proc
            # (se,Just x) = getControlText t1 dial
// selectOutputFile    seleciona um arquivo para ser gravado
            # (talvez, proc) = selectOutputFile "Salvar o resultado: escolha o diretorio!" "*.txt" proc
             | isNothing talvez= (el,proc)
//FWriteText prepara o endereço para gravação
            # (ok,file2, proc)= fopen (fromJust talvez) FWriteText proc
            | not ok = (el,proc)
// fwrites grava o conteúdo desjado no arquivo aberto
             # file2 = fwrites x file2
             # (ok, proc)= fclose file2 proc
            # proc= appPIO (setControlText t1 ("(* Conteudo t1 gravado com sucesso *)"+++
                    {toChar 13,toChar 10}+++ {toChar 13,toChar 10}+++ x))proc
            =(el,proc)
//XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

Observe que daqui pra frente o que se faz é apenas repetir ações e comandos já conhecidos das interfaces anteriores.

- **Acrescentando mais um campo de texto com respectivos botões. Observe que "a receita" se repete.**

O programa está é quatro.icl.

Observe que o número de identificadores de janela foi aumentado para 3, ou seja: **#(ids,proc)=**openIds 3 **proc**




```
module quatro
import StdIO,StdEnv

Start comp
= startIO NDI
        0
        objeto
        []
        comp
objeto proc
#(ids,proc)=openIds 3 proc
#(_,proc)=openDialog "" (dialogo ids) proc
=proc
dialogo ids = Dialog "EXEMPLO INTERFACE NDI"
```

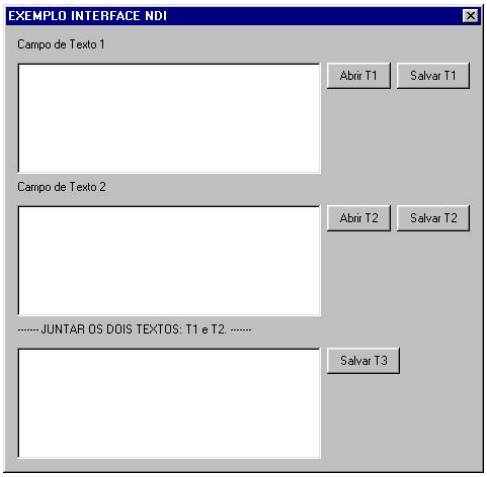
```
                (TextControl "Campo de Texto 1" [ControlPos (Left,zero)]
                 :+:
                EditControl "" (PixelWidth 300) 8
                            [ControlId t1,ControlPos (Left,zero)]
                :+:
                ButtonControl "Abrir T1" [ControlFunction ler1]
                :+:
                ButtonControl "Salvar T1" [ControlFunction salvar1]
                :+:
                TextControl "Campo de Texto 2" [ControlPos (Left,zero)]
                :+:
                EditControl "" (PixelWidth 300) 8
                            [ControlId t2,ControlPos (Left,zero)]
                :+:
                ButtonControl "Abrir T2" [ControlFunction ler2]
                :+:
                ButtonControl "Salvar T2" [ControlFunction salvar2]
                )
                [WindowClose (noLS closeProcess)
                ,WindowId wd1
                ]
where
        t1  =ids!!0
        t2  =ids!!1
        wd1 =ids!!2
//ABRIR ARQUIVO ATRAVES DO EXPLORER NO CAMPO DE TEXTO t1
        ler1 (el,proc)
        #           (maybeFile, proc)= selectInputFile proc
        |           isNothing maybeFile= (el,proc)
        #           (Just nome)= maybeFile
        # (ok, file, proc)= fopen nome FReadText proc
        # (conteudo, file1)= freads file 300000
        # proc= appPIO (setControlText t1 conteudo) proc
        # (ok, proc)= fclose file1 proc
        =(el,proc)
//SALVAR ARQUIVO texto1   ATRAVES DO EXPLORER
        salvar1(el,proc)
        # (Just dial,proc) = accPIO(getWindow wd1) proc
        #  (se,Just x) = getControlText t1 dial
        #  (talvez, proc) = selectOutputFile "Salvar o resultado: escolha o diretorio!" "*.txt" proc
        | isNothing talvez= (el,proc)
        # (ok,file2, proc)= fopen (fromJust talvez) FWriteText proc
        | not ok = (el,proc)
        # file2 = fwrites x file2
        #  (ok, proc)= fclose file2 proc
        #  proc= appPIO (setControlText t1
               ("(* Conteudo t1 gravado com sucesso *)"+++
               {toChar 13,toChar 10}+++ {toChar 13,toChar 10}
               +++ x))proc
        =(el,proc)
//XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

 //ABRIR ARQUIVO ATRAVES DO EXPLORER NO CAMPO DE TEXTO t2
        ler2 (el,proc)
        # (maybeFile, proc)= selectInputFile proc
        | isNothing maybeFile= (el,proc)
        # (Just nome)= maybeFile
        # (ok, file, proc)= fopen nome FReadText proc
        # (conteudo, file1)= freads file 300000
        # proc= appPIO (setControlText t2 conteudo) proc
        # (ok, proc)= fclose file1 proc
        =(el,proc)


//SALVAR ARQUIVO texto2 ATRAVES DO EXPLORER
        salvar2(el,proc)
        # (Just dial,proc) = accPIO(getWindow wd1) proc
        #  (se,Just x) = getControlText t2 dial
        #  (talvez, proc) = selectOutputFile "Salvar o resultado: escolha o diretorio!" "*.txt" proc
        | isNothing talvez= (el,proc)
        # (ok,file2, proc)= fopen (fromJust talvez) FWriteText proc
        | not ok = (el,proc)
        # file2 = fwrites x file2
        #  (ok, proc)= fclose file2 proc
        #  proc= appPIO (setControlText t2
               ("(* Conteudo t2 gravado com sucesso *)"+++
```

```
            {toChar 13,toChar 10}+++ {toChar 13,toChar 10}
            +++ x))proc
        =(el,proc)
//XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

- **Acrescentando mais um campo de texto com um botão de salvar e um lable antes, predizendo o que se deseja fazer neste campo de texto**

O programa é cinco.icl.

Observe que o número de identificadores de janela foi aumentado para 3, ou seja: **#(ids,proc)=**openIds 4 **proc**




```
module cinco
import StdIO,StdEnv

Start comp
= startIO NDI
        0
        objeto
        []
        comp
objeto proc
#(ids,proc)=openIds 4 proc
#(_,proc)=openDialog "" (dialogo ids) proc
=proc
dialogo ids = Dialog "EXEMPLO INTERFACE NDI"
            (TextControl "Campo de Texto 1" [ControlPos (Left,zero)]
             :+:
            EditControl "" (PixelWidth 300) 8
                    [ControlId t1,ControlPos (Left,zero)]
            :+:
            ButtonControl "Abrir T1" [ControlFunction ler1]
            :+:
            ButtonControl "Salvar T1" [ControlFunction salvar1]
            :+:
            TextControl "Campo de Texto 2" [ControlPos (Left,zero)]
            :+:
            EditControl "" (PixelWidth 300) 8
                    [ControlId t2,ControlPos (Left,zero)]
            :+:
            ButtonControl "Abrir T2" [ControlFunction ler2]
            :+:
            ButtonControl "Salvar T2" [ControlFunction salvar2]
            :+:
```

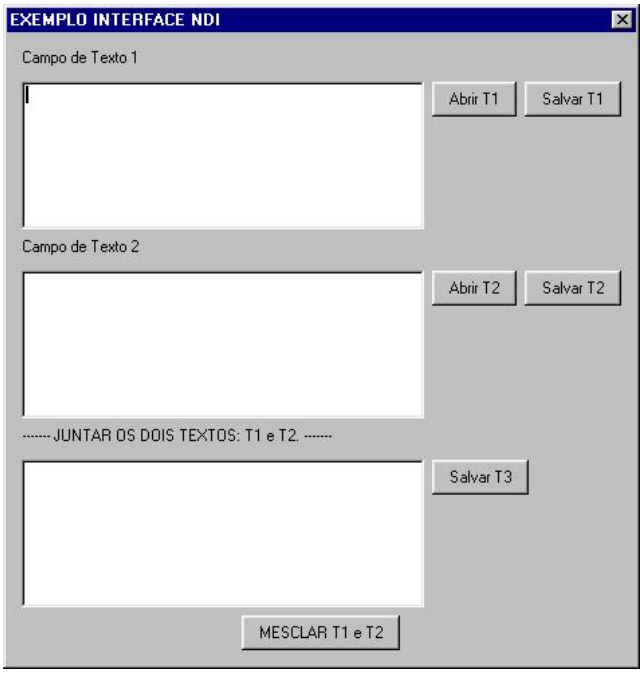
```
                TextControl "------- JUNTAR OS DOIS TEXTOS: T1 e T2. -------" [ControlPos (Left,zero)]
                :+:
                EditControl "" (PixelWidth 300) 8 [ControlId t3,ControlPos (Left,zero)]
                :+:
                ButtonControl "Salvar T3" [ControlFunction salvar3]
                )
                [WindowClose (noLS closeProcess)
                ,WindowId wd1
                ]
where
        t1  =ids!!0
        t2  =ids!!1
       t3     = ids!!2
       wd1 = ids!!3
//ABRIR ARQUIVO ATRAVES DO EXPLORER NO CAMPO DE TEXTO t1
        ler1 (el,proc)
        #           (maybeFile, proc)= selectInputFile proc
        |           isNothing maybeFile= (el,proc)
        #           (Just nome)= maybeFile
        # (ok, file, proc)= fopen nome FReadText proc
        # (conteudo, file1)= freads file 300000
       # proc= appPIO (setControlText t1 conteudo) proc
       # (ok, proc)= fclose file1 proc
       =(el,proc)
//SALVAR ARQUIVO texto1   ATRAVES DO EXPLORER
        salvar1(el,proc)
        # (Just dial,proc) = accPIO(getWindow wd1) proc
        #  (se,Just x) = getControlText t1 dial
        #  (talvez, proc) = selectOutputFile "Salvar o resultado: escolha o diretorio!" "*.txt" proc
        | isNothing talvez= (el,proc)
        # (ok,file2, proc)= fopen (fromJust talvez) FWriteText proc
        | not ok = (el,proc)
       # file2 = fwrites x file2
       #  (ok, proc)= fclose file2 proc
        #  proc= appPIO (setControlText t1
               ("(* Conteudo t1 gravado com sucesso *)"+++
               {toChar 13,toChar 10}+++ {toChar 13,toChar 10}
               +++ x))proc
        =(el,proc)
//XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 //ABRIR ARQUIVO ATRAVES DO EXPLORER NO CAMPO DE TEXTO t2
        ler2 (el,proc)
        # (maybeFile, proc)= selectInputFile proc
        | isNothing maybeFile= (el,proc)
        # (Just nome)= maybeFile
        # (ok, file, proc)= fopen nome FReadText proc
        # (conteudo, file1)= freads file 300000
       # proc= appPIO (setControlText t2 conteudo) proc
       # (ok, proc)= fclose file1 proc
       =(el,proc)
//SALVAR ARQUIVO texto2 ATRAVES DO EXPLORER
        salvar2(el,proc)
        # (Just dial,proc) = accPIO(getWindow wd1) proc
       #  (se,Just x) = getControlText t2 dial
        #  (talvez, proc) = selectOutputFile "Salvar o resultado: escolha o diretorio!" "*.txt" proc
        | isNothing talvez= (el,proc)
        # (ok,file2, proc)= fopen (fromJust talvez) FWriteText proc
        | not ok = (el,proc)
       # file2 = fwrites x file2
       #  (ok, proc)= fclose file2 proc
        #  proc= appPIO (setControlText t2
               ("(* Conteudo t2 gravado com sucesso *)"+++
               {toChar 13,toChar 10}+++ {toChar 13,toChar 10}
               +++ x))proc
        =(el,proc)
//XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
//SALVAR ARQUIVO texto3 ATRAVES DO EXPLORER
        salvar3(el,proc)
        # (Just dial,proc) = accPIO(getWindow wd1) proc
       #  (se,Just x) = getControlText t3 dial
        #  (talvez, proc) = selectOutputFile "Salvar o resultado: escolha o diretorio!" "*.txt" proc
        | isNothing talvez= (el,proc)
        # (ok,file2, proc)= fopen (fromJust talvez) FWriteText proc
        | not ok = (el,proc)
       # file2 = fwrites x file2
       #  (ok, proc)= fclose file2 proc
```

```
#  proc= appPIO (setControlText t3
        ("(* Conteudo t2 gravado com sucesso *)"+++
        {toChar 13,toChar 10}+++ {toChar 13,toChar 10}
        +++ x))proc
=(el,proc)
```

- **Acrescentando um botão que ao ser acionado faça um merge dos dois textos, ou seja, t1 com t2, e coloca o resultado no campo de texto t3.**

O programa está em **ultimo.icl.**

Foi criada uma função externa à interface (você pode verificar que a endentação não é a mesma, ou seja, está iniciando no alinhamento esquerdo do texto. Esta função serve para saltar uma linha. A mesma já estava sendo utilizada antes mas sem ser uma função {toChar 13, toChar 10}.




```
module cinco
import StdIO,StdEnv

Start comp
= startIO NDI
        0
        objeto
        []
        comp
objeto proc
#(ids,proc)=openIds 4 proc
#(_,proc)=openDialog "" (dialogo ids) proc
=proc
dialogo ids = Dialog "EXEMPLO INTERFACE NDI"
              (TextControl "Campo de Texto 1" [ControlPos (Left,zero)]
               :+:
              EditControl "" (PixelWidth 300) 8
```

```
                    [ControlId t1,ControlPos (Left,zero)]
                :+:
                ButtonControl "Abrir T1" [ControlFunction ler1]
                :+:
                ButtonControl "Salvar T1" [ControlFunction salvar1]
                :+:
                TextControl "Campo de Texto 2" [ControlPos (Left,zero)]
                :+:
                EditControl "" (PixelWidth 300) 8
                    [ControlId t2,ControlPos (Left,zero)]
                :+:
                ButtonControl "Abrir T2" [ControlFunction ler2]
                :+:
                ButtonControl "Salvar T2" [ControlFunction salvar2]
                :+:
                TextControl "------- JUNTAR OS DOIS TEXTOS: T1 e T2. -------" [ControlPos (Left,zero)]
                :+:
                EditControl "" (PixelWidth 300) 8 [ControlId t3,ControlPos (Left,zero)]
                :+:
                ButtonControl "Salvar T3" [ControlFunction salvar3]
                :+:
                ButtonControl "MESCLAR T1 e T2" [ControlFunction mesclar,ControlPos (Center,zero)]
                )
                [WindowClose (noLS closeProcess)
                ,WindowId wd1
                ]
where
        t1    = ids!!0
        t2    = ids!!1
       t3     = ids!!2
      wd1 = ids!!3
//ABRIR ARQUIVO ATRAVES DO EXPLORER NO CAMPO DE TEXTO t1
        ler1 (el,proc)
        #           (maybeFile, proc)= selectInputFile proc
        |           isNothing maybeFile= (el,proc)
        #           (Just nome)= maybeFile
        # (ok, file, proc)= fopen nome FReadText proc
        # (conteudo, file1)= freads file 300000
       # proc= appPIO (setControlText t1 conteudo) proc
       # (ok, proc)= fclose file1 proc
       =(el,proc)
//SALVAR ARQUIVO texto1   ATRAVES DO EXPLORER
        salvar1(el,proc)
        # (Just dial,proc) = accPIO(getWindow wd1) proc
        #  (se,Just x) = getControlText t1 dial
        #  (talvez, proc) = selectOutputFile "Salvar o resultado: escolha o diretorio!" "*.txt" proc
        | isNothing talvez= (el,proc)
        # (ok,file2, proc)= fopen (fromJust talvez) FWriteText proc
        | not ok = (el,proc)
       # file2 = fwrites x file2
       #  (ok, proc)= fclose file2 proc
        #  proc= appPIO (setControlText t1
              ("(* Conteudo t1 gravado com sucesso *)"+++
              {toChar 13,toChar 10}+++ {toChar 13,toChar 10}
              +++ x))proc
        =(el,proc)
//XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 //ABRIR ARQUIVO ATRAVES DO EXPLORER NO CAMPO DE TEXTO t2
        ler2 (el,proc)
        # (maybeFile, proc)= selectInputFile proc
        | isNothing maybeFile= (el,proc)
        # (Just nome)= maybeFile
        # (ok, file, proc)= fopen nome FReadText proc
        # (conteudo, file1)= freads file 300000
       # proc= appPIO (setControlText t2 conteudo) proc
       # (ok, proc)= fclose file1 proc
       =(el,proc)
//SALVAR ARQUIVO texto2 ATRAVES DO EXPLORER
        salvar2(el,proc)
        # (Just dial,proc) = accPIO(getWindow wd1) proc
       #  (se,Just x) = getControlText t2 dial
        #  (talvez, proc) = selectOutputFile "Salvar o resultado: escolha o diretorio!" "*.txt" proc
        | isNothing talvez= (el,proc)
        # (ok,file2, proc)= fopen (fromJust talvez) FWriteText proc
        | not ok = (el,proc)
       # file2 = fwrites x file2
```

```
            #  (ok, proc)= fclose file2 proc
             #  proc= appPIO (setControlText t2
                    ("(* Conteudo t2 gravado com sucesso *)"+++
                    {toChar 13,toChar 10}+++ {toChar 13,toChar 10}
                    +++ x))proc
             =(el,proc)
      //XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
      //SALVAR ARQUIVO texto3 ATRAVES DO EXPLORER
             salvar3(el,proc)
             # (Just dial,proc) = accPIO(getWindow wd1) proc
             #  (se,Just x) = getControlText t3 dial
             #  (talvez, proc) = selectOutputFile "Salvar o resultado: escolha o diretorio!" "*.txt" proc
             | isNothing talvez= (el,proc)
             # (ok,file2, proc)= fopen (fromJust talvez) FWriteText proc
             | not ok = (el,proc)
            # file2 = fwrites x file2
            #  (ok, proc)= fclose file2 proc
             #  proc= appPIO (setControlText t3
                    ("(* Conteudo t2 gravado com sucesso *)"+++
                    {toChar 13,toChar 10}+++ {toChar 13,toChar 10}
                    +++ x))proc
             =(el,proc)
//XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
//mesclar t1 e t2
             mesclar(el,proc)
             #(Just dial,proc) = accPIO(getWindow wd1) proc
            # (_,Just x) = getControlText t1 dial
            # (_,Just y) = getControlText t2 dial
            # proc= appPIO (setControlText t3 (x+++saltaLinha+++y))proc
            =(el,proc)

//Função externa para saltar linha  (\ = toChar 13)  e (n = toChar 10)
saltaLinha = {toChar 13,toChar 10}
```

A partir dos conceitos e ferramentas apresentadas neste capitulo, fica mais fácil o entendimento do código gerado no aplicativo desta tese, a aderência da linguagem ao problema proposto, bem como facilita a um usuário leigo nesta linguagem iniciar seus primeiros trabalhos, com interface visual ou não, implementando alguns aplicativos em diversas áreas do conhecimento. Para maiores detalhes, indica-se o banco de teses e dissertações da FEELT – UFU, a qual possui informações e trabalhos relevantes utilizando este paradigma, o funcional, e a linguagem CLEAN.

APÊNDICE 1-

FONOLOGIA E FONÉTICA

Extraído do endereço:

http://images.google.com.br/imgres?imgurl=http://criarmundos.do.sapo.pt/Linguistica/images/aparelho-fonador.gif&imgrefurl=http://criarmundos.do.sapo.pt/Linguistica/pesquisalinguistica02.html&h=358&w=400&sz=11&tbnid=Lgoeo4WFI76VkM:&tbnh=107&tbnw=120&hl=pt-BR&start=1&prev=/images%3Fq%3D%2522aparelho%2Bfonador%2522%26svnum%3D10%26hl%3Dpt-BR%26lr%3D%26sa%3DG

e registrado aqui para consulta, caso o site saia do ar ou mude de endereço.

Fonologia e Fonética

Na construção de uma língua é preciso, em primeiro lugar, pensar em fonologia e fonética, ou seja, saber o que são e como tratar os sons. Então e qual é a diferença entre fonologia e fonética? Bom, a fonologia estuda o comportamento dos sons e dos fonemas numa língua, enquanto a fonética estuda os sons e os fonemas (incluindo a sua evolução).

Claro que, antes de estudarmos os sons e os seus comportamentos, é preciso saber como são os sons produzidos. Afinal, quem quiser inventar uma língua extraterrestre tem de pensar no modo como os seus extraterrestres produzem sons.

O Aparelho Fonador e o seu Funcionamento

Para que se produzam os sons que caracterizam a fala humana são necessárias três condições:

corrente de ar;

obstáculo à corrente de ar;

caixa de ressonância;

o que se traduz no **aparelho fonador** humano:

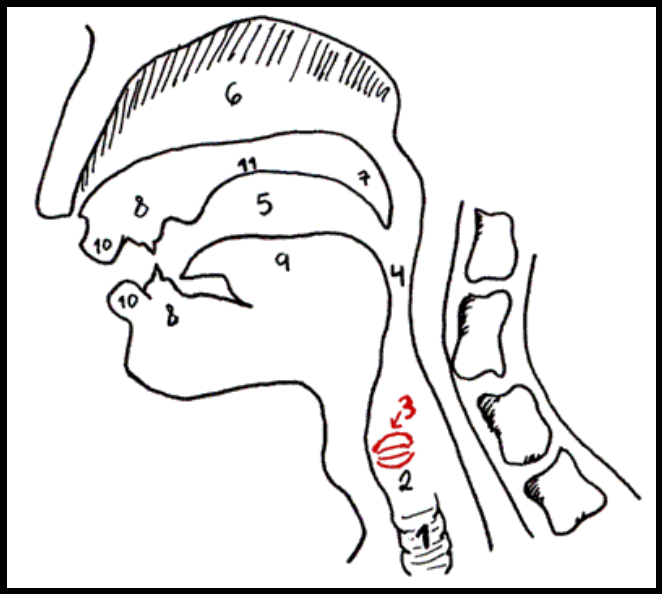
Os pulmões, brônquios e traqueia - São os órgãos respiratórios que permitem a corrente de ar, sem a qual não existiriam sons. A maioria dos sons que conhecemos são produzidos na expiração, servindo a inspiração como um momento de pausa; no entanto, há línguas que produzem sons na inspiração, como o zulo e o boximane - são os chamados cliques.

A laringe, onde ficam as cordas vocais - Determinam a sonoridade (a vibração das cordas vocais) dos sons.

A faringe, boca (e língua) e as fossas nasais - Formam a caixa de ressonância responsável por grande parte da variedade de sons.

Olhemos por um momento para o esquema do aparelho fonador antes de seguir o percurso do ar na produção de sons.

Esquema do Aparelho Fonador



1 - Traqueia

2 - Laringe

3 - Glote (Cordas vocais)

4 - Faringe

5 - Cavidade bucal

6 - Cavidade nasal

7 - Véu palatino ou Palato mole

8 - Maxilares (dentes)

9 - Língua

	10 - Lábios 11 - Palato duro (céu da boca)

Ao expirar, os pulmões libertam ar que passa pelos brônquios para entrar na **traqueia(1)** e chegar à **laringe(2)**. Na laringe o ar encontra o seu primeiro obstáculo: a **glote(3)** (mais ao menos ao nível da maçã-de-adão, chamada de *gogó* no Brasil), mais conhecida como cordas vocais. Semelhantes a duas pregas musculares, as cordas vocais podem estar fechadas ou abertas: se estiverem abertas, o ar passa sem real obstáculo, dando origem a um **som surdo**; se estiverem fechada, o ar força a passagem fazendo as pregas muscular vibrar, o que dá origem a um **som sonoro**.

Para se perceber melhor a diferença, experimente-se dizer "k" e "g" (não "kê" ou "kapa", nem "gê" ou "jê"; só os sons "k" e "g") mantendo os dedos na maçã-de-adão. No primeiro caso não se sentirá vibração, mas com o "g" sentir-se-á uma ligeira vibração - cuidado apenas para não se dizerem vogais, pois são todas sonoras.

Depois de sair da **laringe(2)**, o ar entra na **faringe(4)** onde encontra uma encruzilhada: primeiro a entrada para a **boca(5)** e depois a para as **fossas nasais(6)**. No meio está o **véu palatino(7)** que permite que o ar passe livremente pelas duas cavidades, originando um **som nasal**; ou que impede a passagem pela cavidade nasal, obrigando o ar a passar apenas pela cavidade bucal - resultando num **som oral**.

A diferença é óbvia: compare-se o primeiro "a" em "**A**na" com o de "m**a**nta". A primeira vogal é oral e a segunda é nasal.

Por fim, o ar está na cavidade bucal (a boca) que funciona como uma caixa de ressonância onde, usando os **maxilares(8)**, as bochechas e, especialmente, a **língua(9)** e os **lábios(10)**, podem modular-se uma infinidade de sons.

A título de curiosidade, gostaria apenas de recordar um pouco a história do Homem. Discute-se que a linguagem humana pode ter surgido há cerca de 100 mil anos, mas pensemos numa época mais recente - há cerca de 40 mil anos. Nesta altura, e devido a reconstruções tendo por base o registo arqueológico, sabe-se que o aparelho fonador dos Neandertais tinha algumas diferenças marcantes do Homem moderno, nomeadamente, a laringe encontrava-se mais elevada. Isto significa que a língua tinha uma mobilidade menor, limitando a possibilidade da produção de sons.

Som e Fonema - Transcrições

Bom, até aqui já vimos como são os sons produzidos de um modo básico. Mas muitas questões estão ainda por resolver: por exemplo, qual a diferença entre um "p" e um "k"? Onde e como são estes sons produzidos? A resposta, no entanto, tem de ser um pouco

adiada. Primeiro é preciso estabelecer algumas noções relativas aos sons e à sua transcrição para que uns não falem de "alhos" e outros entendam "bugalhos"!

Para começar é preciso distinguir som de fonema. Se todos sabemos o que é um som (ainda agora mesmo vimos como se produziam!), então o que é um fonema? Um fonema é um elemento de significado, o mais pequeno que existe numa palavra - e que quase se pode confundir com um som! Repare-se nas seguintes palavras:

saco **taco**

Se não fosse pelo "s" e "t" iniciais, as palavras não se distinguiriam. Assim, tratam-se de duas unidades - representadas fisicamente pelo som (tornam-se audíveis) - que representam uma idéia. E como se distinguem sons de fonemas? Porque o som é representado entre [parêntesis rectos] e o fonema entre /barras/, enquanto as letras são representadas entre "aspas". Concluindo: nas palavras "saco" e "taco" os sons [s] e [t], representados pelas letras "s" e "t", correspondem aos fonemas /s/ e /t/. No entanto, o fonema /s/ pode também ser escrito com "ss" ("assado"), com "ç" ("aço"), com "c" ("cerca"), ou com "x" ("próximo"); podendo ser realizado quer com o som [s], no português normal, quer com o som [S], em certas regiões do Norte de Portugal e da Galiza.

Agora vem um outro problema: como é que se sabe que som é qual quando se escreve [a]? Será o [a] de "àrvore" ou de "cana"? Sabe-se que é o [a] de "àrvore" porque existe um **alfabeto fonético internacional**, que convencionou os símbolos que representam cada som e fonema. (Apesar de poder haver algumas interpretações ligeiramente diferentes dos símbolos de língua para língua.)

E, como não há memória que consiga reter tão grande lista, apresentamo-la numa página à parte de modo que possa ficar sempre à mão à medida que for lendo este capítulo:

Alfabeto Fonético Internacional	
Nota 1: Este quadro apresenta os sons característicos da língua portuguesa, apresentando ainda alguns outros sons usados (dialectais e não só). *Nota 2: Cada som é dado com exemplos, sendo que o som correspondente será marcado nos exemplos a **cor**.*	
Vogais	
[a]	Como em "**á**gua"; produzido com a língua numa posição de repouso e sem

	elevação do seu dorso, sem arredondamento dos lábios e boca ligeiramente aberta.
[ɑ]	Como em "c**a**n**a**"; produzido com a língua numa posição de repouso e com o seu dorso um pouco elevado, em comparação ao [a], sem arredondamento dos lábios e boca ligeiramente fechada.
[ɛ]	Como em "p**é**"; produzido com a língua elevada em direcção ao palato duro (céu da boca) e com o seu dorso ligeiramente elevado, sem arredondamento dos lábios e boca ligeiramente aberta.
[e]	Como em "m**e**do"; produzido com a língua elevada em direcção ao palato duro (céu da boca) e com o seu dorso um pouco elevado, sem arredondamento dos lábios e boca ligeiramente fechada.
[ə]	Como em "sed**e**"; produzido com a língua numa posição de repouso e com o seu dorso elevado, sem arredondamento dos lábios e boca quase fechada.
[ɔ]	Como em "c**o**la"; produzido com a língua elevada em direcção ao véu palatino e com o seu dorso ligeiramente elevado, com arredondamento dos lábios e boca ligeiramente aberta.
[o]	Como em "b**o**lo"; produzido com a língua elevada em direcção ao véu palatino e com o seu dorso um pouco elevado, com arredondamento dos lábios e boca ligeiramente fechada.
[i]	Como em "p**i**lha"; produzido com a língua elevada em direcção ao palato duro (céu da boca) e com o seu dorso elevado, sem arredondamento dos lábios e boca ligeiramente fechada.
[u]	Como em "s**u**l"; produzido com a língua elevada em direcção ao véu palatino e com o seu dorso elevado, com arredondamento dos lábios e boca quase fechada.
Semivogais	
[j]	Como em "pra**i**a"
[w]	Como em "pa**u**"
Consoantes	
[b]	Como em "am**b**os"; os lábios tocam-se obstruindo a passagem do ar, as cordas vocais vibram.

[β]	Como em "**b**oi"; os lábios tocam-se apenas muito ligeiramente obstruindo a passagem do ar, as cordas vocais vibram.
[d]	Como em "an**d**ar"; a parte imediatamente anterior à ponta da língua toca a parte interior dos dentes incisivos do maxilar superior obstruindo a passagem do ar, as cordas vocais vibram.
[ð]	Como em "espa**d**a"; a parte imediatamente anterior à ponta da língua mal toca a parte interior dos dentes incisivos do maxilar superior obstruindo a passagem do ar, as cordas vocais vibram.
[g]	Como em "fran**g**o"; a parte posterior da língua toca o palato mole, ou véu palatino, obstruindo a passagem do ar, as cordas vocais vibram.
[ɣ]	Como em "a**g**rado"; a parte posterior da língua mal toca o palato mole, ou véu palatino, obstruindo a passagem do ar, as cordas vocais vibram.
[p]	Como em "**p**ata"; os lábios tocam-se obstruindo a passagem do ar, as cordas vocais não vibram.
[t]	Como em "a**t**ado"; a parte imediatamente anterior à ponta da língua toca a parte interior dos dentes incisivos do maxilar superior obstruindo a passagem do ar, as cordas vocais não vibram.
[t∫]	Como em "**ch**ave" nalgumas zonas do Norte de Portugal, e como em "**tch**au"; a parte imediatamente anterior à ponta da língua toca a parte interior dos dentes incisivos do maxilar superior obstruindo a passagem do ar, e depois a língua desliza depressa para trás, forçando o ar a passar por uma fenda estreita entre o dorso da língua e o palato duro, ou céu da boca, as cordas vocais não vibram.
[k]	Como em "por**c**o"; a parte posterior da língua toca o palato mole, ou véu palatino, obstruindo a passagem do ar, as cordas vocais não vibram.
[m]	Como em "ar**m**a"; os lábios tocam-se obstruindo a passagem do ar, as cordas vocais vibram e é uma consoante nasal.
[n]	Como em "ca**n**o"; a ponta da língua toca os alvéolos no maxilar superior obstruindo a passagem do ar, as cordas vocais vibram e é uma consoante nasal.
[ɲ]	Como em "vi**nh**a"; o dorso da língua toca o palato duro, ou céu da boca, obstruindo a passagem do ar, as cordas vocais vibram e é uma consoante nasal.
[l]	Como em "ca**l**o"; a ponta da língua toca os alvéolos no maxilar superior permitindo a passagem do ar lateralmente, as cordas vocais vibram.
[ɫ]	Como em "me**l**"; a ponta da língua mal toca os alvéolos no maxilar superior

	permitindo a passagem do ar lateralmente, as cordas vocais vibram.
[]	Como em "a**lh**o"; o dorso da língua toca o palato duro, ou céu da boca, forçando o ar a passar lateralmente, as cordas vocais vibram.
[r]	Como em "ca**r**o"; a ponta da língua toca, vibrando, os alvéolos no maxilar superior, as cordas vocais vibram.
[]	Como em "ca**r**o" nalgumas zonas de Portugal; a ponta da língua toca, vibrando, os alvéolos no maxilar superior interrompendo por diversas vezes a passagem do ar, as cordas vocais vibram.
[R]	Como em "ca**rr**o"; a parte posterior da língua toca, vibrando, no palato mole, ou véu palatino, interrompendo por diversas vezes a passagem do ar, as cordas vocais vibram.
[f]	Como em "**f**aca"; o ar é forçado a passar por entre os dentes incisivos do maxilar superior e o lábio inferior, as cordas vocais não vibram.
[v]	Como em "**v**aca"; o ar é forçado a passar por entre os dentes incisivos do maxilar superior e o lábio inferior, as cordas vocais vibram.
[s]	Como em "po**ss**o"; a parte imediatamente anterior à ponta da língua aproxima-se da parte interior dos dentes incisivos do maxilar superior formando uma passagem estreita (como uma fenda) à passagem do ar, as cordas vocais não vibram.
[]	Como em "po**ss**o" nalgumas zonas Norte de Portugal; a ponta da língua aproxima-se da parte interior dos dentes incisivos do maxilar superior formando uma passagem estreita (como uma fenda) à passagem do ar, as cordas vocais não vibram.
[z]	Como em "ca**s**a"; a parte imediatamente anterior à ponta da língua aproxima-se da parte interior dos dentes incisivos do maxilar superior formando uma passagem estreita (como uma fenda) à passagem do ar, as cordas vocais vibram.
[]	Como em "ca**s**a" nalgumas zonas Norte de Portugal; a ponta da língua aproxima-se da parte interior dos dentes incisivos do maxilar superior formando uma passagem estreita (como uma fenda) à passagem do ar, as cordas vocais vibram.
[]	Como em "a**ch**o"; o ar é forçado a passar por uma fenda estreita entre o dorso da língua e o palato duro, ou céu da boca, as cordas vocais não vibram.
[]	Como em "**g**enro"; o ar é forçado a passar por uma fenda estreita entre o dorso da língua e o palato duro, ou céu da boca, as cordas vocais vibram.

A Classificação dos Sons Linguísticos

Para a classificação dos sons é preciso ter em mente três questões importantes:

- Como é que os sons são produzidos?
- Como são transmitidos?
- Como são entendidos?

Tradicionalmente, devido à complexidade óbvia na classificação segundo a transmissão e a compreensão, a classificação dos sons baseia-se essencialmente no modo como os sons são produzidos, ou seja, na sua articulação. No entanto, em alguns pontos classificatórios também se baseia no modo como são transmitidos, ou seja, na acústica. Como este capítulo não pretende ser exaustivo, mas ajudar quem não tem conhecimentos neste campo, tentarei ser o mais simples e clara que for possível (mesmo que, para isso, simplifique demais a gramática).

Os sons classificam-se segundo três categorias:

Vogais: os sons produzidos sem obstáculo à passagem do ar na cavidade bucal (apenas varia a abertura à passagem do ar causada pelos maxilares, língua e lábios), e com vibração das cordas vocais.

Consoantes: os sons produzidos com obstáculo à passagem do ar na cavidade bucal.

Semivogais: dois sons , [j] e [w], que formam uma sílaba com uma vogal – ditongos e tritongos. Pode-se dizer que são quase "formas fracas" de [i] e [u], estando a meio-caminho entre vogais e consoantes.

Classificação das Vogais

As vogais da língua portuguesa podem ser classificadas quanto:

- à região de articulação (ver o esquema do aparelho fonador)
- palatais ou anteriores (língua elevada na zona do **palato duro(11)**)
- centrais ou médias (língua na posição de descanso)
- velares ou posteriores (língua elevada na zona do **véu palatino(7)**)
- ao grau de abertura (elevação do dorso da língua em direcção ao palato)
- abertas (o maior grau de abertura à passagem do ar)
- semi-abertas
- semi-fechadas
- fechadas (o menor grau de abertura à passagem do ar)
- ao arredondamento ou não dos lábios
- arredondadas
- não-arredondadas
- ao papel das cavidades bucal e nasal
- orais
- nasais

De acordo com esta caracterização pode-se preencher o quadro abaixo (em que a nasalidade é marcada pelo til como, por exemplo, em [ã])

	Palatais	**Médias**	**Posteriores**
Fechadas	[i] [ĩ]	[ə]	[u] [ũ]
Semi-fechadas	[e] [ẽ]	[ɐ] [ɐ̃]	[õ] [o]
Semi-abertas	[ɛ]		[ɔ]
Abertas		[a]	
	Não-arredondadas	**Não-arredondadas**	**Arredondadas**

Classificação das Consoantes
As dezanove consoantes da língua portuguesa podem ser classificadas quanto:

ao modo de articulação (o ar encontra sempre obstáculo à sua passagem - ver o esquema da cavidade bucal)

oclusivas (passagem do ar interropida momentaneamente)

constritivas (passagem do ar parcialmente obstruída)

fricativas (passagem do ar por uma fenda estreita no meio da via bucal; som que lembra o de fricção)

laterais (passagem do ar pelos dois lados da cavidade bucal, pois o meio encontra-se obstruído de algum modo)

vibrantes (caracterizadas pelo movimento vibratório rápido da língua ou do véu palatino)

ao ponto ou zona de articulação (o local onde é feita a obstrução à passagem do ar)

bilabiais (contacto dos lábios superior e inferior)

labiodentais (contacto dos dentes do maxilar superior com o lábio inferior)

linguodentais (aproximação ou contacto da zona anterior à ponta da língua com a face interior dos dentes do maxilar superior)

alveolares (contacto da ponta da língua com os alvéolos no maxilar superior)

palatais (contacto do dorso da língua com o palato duro, ou céu da boca)

velares (contacto da parte posterior da língua com o palato mole, ou véu palatino)

ao papel das cordas vocais

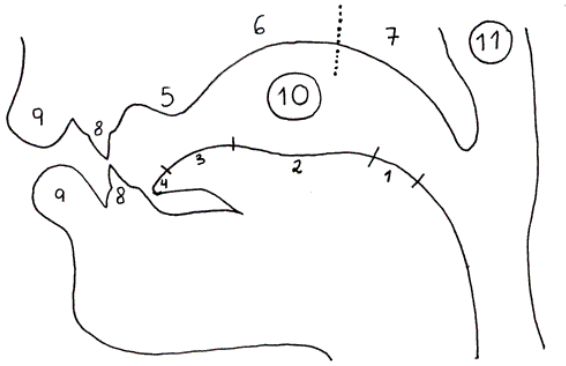
surdas (ausência de vibração das cordas vocais)

sonoras (vibração das cordas vocais)

ao papel das cavidades bucal e nasal

orais (passagem do ar apenas pela cavidade bucal)

nasais (passagem do ar pelas cavidades bucal e nasal)

Esquema da Cavidade Bucal	
	1 - Parte posterior da língua 2 - Dorso da língua 3 - Pré-dorso da língua 4 - Ápice ou ponta da língua 5 - Alvéolos 6 - Palato duro (céu da boca) 7 - Véu palatino ou Palato mole 8 - Dentes 9 - Lábios 10 - Cavidade bucal 11 - Passagem para a cavidade nasal

Através desta classificação pode-se preencher o seguinte quadro das 19 consoantes portuguesas:

Papel das cavidades bucal e nasal	**Orais**						**Nasais**
Modo de articulação	**Oclusivas**		**Fricativas**		**Laterais**	**Vibrantes**	**Oclusivas**
Papel das Cordas Vocais	**Surd**	**Son**	**Surd**	**Son**	**Son**	**Son**	**Son**

Bilabiais	[p]	[b]					[m]
Labiodentais			[f]	[v]			
Linguodentais	[t]	[d]	[s]	[z]			
Alveolares					[l]	[r]	[n]
Palatais			[ʃ]	[ʒ]	[ʎ]		[ɲ]
Velares	[k]	[g]				[R]	

Gostaria ainda de fazer uma nota quanto ao número de 19 consoantes que foi acima referido, pois este número não contempla certas variantes (como o [tʃ] ou o [ɣ]), nem as limitações que a língua impõe. Neste último caso, como em todas as línguas, existem algumas proibições quanto à posição de certas consoantes no início ou final de palavra, assim como em seguimento de certas palavras. Por exemplo, [r] nunca pode surgir em início de palavra.

Encontros vocálicos - Ditongos e Tritongos
Encontros vocálicos é o mesmo que dizer **ditongo** ou **tritongo**, ou seja, um conjunto de uma vogal e uma ou duas semivogais - que é a única altura em que surgem semivogais no português. Não devem, portanto, ser confundidos com **hiatos**: o encontro de duas vogais.

Os ditongos podem ser crescentes (pouco comum, pois são instáveis) ou decrescentes, consoante a vogal esteja no final ou no início do ditongo:

[k**wa**l] - "qual" [p**aj**] - "pai"

E podem ser orais ou nasais:

[p**aj**] - "pai" [p**õj**] - "põe"
[m**aw**] - "mau" [m**ɐ̃j**] - "mãe"
[b**oj**] - "boi" [m**ɐ̃w**] - "mão"

Estes exemplos foram todos escolhidos para ajudar a exemplificar a diferença entre ditongo e hiato. Se se reparar, todos estes ditongos correspondem a uma única sílaba, mas os hiatos formam duas sílabas. Observe-se os dois exemplos em comparação:

Ditongo	**Hiato**
[paj**ʃ**] - "pais"	[pa ' i**ʃ**] - "pa-ís"

Mas uma língua é um organismo vivo, e as pessoas dizem as coisas de modos diferentes consoante a situação em que se encontram - são estes pormenores que fazem uma língua evoluir e modificar-se mais depressa. Assim, um hiato pode passar a ditongo, se dito

muito depressa; e um ditongo pode passar a hiato se for dito pausadamente de modo a salientar bem todos os sons:

[l**u** ' **a**r] - "lu-ar"	-->	[l**wa**r] - "luar"
[s**aw** ' da ' də] - "sau-da-de"	-->	[s**a** ' **u** ' da' də] - "sa-u-da-de"

Por fim, os tritongos são formados por uma semivogal, uma vogal e outra semivogal, podendo ser orais ou nasais:

[urug**waj**] - "Uruguai"	[sag**w**$\tilde{\alpha}$ **w**] - "saguão"
[$\tilde{e}\int$ ag**w**α **j**] - "enxaguei"	[dəl$\tilde{i}$ k**w**$\tilde{\alpha}$ **j**] - "delinquem"

Encontros consonânticos

É o nome que se dá a um agrupamento de consoantes. Os agrupamentos mais comuns são aqueles em que a segunda consoante é "l" ou "r", embora em alguns casos não surjam no início da palavra:

bloco	a**bl**uir
---	a**tl**as
dragão	vi**dr**o
---	pala**vr**a

Outros agrupamentos são mais raros, como os que se seguem:

Gnomo	**mn**emónico
pneu	**ps**icológico
a**pt**o	di**gn**o

Nestes agrupamentos, as consoantes pertencem sempre a uma única sílaba. No entanto, quando se encontram no meio da palavra podem pertencer a duas sílabas. Por outro lado, por vezes a língua ao evoluir começa a "considerar" estes agrupamentos como "incómodos" e introduz uma vogal. Veja-se os exemplos abaixo:

a - **pt**o	di - **gn**o
a**p** - **t**o	di**g** - **n**o
a - **p***e* - to	di - **g***ui* - **n**o

Por fim, é preciso um pouco de atenção para não confundir consoantes com letras; evitando, assim, confundir os encontros consonantais com **dígrafos**. Ou seja, um encontro consonantal é um grupo de dois sons consonânticos - [pn] e [kl], por exemplo - enquanto um dígrafo é um grupo de duas letras que representam um som - "rr" representa o [R], por exemplo.

O mais importante a ter em mente relativamente aos encontros vocálicos e consonantais, é que a língua estabelece regras que impedem o "encontro" entre certos sons e em certas posições dentro de uma palavra.

As Sílabas

A sílaba é um som ou conjunto de sons que podem ser ditos numa só expiração. Ou seja, se se disser uma palavra devagar, ninguém dirá:

a - l - u - n - o

Afinal, isso seria soletrar. Qualquer pessoa dirá:

a - l u - n o

Portanto, vamos aos factos:

Uma sílaba forma-se tendo por base uma vogal ou ditongo (ou tritongo), podendo ser ou não rodeada de consoantes. Vejamos alguns exemplos:

i - na - cen - tu - a - dos;

trans - por;

As sílabas podem ser **abertas** ou **fechadas**:

a - ca - ma - do, as quatro sílabas são abertas pois acabam em vogais;

trans - por - tar, as três sílabas são fechadas pois acabam em consoantes;

Por fim, as palavras são classificadas quanto ao número de sílabas que as constituem:

as palavras **monossílabas** possuem apenas uma sílaba, como em **mão**

as palavras **dissílabas** possuem duas sílabas, como em **trans - por**

as palavras **trissílabas** possuem três sílabas, como em **trans - por - tar**

as palavras **polissílabas** possuem mais de três sílabas, como em **a - ca - ma - do**, **i - na - cen - tu - a - dos** ou em **o - to - rri - no - la - rin - go - lo - gis - ta**

A noção de sílaba pode parecer pouco importante, mas é uma base essencial para se compreender muitas noções de que falaremos mais à frente, desde a acentuação das palavras à sintaxe e morfologia.

Acento Tónico e Outros Tipos de Acento

As palavras são, portanto, constituídas por sílabas, mas estas não têm todas o mesmo valor. O que quero dizer? Tomemos a palavra "árvore" como exemplo, dizemos:

ár - vo - re

e não:

ar - **vó** - re ou ar - vo - **ré**

Ou seja, **acentuamos** a primeira sílaba enquanto as restantes não são acentuadas. esta acentuação faz-se por meio de uma maior ou menor utilização de certas características do nosso aparelho fonador:

Intensidade - a força com que uma sílaba é expirada

Os sons podem ser **fortes** (ou seja, tónica)

Os sons podem ser **fracos** (ou seja, átona)

Tom - (também chamado de altura musical) a frequência com que as cordas vocais vibram na altura em que se produz uma sílaba

Os sons podem ser **agudos** (ou seja, altos)

Os sons podem ser **graves** (ou seja, baixos)

Timbre - (também chamado de metal da voz) o tom principal, dos sons que são produzidos, ressoa nas cavidades (bucal e nasal) por onde passa o ar expirado (a posição da língua e a abertura da boca são factores importantes) produzindo tons secundários - é ao conjunto destes tons (o principal e os secundários) que se dá o nome de timbre

Os sons podem ser **abertos**

Os sons podem ser **fechados**

Quantidade - a duração com que os sons são emitidos

Os sons podem ser **longos**

Os sons podem ser **breves**

Esta classificação, claro, está incompleta: uma língua pode fazer a distinção de até 4 (Mandarim) ou 8 tons (Cantonês), por exemplo. Do mesmo modo, a quantidade longa ou breve pode não ter valor distintivo, como acontece nas vogais portuguesas em que a quantidade surge apenas no âmbito de acentuações de insistência ou ênfase.

Tom (exemplo do mandarim)	
fan (nível elevado) -	"vela"
fan (a subir) -	"embaraço"

Quantidade (exemplo do português)	
sim (breve) -	afirmação positiva

fan (a descer) -	"virar"	sim (longo) -	afirmação positiva (feita com enfado)
fan (queda) -	"arroz"		

Mas, pode agora alguém dizer, o que foi dito anteriormente sobre uma sílaba acentuada numa palavra não está correcto, pois algumas palavras apresentam mais do que uma sílaba acentuada. Claro que sim, do mesmo modo que essa acentuação pode mudar na mesma palavra consoante a frase em que está inserida, mas vamos com calma.

Como regra, todas as palavras possuem um **acento tónico**, ou seja, a sílaba é pronunciada de um modo forte; e **acentos átonos**, sílabas pronunciadas de modo fraco. Sendo as palavras classificadas consoante a sílaba em que recai o acento tónico:

palavras agudas ou oxítonas - quando o acento tónico se encontra na última sílaba (p. ex.: ca**fé**; fu**nil**)

palavras graves ou paroxítonas - quando o acento tónico se encontra na penúltima sílaba (p. ex.: es**co**la; di**ton**go)

palavras esdrúxulas ou proparoxítonas - quando o acento tónico se encontra na antepenúltima sílaba (p. ex.: **lâ**mina; qui**ló**metro)

palavras bisesdrúxulas - quando uma palavra é combinada com certos monossílabos àtonos o acento pode recuar (p. ex.: estu**dá**mo-lo; **fa**ça-se-lhe)

monossílabos átonos - a única sílaba é pronunciada tão fracamente que se une a uma outra palavra, utilizando o acento tónico dessa palavra como apoio (p. ex.: diga-**me**; **o** carro)

monossílabos tónicos - a única sílaba é pronunciada de um modo forte (p. ex.: **flor**; **sim**)

Em alguns casos, a variação do acento tónico pode inclusivé marcar a diferença entre uma palavra e outra, um significado e outro, por exemplo: "**dú**vida" e "du**vi**da", em que no primeiro caso temos o substantivo e no segundo temos o verbo duvidar (tanto pode ser a forma imperativa como o presente do Indicativo). Isto, no entanto, é mais comum na variante do português falado no Brasil pois, em Portugal, as vogais átonas tendem a sofrer um enfraquecimento do timbre, note-se (a sílaba tónica é identificada por um apóstrofo imediatamente antes - p.ex.: di**á**logo = [di'alugu]):

Português do Brasil		**Português de Portugal**	
correram	[ko'Rer ɐ̃ w]	correram	[ku'Rer ɐ̃ w]
correrão	[koRe'r ɐ̃ w]	correrão	[kuRɘ'r ɐ̃ w]

Como se referiu acima, as palavras não são compostas apenas por uma sílaba tónica rodeada das restantes átonas - especialmente as palavras longas, com atenção para as palavras derivadas, possuem um ou mais acentos que se encontram entre os "sons

fracos" e os "sons fortes". São as **sílabas subtónicas**, demasiado fortes para serem àtonas, e não o suficiente para serem tónicas. Assim, repare-se:

deci**di**da

Esta palavra possui um acento tónico na penúltima sílaba, que é, regra geral, a norma da acentuação tónica no português.

deci***di***da**men**te

Aqui, a palavra mantém o acento tónico na penúltima sílaba, mas permanece uma reminescência da acentuação da palavra original num acento subtónico (em itálico).

Para além destes três acentos - tónico, subtónico, átono - existem ainda os **acentos de insistência**, cuja função é a de realçar certa palavra de acordo com o contexto. São os acentos afectivo e intelectual.

O **acento afectivo** tem um carácter emocional, pois o acento é utilizado quando pretendemos demonstrar a nossa relação afectiva com uma dada situação:

Ele é um ***mi***se**rá**vel! (sentimento de cólera ou desprezo)

Isto é a***bo***mi**ná**vel! (sentimento de cólera ou repulsa)

Olha que am***or***! (sentimento de afecto)

A palavra passa a ter duas sílabas acentuadas (em palavras pequenas o acento afectivo coincide com o acento tónico), sendo que o acento afectivo é quase tão forte como o acento tónico, podendo mesmo ultrapassá-lo. O mesmo efeito acontece com o **acento intelectual**, mas o recurso a este acento tem como função realçar uma noção ou caracterização:

Eu quero razões ***o***bjec**ti**vas!

Isto não é ***i***mo**ral**, é ***a***mo**ral**!

Não quero razões ***sub***bjec**ti**vas!

Então, em termos sonoros, qual a diferença entre o acento afectivo e o acento intelectual?

Acento Intelectual	**Acento Afectivo**
O acento intelectual coincide sempre com a primeira sílaba, seja a palavra iniciada por consoante ou vogal.	O acento afectivo coincide com a primeira sílaba se a palavra for iniciada por consoante.
	Coincide com a segunda sílaba se a palavra for iniciada por vogal.
	Coincide com o acento tónico se for

O acento intelectual aumenta a vogal em duração, altura e sobretudo em **intensidade**.	uma palavra pequena. O acento afectivo aumenta a vogal em intensidade, mas sobretudo em **duração** e **altura**.

No entanto, como também já foi dito, a acentuação nas palavras pode ser modificada quando estas estão integradas numa frase. Afinal, ao dizermos uma frase estamos a articular e, por vezes, a fundir as palavras umas nas outras, sendo que a frase pode ser dividida em **grupos acentuais** ou de **intensidade**, cada qual apoiado num acento tónico. Vejamos um exemplo:

/ **Di**as / e **no**ites / os hori**zon**tes / se re**pe**tem. /

Esta frase, quando dita pausadamente, é composta por quatro grupos acentuais, quase todos unindo uma palavra a um monossílabo átono. Mas, se dissermos os dois primeiros grupos depressa, estes fundem-se num só.

/ ***Di***as e **no**ites /

Aqui, o primeiro grupo acentual vê o seu centro tónico enfraquecer para um acento secundário, um subtónico. Assim, este tem de se integrar no grupo seguinte.

A estas situações de dependência chama-se **ênclise** (quando a primeira palavra mantém o acento tónico em detrimento da palavra seguinte, p. ex.: diga-**me**) e **próclise** (quando a primeira palavra perde o acento tónico tornando-se dependente da palavra seguinte, p. ex.: **os** horizontes). Esta perda de independência pode, inclusivé, resultar em alterações às palavras dependentes, que se vêem reduzidas. Observe-se alguns exemplos tão conhecidos da nossa oralidade:

Ele foi de férias **para as** Maldivas. - Ele foi de férias **prás** Maldivas.

Olha! É o **senhor António**! - Olha! É o **seu António**.

Esta é uma forma da língua evoluir em termos de vocabulário. Em português, são muitas as palavras que surgiram por próclise:

cento	-	cem
grande	-	grão
quanto	-	quão
santo	-	são
tanto - tão		

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo