FERRAMENTAS COMPUTACIONAIS PARA A MÚSICA

Eduardo Reck Miranda*

1. Breve histórico e introdução

Desde a sua criação em 1988, o Núcleo de Música Eletrônica do Instituto de Artes da UFRGS vem se preocupando com o desenvolvimento de tecnologia própria em software (programas) e eventualmente em hardware (circuitos eletrônicos).

No que tange ao hardware, com o apoio da equipe do setor de Desenvolvimento de Hardware do Centro de Processamento de Dados (CPD) da UFRGS, desenvolveu-se neste período, menos de dois anos, duas interfaces MIDI: uma para microcomputadores da linha Apple a outra para os da linha IBM-PC XT. Em resumo interface MIDI é um circuito eletrônico responsável pela comunicação de dados MIDI entre um computador e um sintetizador digital. Dados MIDI são dados que obedecem a uma padronização internacional adotada pelos fabricantes de sintetizadores digitais comerciais que os torna compatíveis entre as diferentes marcas lançadas no mercado bem como controláveis via computador externo.

Desde então os programas confeccionados para a nossa configuração de hardware, limitada a um sintetizador digital simples e um microcomputador, sempre foram de ordem específica. Ou seja, cada programa é específico para uma determinada composição ou experiência. Embora restrita a compositores peritos em programação de máquina, em um nível muito baixo, com uma linguagem que não foi pensada para a música, esta prática permitiu a aquisição de um know-how pertinente ao âmago do funcionamento do que tínhamos em mãos. Os "programa-composições" foram codificados em linguagem C e alguns trabalhos foram divulgados

*Aluno do Mestrado em Ciência da Computação da UFRGS. Participou da criação, em 1988, do Núcleo de Música Eletrônica do Instituto de Artes da UFRGS, onde trabalha com pesquisa na área de música computacional.

em concerto no Festival de Música Nova 89, no MASP, em São Paulo, no III Encontro de Música Contemporânea em Santiago do Chile e no II Encontro dos Compositores da Região Sul em Porto Alegre.

Entretanto, com o objetivo de permitir o acesso a um número maior de pessoas interessadas em trabalhar conosco, estamos direcionando nossos esforços à confecção de programas que elevem o nível da linguagem de programação.

Sabe-se que existem inúmeros softwares no mercado internacional. Porém é do interesse da universidade o desenvolvimento de suas próprias ferramentas de trabalho. Entendemos que as possibilidades de utilização do computador, na música, ultrapassam os limites impostos pela grande maioria destes softwares, que muitas vezes não superam sequer o conceito do seqüenciador e do play-back, como se fossem versões sofisticadas dos cilindros de pianola.

No primeiro semestre de 1989, juntamente com o pessoal ligado à área de Inteligência Artificial do Pós-Graduação em Ciência da Computação, deu-se início à pesquisa no sentido de estudar as possibilidades das ferramentas de IA em aplicações musicais. Estamos desenvolvendo uma série de ferramentas, para manipulação de idéias sonoras, que no conjunto formarão um ambiente propício à confecção de programas, para a música em um nível de linguagem bastante alto. Assim, não se exigirá do musicista o conhecimento profundo de técnicas de programação de máquina. Basta que este esteja familiarizado com o ambiente de trabalho proporcionado pelas ferramentas.

A seguir serão demonstradas quatro ferramentas codificadas em linguagem C. De agora em diante convencionaremos que o termo "ferramenta", como ele vem sendo utilizado até o momento, será substituído pelo termo "função", para fins de adequação do discurso ao vocabulário técnico das linguagens de programação.

A primeira é uma função geradora de nota transposta dado um intervalo e uma nota de referência. A segunda constrói uma tabela das possíveis inversões da forma no sentido vertical de um fenômeno de quatro notas, dados os elementos que o compõe. A terceira define o grau de abertura do mesmo. A quarta, em um nível mais alto, se vale das três anteriores para construir um fenômeno musical composto de quatro alturas a partir de uma nota eixo e dados referentes a sua forma vertical.

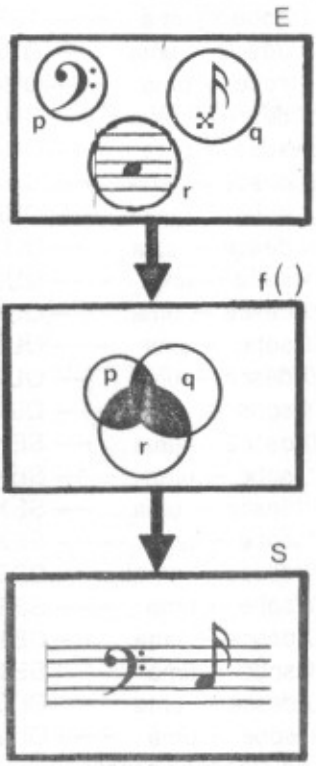
2. Funções

Função, no sentido em que estamos empregando o termo, é um

mecanismo computacional que quando devidamente alimentado produz um resultado. Toda função possui três componentes abstratos distintos (fig. 1):

a. parâmetro de entrada — que poderá ser mais de um, ou nenhum;
b. corpo de processamento;
c. resultado, ou retorno — que poderá ser explicitado diretamente pela emissão de um parâmetro de saída, ou conseqüência implícita do trabalho realizado pela função.

(figura 1)



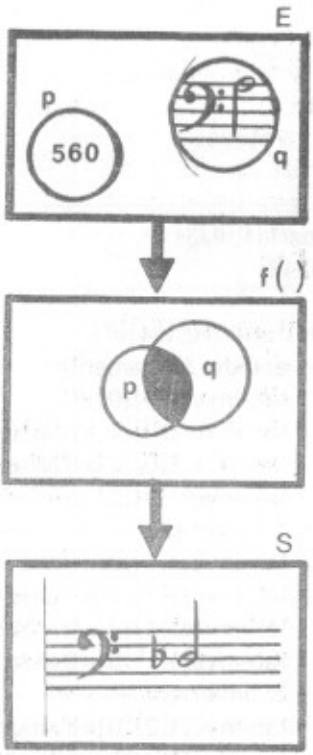
2.1. Função transp.

```
000 /* ---------------------------------------- /*
026 transp(nota,itvl)
027 int nota,itvl;
028 {int nota-transposta = NULO;
029  switch(itvl)
030  {case 200:desce – uma   -- SEGUNDA-MENOR;break;
031   case 201:sobe – uma   --- SEGUNDA-MENOR;break;
032   case 210:desce – uma   -- SEGUNDA-MAIOR;break;
033   case 211:sobe – uma   --- SEGUNDA-MAIOR;break;
034   case 300:desce – uma   -- TERÇA-MENOR;break;
035   case 301:sobe – uma   --- TERÇA-MENOR;break;
036   case 310:desce – uma   -- TERÇA-MAIOR;break;
037   case 311:sobe – uma   --- TERÇA-MAIOR;break;
038   case 440:desce – uma   -- QUARTA-DIMINUTA;break;
039   case 441:sobe – uma   --- QUARTA-DIMINUTA;break;
040   case 450:desce – uma   -- QUARTA-JUSTA;break;
041   case 451:sobe – uma   --- QUARTA-JUSTA;break;
042   case 460:desce – uma   -- QUARTA-AUMENTADA;break;
043   case 461:sobe – uma   --- QUARTA-AUMENTADA;break;
044   case 540:desce – uma   -- QUINTA-DIMINUTA;break;
045   case 541:sobe – uma   --- QUINTA-DIMINUTA;break;
046   case 550:desce – uma   -- QUINTA-JUSTA;break;
047   case 551:sobe – uma   --- QUINTA-JUSTA;break;
048   case 560:desce – uma   -- QUINTA-AUMENTADA;break;
049   case 561:sobe – uma   --- QUINTA-AUMENTADA;break;
050   case 600:desce – uma   -- SEXTA-MENOR;break;
060   case 601:sobe – uma   --- SEXTA-MENOR;break;
061   case 610:desce – uma   -- SEXTA-MAIOR;break;
062   case 611:sobe – uma   --- SEXTA-MAIOR;break;
063   case 700:desce – uma   -- SETIMA-MENOR;break;
064   case 701:sobe – uma   --- SETIMA-MENOR;break;
065   case 710:desce – uma   -- SETIMA-MAIOR;break;
066   case 711:sobe – uma   --- SETIMA-MAIOR;break;
067   case 850:desce – uma   -- OITAVA;break;
068   case 851:sobe – uma   --- OITAVA;break;
069   case 0:não-transpoe;break;
070 return(nota-transposta);
071 }
072 /* ---------------------------------------- */
```

(listagem 1)

Esta função gera uma nota a partir de uma nota de referência dado um intervalo (listagem 1).

(figura 2)



Os parâmetros de entrada desta função são a nota de referência e o intervalo. A saída é a nota desejada. Para informar o parâmetro intervalo foi criada uma sintaxe que otimiza a padronização da informação. Trata-se de um código de três dígitos onde:

a. o primeiro dígito indica o intervalo propriamente dito e poderá assumir respectivamente os valores: {0, 2, 3, 4, 5, 6, 7 e 8};
b. o segundo dígito indica a classificação do intervalo onde:
   0 → intervalo menor;
   1 → intervalo maior;
   5 → intervalo justo;
   4 → intervalo diminuto;
   6 → intervalo aumentado.
c. o terceiro dígito indica o sentido do intervalo, onde:
   1 → sentido ascendente;
   0 → sentido descendente.

Se, por exemplo, (fig. 2) utilizarmos o código 560 estaremos informando que a função deve gerar uma nota com uma distância de quinta aumentada descedente, em relação à nota de referência.

Caso se fizer necessário gerar uma nota cuja distância ultrapasse o âmbito da oitava, basta utilizar a função de modo recursivo.

2.2. Função inversoes

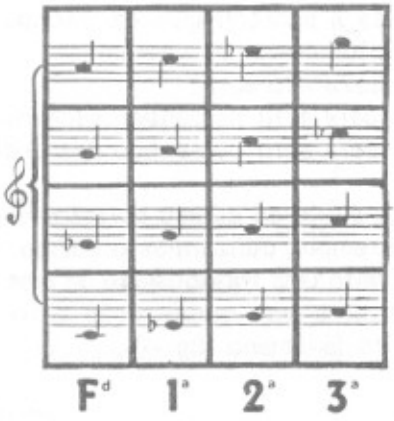
```
100 /* ------------------------------------------ /*
101 inversoes(f,i1,i2,i3)
102 int f,i1,i2,i3;
103 {
104 extern int ab-invert[4][4];
105 /* ----- estado fundamental ---------------------- */
106         tab-invert[0][0]=f;
107         tab-invert[0][1]=transp(f,i1);
108         tab-invert[0][2]=transp(f,i2);
109         tab-invert[0][3]=transp(f,i3);
110 /* ----- 1.ª inversao ---------------------------- */
111         tab-invert[1][0]=transp(f,i1);
112         tab-invert[1][1]=transp(f,i2);
113         tab-invert[1][2]=transp(f,i3);
114         tab-invert[1][3]=transp(f,851);
115 /* ----- 2.ª inversao ---------------------------- */
116         tab-invert[2][0]=transp(f,i2);
117         tab-invert[2][1]=transp(f,i3);
118         tab-invert[2][2]=transp(f,851);
119         tab-invert[2][3]=transp(transp(f,i1),851);
120 /* ----- 3.ª inversao ---------------------------- */
121         tab-invert[3][0]=transp(transp(f,i3),850);
122         tab-invert[3][1]=f;
123         tab-invert[3][2]=transp(f,i1);
124         tab-invert[3][3]=transp(f,i2);          }
125 /* ------------------------------------------ */
```

(listagem 2)

Esta função monta uma tabela (fig. 3), com as possíveis inversões da forma no sentido vertical de um fenômeno musical de quatro alturas, dado uma nota eixo e as outras três notas subseqüentes (listagem 2).

Empregamos o conceito de "inversão vertical da forma do fenômeno musical" no mesmo sentido que ele é empregado quando se trata de inversão de acordes de quatro sons na teoria tradicional da música. Ou seja, existem quatro possibilidades: o estado fundamental – que mantém a forma original –, a primeira, segunda e terceira inversões.

(figura 3)



2.3. Função abertura

```
200 /* ---------------------------------------- */
201 abertura(ordem,codigo)
202 int ordem;
203 int codigo[4];
204 {   int intervalo;
205     switch(codigo[ordem])
206     {       case 0:intervalo=0;break;
207             case 1:intervalo=851;break;
208             case 2:intervalo=850;break;     }
209     return(intervalo);
210 }
211 /* ---------------------------------------- */
```

(listagem 3)

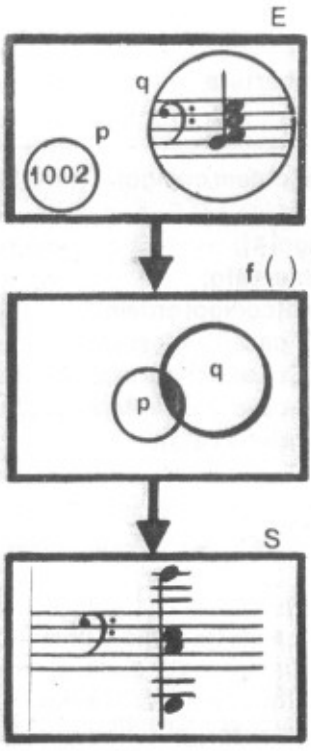
A função **abertura** não é uma função que goza da mesma autonomia das demais (listagem 3). Esta é uma função que depende diretamente das funções **inversoes** e **transp**, e seu resultado interessa somente à função **construtor,** que analisaremos no item 2.4.

Os parâmetros informados na função **abertura** são dois: o fenômeno em uma das quatro formas oferecidas pela tabela de inversões – gerada pela função **inversoes** – e o tipo de abertura vertical desejada. Na verdade a abertura não passa da aplicação do conceito de transposição de oitava, ascendente ou descendente, a uma ou mais notas do fenômeno. Para informar o parâmetro tipo de abertura foi criado uma sintaxe que otimiza a padronização da informação. Trata-se de um código de quatro dígitos que podem assumir:

0 → para não abrir;
1 → para abrir no sentido ascendente;
2 → para abrir no sentido descendente.

Cada dígito do código corresponde a uma das notas do fenômeno. Se, por exemplo, utilizarmos o código **1002** estaremos informando que deve ser feito uma transposição de uma oitava descendente da primeira nota inferior e uma transposição de oitava ascendente da última nota superior do fenômeno (fig. 4).

(figura 4)



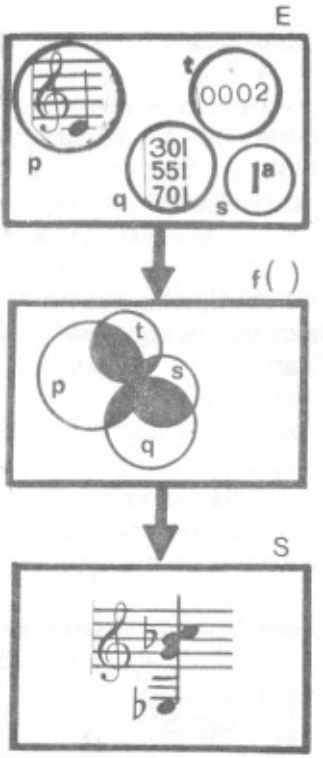
```
300 /* ------------------------------------------------ */
301 construtor(fund,itv1,itv2,itv3,ab1,ab2,ab3,ab4,inv)
302 int fund,itv1,itv2,itv3;
303 int ab1,ab2,ab3,ab4;
304 int inv;
305 {
306 extern int ab[4],notas-evento[4];
307 /* monta codigo de abertura ---------- */
308    ab[3]=ab1;
309    ab[2]=ab2;
310    ab[1]=ab3;
311    ab[0]=ab4;
312 /* monte tabela das inversoes --------- */
313   inversoes(fund,itv1,itv2,itv3);
314 /* monta array com a construcao do evento --------- */
315    for(x=0;x<4;x++)
316    notas-evento[x]=(transp(tab-invert[inv][x],          /
                          abertura(x,ab)));
317 return(SUCCESS);
318 {
319 /* ------------------------------------------------ */
```

(listagem 4)

A função **construtor** gera um fenômeno de quatro sons a partir de uma nota eixo, dados os intervalos a serem observados para a sua construção, o tipo de abertura e o grau de inversão (listagem 4). Esta função é uma função que reúne, além de seu próprio mecanismo, as três funções discutidas nos itens anteriores. Na verdade o que ela faz é montar um "array" com os elementos do fenômeno a partir da tabela das inversões – ver função **inversoes** – e prepará-las para sua utilização (fig. 5).

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
CURSO DE PÓS GRADUAÇÃO MESTRADO EM MÚSICA
Av. Borges de Medeiros, 308 - Conj. 123
90020 - Porto Alegre - RS
FONE: (0512) 26 87 72

(figura 5)



3. Perspectivas e observações

3.1. Não foi tratado em nenhum momento nas definições anteriores o problema da utilização do fenômeno, ou seja, a sua emissão. Propositadamente, a função **construtor** limita-se apenas a uma preparação do fenômeno em um "array" para posterior utilização. Dividiu-se a questão do processamento do fenômeno musical em dois níveis: um primeiro nível trata da construção vertical do fenômeno, e um segundo nível — talvez mais alto — trata da emissão do fenômeno em uma linha de tempo, cuida da forma horizontal. É possível se pensar em uma função que, dado o "array" gerado pela **construtor** e valores temporais de pausa para disparo e duração individuais, emita um fenômeno musical que poderá assumir qualquer um dos "patterns" possíveis de se manifestar no domínio destes elementos.

3.2. É plausível se pensar em uma generalização da função **construtor** para que o número de componentes de um fenômeno não fique limitado a apenas quatro sons, de modo que este valor possa ser assumido por uma variável que responda às necessidades de aplicação.

REFERÊNCIAS BIBLIOGRÁFICAS

CONGER, Jim. *C Programming for MIDI*. California: M&T Publishing, 1988.

FURIA, Steve de. *The MIDI boom*. New Jersey: Third Earth Productions, 1986.

KERNIGAN, B. W. & RITCHIE, D. M. *A Linguagem de programação C*. Trad. EDISA Ltda. 4.ed. Rio de Janeiro: Campus, 1988.

LACERDA, Osvaldo. *Teoria Elementar da música*. 6.ed. São Paulo: Ricordi Brasileira, 1967.

WAITE, Mitchell et al. *C Primer Plus*. 5.ed. Indiana: Howard W. Sans, 1988.