Universidade Federal da Paraíba

Centro de Informática

Programa de Pós-Graduação em Computação, Comunicação e Artes

LUCAS BLATT

TÍTULO:

INTRODUÇÃO A PROGRAMAÇÃO A PARTIR DE CONHECIMENTOS SOBRE MÚSICA

João Pessoa

Abril / 2019

Lucas Blatt

Título: Introdução a programação a partir de conhecimentos sobre música

Disseração apresentada ao Programa de Pós-Graduação em Computação, Comunicação e Artes (PPGCCA) da Universidade Federal da Paraíba, como requisito parcial para a obtenção do título de Mestre em Computação, Comunicação e Artes, na linha de pesquisa Arte Computacional.

Orientador: Prof. Dr. Alexandre Magno e Silva Ferreira

João Pessoa

Abril / 2019

INTRODUÇÃO A PROGRAMAÇÃO A PARTIR DE CONHECIMENTOS SOBRE MÚSICA / Lucas Blatt. - João Pessoa, 2019.
171 f. : il.

Dissertação (Mestrado) - UFPB/Informática.

1. Programação de Computadores. 2. Aprendizagem Significativa. 3. Música. 4. Frescobaldi. I. Título

UFPB/BC

Lucas Blatt

INTRODUÇÃO A PROGRAMAÇÃO A PARTIR DE CONHECIMENTOS SOBRE MÚSICA

Dissertação apresentada ao Programa de Pós-Graduação em Computação, Comunicação e Artes da Universidade Federal da Paraíba, como requisito parcial para a obtenção do título de Mestre em Computação, Comunicação e Artes, na linha de pesquisa Arte Computacional.

A banca considera o presente Trabalho Final: APROVADO

Data: 29 / ABRIL / 2019.

Prof. Dr. Alexandre Magno e Silva Ferreira (Orientador – PPGCCA/UFPB)

Prof. Dr. Cleisson de Castro Melo (membro externo - instituição).

Prof. Dr. Carlos Eduardo Coelho Freire Batista (PPGCCA/UFPB)

Prof. Dr Fábio Henrique Ribeiro (membro externo - programa)

Dedico este trabalho a minha família,

Tamires e Yasmin, pelos momentos

de ausência e pelo incentivo

para essa pesquisa.

AGRADECIMENTOS

A **Deus**, pela dádiva da vida e por me permitir realizar tantos sonhos nesta existência. Obrigado por me permitir errar, aprender e crescer, por Sua eterna compreensão e tolerância, por Seu infinito amor, pela Sua voz "invisível" que não me permitiu desistir e principalmente por ter me dado uma família tão especial, enfim, obrigado por tudo.

Ao **Prof. Alexandre**, pela orientação, competência, profissionalismo e dedicação tão importantes. Tantas vezes que nos reunimos e, embora em algumas eu chegasse desestimulado, bastavam alguns minutos de conversa e umas poucas palavras de incentivo e lá estava eu, com o mesmo ânimo do primeiro dia de aula. Obrigado por acreditar em mim e pelos tantos elogios e incentivos. Tenho certeza que não chegaria neste ponto sem o seu apoio. Você foi e está sendo muito mais que orientador: para mim será sempre mestre e amigo.

Aos membros da banca examinadora, **Prof. Fábio, Prof. Carlos** e **Prof. Cleisson**, que tão gentilmente aceitaram participar e colaborar com esta dissertação.

À minha **família**, por apoiarem e compreenderem o meu isolamento em inúmeras tardes de domingo.

À minha **mãe** e ao meu **pai** deixo um agradecimento especial, por todas as lições de amor, companheirismo, amizade, caridade, dedicação, abnegação, compreensão e perdão que vocês me dão a cada novo dia. Sinto-me orgulhoso e privilegiado por ter pais tão especiais.

À minha amada esposa **Tamires**, por todo amor, carinho, compreensão e apoio em tantos momentos difíceis desta caminhada. Obrigado por permanecer ao meu lado, mesmo sem os carinhos rotineiros, sem a atenção devida e depois de tantos momentos de lazer perdidos. Obrigado pelo presente de cada dia, pelo seu sorriso e por saber me fazer feliz.

À minha princesa **Yasmin**, por todo amor incondicional que você sempre me deu. A sua existência é o reflexo mais perfeito da existência de Deus.

Por fim, a todos aqueles que contribuíram, direta ou indiretamente, para a realização desta dissertação, o meu sincero agradecimento.

RESUMO

O computador tem sido considerado uma ferramenta extremamente atraente para educação musical bem como para criar e manipular sons. Sua precisão, possibilidades de novos timbres e potencial para automação fantástica fazem dele uma plataforma atraente para experimentar e fazer música, mas apenas na medida em que podemos realmente dizer a um computador o que fazer e como fazê-lo. Para isso precisamos aprender como programar um computador. Com isso, o objetivo deste trabalho é introduzir práticas com o software livre Frescobaldi como ferramenta de apoio ao ensino de linguagem de programação para músicos através de uma metodologia baseada na Teoria da Aprendizagem Significativa, que leve em consideração o conhecimento prévio de música que possa ser relacionado com computação. Desta forma, buscou-se um referencial sobre o ensino de programação e a aprendizagem significativa, seguido da relação da música com a tecnologia. A partir da pesquisa ação, elaborou-se uma proposta de ensino que foi implementada com um grupo de estudantes de música, coletando-se as informações necessárias para verificar a validade da proposta, que tem como objetivo ensinar programação para músicos.

Palavras-chave: Programação de Computadores, Aprendizagem Significativa, Música, Frescobaldi

ABSTRACT

The computer has been considered an extremely attractive tool for musical education as well as for creating and manipulating sounds. Its accuracy, possibilities for new timbres and the potential for automation make it an attractive platform for experimentation in music making. But to a certain extent, we need to know to instruct it to perform and how to do it. Based on this, we need to learn how to program a computer. In the courses of programming languages, the difficulty that many students face while learning such subject, mostly because to master this skill as a whole, a great deal of abstraction and conceptual knowledge is necessary. The activities in the classroom and on the notebook or even on the desktop become uninteresting because they do not allow the student to visualize in the real world its output. In this way, a reference was made on programming teaching and meaningful learning, followed by the relationship between music and technology. From the action research, a teaching proposal was developed that was implemented with a group of music students, gathering the necessary information to verify the validity of the proposal, which aims to teach programming for musicians.

Keywords: Computer Programming, Significant Learning, Music, Frescobaldi

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE GRÁFICOS

SUMÁRIO

1 INTRODUÇÃO

O computador tem sido considerado por pesquisadores, músicos e educadores, uma ferramenta extremamente atraente para criar e manipular sons, ferramenta de apoio metodológico na educação musical, além de meio indispensável na produção e distribuição da música. Sua precisão, possibilidades para o desenvolvimento de novos timbres e potencial para automação sonora fazem dele uma plataforma atraente para experimentar e fazer música. Esse processo só é possível quando podemos dizer a um computador o que e como fazer. Para que o computador possa realizar as atividades solicitadas por um usuário, são desenvolvidos programas com finalidades específicas para uma determinada função. Desta forma, um programa é uma sequência de instruções para um computador e uma linguagem de programação é uma coleção de regras sintáticas e semânticas para especificar essas instruções e, eventualmente, fornecer a tradução de programas escritos por humanos às instruções correspondentes que computadores executam.

Na história da computação, muitos foram os projetos para instruir computadores, entre elas estão as linguagens de programação. Ao contrário da maioria das outras classes de interfaces homem-computador, as linguagens de programação não executam diretamente nenhuma tarefa específica (como processamento de texto ou edição de vídeo), mas nos permitem criar softwares que podem executar praticamente qualquer função personalizada. A linguagem de programação atua como um mediador entre a intenção humana e as instruções correspondentes que fazem sentido para um computador. Os programas existem em muitos níveis, desde o código assembler (extremamente baixo nível) até linguagens de script de alto nível que frequentemente incorporam estruturas mais legíveis, como aquelas que se assemelham a linguagens faladas ou representação gráfica de objetos familiares. Portanto, o computador enquanto máquina, com seus componentes (hardware) é uma ferramenta genérica que pode ser configurado ou programado (software) para demandas e atividades específicas.

Desta forma, com o advento da tecnologia e aplicabilidade nas mais diversas áreas do conhecimento, o ensino da programação de computadores é um tema estudado e discutido amplamente devido as diversas formas de se abordar o aspecto da aprendizagem nesta área. A importância do ensino da programação para novos estudantes dessa área não está somente no

conteúdo abordado, mas também na capacidade de transpor alguns conceitos conhecidos de outras áreas para uma linguagem de programação (SALGADO, 2013).

É possível observar que o processo de ensino/aprendizagem de programação tem gerado desistência, abandono de cursos, grandes índices de reprovação, cenários esse, complexo para estudantes e professores. A literatura aponta diversos fatores para essas ocorrências, como falta de competências na resolução de problemas, poucas habilidades matemáticas, baixo nível de abstração, dificuldades de interpretação do problema e compreensão de texto por parte dos estudantes. Diversas são as maneiras ou estratégias que podem ser adotadas no ensino de programação em cursos superiores nas mais diversas áreas, sendo que cada vez mais, professores buscam maneiras motivadoras para transmitir para os alunos os conceitos que envolvem as técnicas de programação.

Segundo Greening (2000), a estratégia pedagógica mais comum de ensino de programação se constitui em aulas teóricas que são complementadas por laboratórios onde os alunos reproduzem as aulas, através de uma linguagem de programação. Embora esta seja uma estratégia importante dentro do processo de ensino da programação ela não deve ser isolada, pois pode acabar por gerar um processo de reprodução daquilo que foi transmitido em sala de aula em detrimento com a real construção do conhecimento.

Ainda segundo Greening (2000), as principais dificuldades encontradas durante o processo de aprendizagem são reflexos de diversos problemas como lógica de programação e a não familiaridade com um determinado paradigma. É fundamental haver, durante o processo de aplicação dos conceitos, um maior foco na solução do problema levando em consideração a vivência e os conhecimentos trazidos previamente pelos estudantes. Segundo Zaina (2005), as diretrizes que regem o ensino da programação interligadas com as necessidades de formação profissional tornam claro que a interdisciplinaridade entre diferentes conteúdos que compõem o eixo do ensino de programação, são de maneira geral, uma forma de mostrar ao aluno como diversos conteúdos de disciplinas diferentes podem ser integrados para a solução de um problema.

Desta forma, considerando o panorama sobre dificuldades e propostas para superar essas barreiras apontadas pela literatura, propõe-se um trabalho que tem aderência a pesquisas sobre processos metodológicos e estratégias de ensino. Além disso, buscou-se um alicerce na

Teoria de Aprendizagem Significativa elaborada por David Ausubel, propondo uma abordagem significativa na apresentação dos conteúdos de programação aos estudantes.

Na Teoria da Aprendizagem Significativa considera-se que o fator mais importante para que a aprendizagem ocorra, é verificar aquilo que o estudante já sabe ou conhece, para que estes conhecimentos prévios sirvam como ponto de ancoragem para os novos conhecimentos a serem adquiridos. Ao considerarmos que para músicos iniciantes em computação, a programação de computadores é um assunto completamente novo, a ausência de conhecimentos prévios que possam ser relacionados pode tornar o processo de ensino/aprendizagem mais delicado. Assim, ao se trabalhar com um assunto desconhecido é importante resgatar alguma referência que sirva de ponto de ancoragem para os novos conhecimentos.

Desta forma, na pesquisa propomos a utilização do conhecimento prévio sobre música como ponto de partida e o software Frescobaldi como ferramenta de apoio ao ensino de programação para músicos.

1.1 Problema de Pesquisa

As rápidas mudanças provocadas pela tecnologia levaram a novas formas de se comunicar e novos modos de produção. Com larga aplicação de tecnologia em diversos segmentos, qualquer profissional poderá ter uma forte interação entre softwares e dispositivos eletrônicos que darão um ganho de produtividade e eficiência. Desta forma, as habilidades desenvolvidas ao aprender programação são úteis para qualquer profissional ao planejar o uso de sistemas computacionais ou ferramentas em sua área de atuação.

Sendo assim, o cenário que orientou esse trabalho é o de que uma proposta de ensino fundamentada na Aprendizagem Significativa possibilite a aprendizagem de programação para músicos. Partindo desta proposta, algumas das contribuições esperadas são facilitar o uso prático do computador na elaboração de soluções para demandas como a criação de *webpage* para divulgar seu trabalho, criação de aplicativos que auxiliem o professor em sala de aula, optimização de softwares para gravação, entre outros. Desta forma elaborou-se a seguinte questão norteadora para esta pesquisa:

Quais são as principais possibilidades de uso do software Frescobaldi através da aprendizagem significativa como ferramenta de ensino da programação de computadores para músicos?

1.2 Objetivos

A objetivo deste trabalho é introduzir práticas com o software livre Frescobaldi como ferramenta de apoio ao ensino de linguagem de programação para músicos através de uma metodologia baseada na Teoria da Aprendizagem Significativa, que leve em consideração o conhecimento prévio de música que possa ser relacionado com computação. Além disso, o trabalho apresenta os objetivos específicos:

- Identificar dificuldades e soluções propostas para aprendizagem de programação de computadores presentes na literatura;
- Compreender a relação entre música e tecnologia;
- Identificar as principais relações entre o conteúdo de programação para computadores e música;
- Compreender o funcionamento do software Frescobaldi
- Elaborar e implementar uma proposta baseada na aprendizagem significativa ;
- Analisar as contribuições desta proposta.

1.3 Justificativa

Qualquer indivíduo relacionado à música é invariavelmente requisitado a um envolvimento com a tecnologia, travando contato com um linguajar que se estende desde o mero manuseio de um aparelho de som até ao funcionamento de complexos processadores digitais de áudio. Sendo assim, considera-se de grande relevância o ensino de programação de computadores para músicos, proporcionando conhecimento que dará suporte na elaboração de soluções para os seus problemas, além de promover formas de aprendizagem mais eficientes e efetivas através da aplicação da tecnologia no ensino da música.

Este trabalho está inserido em um contexto interdisciplinar com abordagem educacional, computacional e musical. Desta forma, esta pesquisa contribui para o campo da computação, tendo como enfoque a informática na educação e ensino de música para programadores, contribuindo com trabalhos já desenvolvidos na área do ensino de

programação, bem como no campo da música, onde estão compreendidas a educação musical e composição.

1.4 Estrutura do trabalho

O presente trabalho está organizado da seguinte forma:

No Capítulo 2 apresentam-se as bases teóricas que subsidiam e fundamentam a pesquisa, partindo de definição sobre programação de computadores, passando pelo ensino de programação no Brasil com seus desafios e propostas de soluções dando enfoque as principais ferramentas de apoio utilizadas no ensino de programação. Também neste capitulo faremos uma abordagem histórica sobre a evolução da tecnologia e computação na música até a descrição da teoria de aprendizagem significativa que servirá de base a nossa proposta de ensino.

O Capítulo 3 descreve a metodologia utilizada para o desenvolvimento deste estudo, bem como os procedimentos metodológicos, referenciando os instrumentos de coleta de dados e as atividades que serão desenvolvidas.

Já no Capítulo 4 é descrito a ferramenta Frescobaldi que será utilizada durante a aplicação da proposta desenvolvida neste trabalho bem como as principais relações entre o conteúdo de programação e de música.

No Capitulo 5 serão descritos os dados coletados junto com a discussão desses dados.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta os pressupostos teóricos que norteiam este trabalho. Inicialmente são descritos alguns conceitos de programação de computadores juntamente com algumas dificuldades apresentadas no processo de ensino/aprendizagem de introdução a programação e as soluções propostas com enfoque nas ferramentas que são utilizadas como apoio neste processo encontradas na literatura.

No segundo momento será exposto um breve histórico da relação entre música e tecnologia computacional, desde as primeiras maquinas capazes de transformar meio físico (onda sonora) em pulso elétrico, passando pelos experimentos com softwares para a sintetização de sons, até os mais modernos equipamentos de gravação, manipulação sonora e distribuição da música.

Por fim, apresentamos a teoria da aprendizagem significativa de David Ausubel que servirá como ponto de apoio a esta proposta e como ela poderia contribuir no ensino/aprendizagem de programação de computadores para músicos.

2.1 A programação de computadores e o processo de ensino/aprendizagem

Os computadores são considerados como uma ferramenta indispensável a vida moderna, alterando significativamente o modo de vida das pessoas. Essas mudanças acontecem em razão de desenvolver-se maquinas, ferramentas e aplicações para as mais diversas demandas da humanidade, com o auxílio da programação desses dispositivos para as mais diversas finalidades.

Na subseção abaixo será apresentada a linha central do que é programação de computadores e como ela acontece.

2.1.1 O que é programação de computadores

Hoje, para a maioria dos usuários o computador é um dispositivos simples de utilizar e isso é possível devido aos avanços que o tornaram cada vez mais intuitivo e de fácil interação. A facilidade de interação entre o homem e a máquina se dá através de softwares que são desenvolvidos com esse objetivo. Um software ou programa de computador, é o resultado da compilação ou interpretação de um conjunto de instruções (roteiros) elaboradas para realização de uma tarefa ou resolução de um problema. Estas instruções devem seguir padrões

de sintaxe e regras para que possam ser compreendidas e executadas pelo computador. O desenvolvimento da solução e implementação por meio da codificação engloba criar ou deduzir o raciocínio necessário para resolução do problema pelo computador, ou seja, o programador deve pensar igual a um computador no momento em que vai escrever o seu programa.

O processo de desenvolvimento de um programa de computador possui inúmeras etapas, que vão desde o estudo do problema, estrutura e arquitetura, possíveis erros e soluções até a implementação do software em um linguagem de programação

> Há diversas linguagens de programação com características específicas. A escolha da linguagem a ser utilizada está associada a diversos fatores, dentre os mais importantes pode-se citar o paradigma de desenvolvimento do software. Um paradigma pode ser definido como um modelo de programação, ou seja, é um padrão utilizado pelo programador para orientar a maneira como o software será desenvolvido. (BERSSANETTE, 2016, p.17)

Assim, programar nos diferentes paradigmas significa representar segundo padrões diferentes a solução do problema a ser resolvido. Pode-se dizer que um mesmo problema pode ser abordado empregando qualquer um dos paradigmas existentes, resultando em maior ou menor esforço do programador.

> ...programar computadores é uma atividade que requer o conhecimento do conteúdo que está sendo tratado, o domínio de uma linguagem de programação onde a solução do problema deve ser representada, e criatividade, uma vez que há sempre inúmeras maneiras de se chegar a uma solução por meio da programação. A programação de computadores é uma atividade exigente, que requer do programador certas habilidades a fim de que possa implementar soluções para um determinado problema e representa-las no ambiente computacional. A programação está relacionada com outras atividades como especificação, projeto e modelagem. (BERSSANETTE, 2016, p.17)

Para a execução de todas estas atividades, a literatura menciona diversas características, visando sintetizar as principais, apresentamos o Quadro 1.

Principais fatores envolvidos na programação mencionadas na literatura		
Habilidade/Competência	**Definição/relação com a programação**	**Autores**
Raciocínio lógico / Raciocínio crítico /Pensamento Sistêmico	Processo de estruturação do pensamento de acordo com as normas da lógica que permite chegar a uma determinada conclusão ou resolver um problema.	(DEREMER, 1993); (WILSON; SHROCK, 2001); (RAABE; SILVA, 2005); (PEREIRA JÚNIOR et al.,2005); (SANTOS; COSTA, 2006); (GOMES; MENDES, 2007)

Resolução de problemas	Habilidade que envolve processos cognitivos como criatividade e racionalidade, a partir de um conjunto de meta-habilidades mentais tais como a abstração, inferência, dedução entre outros, além disso, esta habilidade está diretamente ligada a leitura e compreensão.	(FALKEMBACH; AMORETTI; et al., 2003); (GOMES; MENDES, 2007); (MARTINS; MENDES; FIGUEIREDO, 2010)
Abstração	Processo de generalização ao reduzir o conteúdo de informação de um conceito, a fim de reter apenas uma informação que seja relevante para um propósito particular.	(PEREIRA JÚNIOR; RAPKIEWICZ, 2004); (RAABE; SILVA, 2005); (HABERMAN; MULLER, 2008); (PIVA JR; FREITAS, 2010), (MARTINS; MENDES; FIGUEIREDO, 2010)
Habilidade matemática	Existe a crença de que os conceitos a fim de dominar problemas matemáticos são semelhantes aos de programação.	(HENDERSON, 1987); (DEREMER, 1993); (WILSON; SHROCK, 2001); (KOLIVER; DORNELES; CASA, 2004); (RAABE; SILVA, 2005); (GOMES, A.; MENDES, 2007); (MOTA et al., 2009)

Quadro 1 - Síntese dos principais fatores envolvidos na programação mencionadas na Literatura (BERSSANETTE, 2016)

Observando o Quadro 1 é possível constatar que programadores experientes possuem um conjunto de recursos (habilidades, competências, experiências) que podem utilizar em qualquer parte do processo da solução do problema para computador. Portanto, o aprendizado de programação está associado ao desenvolvimento deste conjunto de recursos. Buscar metodologias que desenvolvam de forma conjunta essas habilidades e competência podem trazer resultados significativos para o processo de ensino e aprendizagem.

2.1.2 O ensino de programação no Brasil

No Brasil, o ensino de programação ocorre principalmente nas Universidades, em cursos superiores nas áreas de Computação e Informática. Esse conteúdo também é trabalhado em alguns cursos técnicos profissionalizantes subsequentes ou na modalidade integrada ao ensino médio. Também é possível observar algumas iniciativas relacionadas ao ensino de programação na educação básica com o uso de ferramentas mais lúdicas.

Segundo as diretrizes curriculares para os cursos de informática e computação do MEC, a disciplina de Programação faz parte da área de formação básica em Computação e

Informática, juntamente com as matérias de computação, algoritmos e arquitetura de computadores (BRASIL, 2001).

> Geralmente o processo inicial de ensino/aprendizagem de programação ocorre por meio de um conjunto de disciplinas introdutórias que são identificadas por nomes como: algoritmos, lógica de programação, linguagem de programação, técnicas de programação entre outras. Estas disciplinas têm como objetivo fornecer aos alunos os conceitos básicos de programação e isto representa um pequeno conjunto de comandos e conceitos, dos quais os alunos devem utilizar para implementar soluções para um determinado problema e representa-las num ambiente computacional. (BERSSANETTE, 2016, p.20)

A programação é possivelmente uma das únicas matérias que tenta ensinar métodos de resolução de problemas gerais, visto que em cursos como química, matemática e física, o estudante lê e aprende conceitos para se resolver problemas específicos não sendo capaz de generalizá-los (GRIES, 1974).

Alguns aspectos fundamentais em um curso introdutório seriam: 1º - Identificar um problema; 2º - Como resolver problemas; 3º - Como descrever uma solução algorítmica; 4º - Como verificar se um algoritmo está correto (GRIES, 1974; GOMES; HENRIQUES; MENDES, 2008).

De modo geral, nestas disciplinas introdutórias, o conteúdo tratado inclui principalmente a descrição dos passos necessários para se solucionar um problema, entradas e saídas, constantes e variáveis, tipos primitivos de dados, instrução de atribuição, operadores aritméticos, relacionais e lógicos, estruturas simples de controle de fluxo, decisão e repetição. Esses conceitos normalmente são tratados dentro de algoritmos e evita-se num primeiro momento o contato com a linguagem de programação.

Ao analisarmos o Quadro 1 que destaca os principais fatores envolvidos na programação mencionadas na literatura vemos que alguns conteúdos poderiam ser mais explorados, o que seria uma via possível para se conter problemas no aprendizado, a evasão e a desistência. A exploração de aspectos como raciocínio lógico, raciocínio crítico, conhecimento prévio, pensamento sistêmico e abstração deve andar aliados a fatores práticos relacionados aos passos necessários para se solucionar um problema mencionadas no parágrafo anterior.

Desta forma, a realização de atividades práticas podem ter uma grande importância, pois conforme apontam (HABERMAN; MULLER, 2008; NEDSSEN; CASPERSEN, 2008),

a forte carga de conceitos abstratos presentes no processo da elaboração e implementação da solução do problema no ambiente computacional pode não ser motivador para continuar os estudos

Sendo assim, é importante dar oportunidade ao estudante de observar a execução de um programa e seus resultados, o que lhe oferece mais um caminho para tratar dificuldades encontradas em aspectos teóricos, uma vez que outra limitação encontrada em alguns cursos é justamente, deixar de enfatizar a solução de problemas para se concentrar em conceitos teóricos (NOBRE; MENEZES, 2002; KOLIVER; DORNELES; CASA, 2004; GOMES; MENDES, 2007).

Ambrósio (2001) destaca uma característica interessante relacionada a estudantes de programação, é o ganho significativo de aprendizagem em determinados momentos por alguns alunos e com isso avançam sem problemas durante os conteúdos. No entanto outros alunos mesmo se esforçando não conseguem atingir os objetivos propostos pela disciplina.

Dunican (2002), identifica três categorias de estudantes iniciantes em programação:

I. Alunos que não têm a aptidão para compreender os conceitos básicos, muitas vezes, resultado de uma escolha equivocada do curso;

II. Alunos que podem captar os conceitos essenciais se expostos a abordagem de ensino eficaz; e

III. Alunos que são totalmente confortáveis com a natureza abstrata de conceitos de programação.

Seguindo essa classificação e assumindo que esteja correta, o professor deve ficar atento e se possível identificar os alunos pertencentes à segunda categoria, assegurar condições adequadas de ensino/aprendizagem para que a maioria destes possam progredir. Entretanto, não devemos considerar que o processo de ensino/aprendizagem de programação possa ser considerado apenas uma receita didática.

Conforme apresentado durante esta seção, é possível observar que o aprendizado de programação de computadores é uma das bases na formação de estudantes dos cursos de informática e computação. Entretanto aprender a programar computadores não é uma tarefa simples, tampouco trivial (JENKINS, 2002; ROBINS; ROUNTREE; ROUNTREE, 2003).

Ao longo dos anos, o processo de ensino/aprendizagem dos fundamentos de programação de computadores, tem se mostrado difícil para estudantes e professores, estas dificuldades levaram muitos professores e pesquisadores a estudar suas causas e propor soluções variadas que visam de alguma maneira atenuar estes problemas. Grande parte destas pesquisas são voltadas especialmente para estudantes novatos em programação, como citado em Delgado et al. (2004), Pereira Júnior e Rapkiewicz (2004).

Conforme Sheard (2009), o grande volume de pesquisas referente à programação introdutória é reflexo das dificuldades relacionadas ao tema que faz com que o ensino de programação de computadores seja considerado um dos sete grandes desafios na educação em informática.

Desta forma, essas pesquisas buscam apresentar iniciativas que podem de alguma maneira contribuir para atenuar dificuldades apresentadas no processo de ensino/aprendizagem de programação. A subseção a seguir busca apresentar parte destas dificuldades destacadas na literatura.

2.1.3 O ensino de programação e suas dificuldades

Nesta seção serão tratadas algumas das dificuldade identificadas no processo de ensino/aprendizagem e quais as ferramentas professores utilizam como apoio a esse processo.

Podemos observar na literatura relacionada as dificuldades apresentadas por alunos e professores durante o processo de ensino/aprendizagem de programação originam-se em dois fatores, sendo:

I. Os elevados níveis de insucesso; e

II. As elevadas taxas de desistência e até mesmo abandono do curso;

Berssanete (2016) observa que as disciplinas de introdução a programação podem ser consideradas um dos gargalos existentes nos cursos, dificultando ou até mesmo impedindo a continuidade dos estudantes. As dificuldades abrangem os quatro integrantes envolvidos no processo que é a instituição, o professor, o aluno, e os conteúdos.

Ao observarmos as instituições estamos falando sobre a falta de infraestrutura, como laboratórios, equipamentos e em alguns casos até mesmo a falta de internet. Somados esses

problemas com uma metodologia que não é atraente para os alunos, temos os ingredientes para os altos índices de insucesso.

Tradicionalmente, os conteúdos das disciplinas introdutórias a programação são apresentados da forma mais comum, como em todas as áreas de conhecimento, ou seja, apresentação da teoria, construção de exemplos e proposição de exercícios práticos inicialmente simples e paulatinamente complexos (RODRIGUES JÚNIOR, 2004).

O modelo tradicional de ensino de programação não consegue facilmente motivar os alunos a se interessar pela disciplina. Pois não é clara para os mesmos a importância destes conteúdos para sua formação. Essa falta de motivação dos alunos possivelmente pode ser agravada especialmente quando os conteúdos são apresentados sem o auxílio de uma linguagem de programação, podendo dificultar o entendimento e utilidade dos conteúdos apresentados (AUSUBEL; NOVAK; HANESIAN, 1980; HABERMAN; MULLER, 2008).

Desta forma BERSSANETE 2016 apresenta outro quadro que mostra algumas dificuldades enfrentadas por professores no que diz respeito às metodologias implementadas.

Dificuldades inerentes aos professores mencionadas na literatura	
Dificuldades	**Autores**
Prática docente	(BIGGS, 2003); (JENKINS, 2001); (GIRAFFA; MARCZAK; ALMEIDA, 2002); (PIMENTEL; FRANÇA; OMAR, 2003); (CASPERSEN; BENNEDSEN, 2007); (RODRIGUES JÚNIOR, 2004)
Ausência de metodologias e/ou práticas de ensino adequadas.	(JENKINS, 2001); (GIRAFFA; MARCZAK; ALMEIDA, 2002); (PIMENTEL; FRANÇA; OMAR, 2003); (RODRIGUES JÚNIOR, 2004); (SANTOS; COSTA, 2006); (CASPERSEN; BENNEDSEN, 2007); (GOMES; HENRIQUES; MENDES, 2008)
Restrição de tempo	(KOLIVER; DORNELES; CASA, 2004); (ROCHA et al., 2010)
Impossibilidade de acompanhamento individual da	(JENKINS, 2001); (TOBAR et al., 2001); (BIGGS, 2003);

aprendizagem dos alunos	(RAABE; SILVA, 2005); (ROCHA et al., 2010)
Ensino sem enfoque na resolução de problemas	(NOBRE; MENEZES, 2002); (KOLIVER; DORNELES; CASA, 2004); (GOMES; MENDES, 2007)
Abordagem pouco motivadora	(BORGES, 2000); (KOLIVER; DORNELES; CASA, 2004); (RODRIGUES JÚNIOR, 2004)
Heterogeneidade dos alunos	(ROBERTS, 1999); (ROBERTS, 2001); (KOLIVER; DORNELES; CASA, 2004)

Quadro 2 - Síntese das principais dificuldades inerentes aos professores mencionadas na literatura (BERSSANETE, 2016)

O uso de metodologias e/ou práticas de ensino adequadas pode contribuir para atenuar algumas destas dificuldades. Para isto é importante que o docente vise um ensino contextualizado com enfoque na resolução de problemas, e sempre que possível utilizando-se de abordagens motivadoras levando em consideração os conhecimentos prévios do aluno, com exemplos do dia a dia.

Nobre e Menezes (2002) apontam as seguintes dificuldades vivenciadas pelos professores:

I. Reconhecer as habilidades inatas de seus alunos;

II. Apresentar técnicas para soluções de problemas;

III. Trabalhar a capacidade de abstração do aluno, tanto na busca das possíveis soluções como na escolha da estrutura de dados a ser utilizada; e

IV. Promover a cooperação e a colaboração entre os alunos.

A literatura apresenta desta forma algumas soluções para minimizar estes problemas encontrados durante o processo de ensino. Na subseção seguinte será apresentado uma das soluções presentes na literatura que visam atenuar estas dificuldades, que é o utilização de ferramentas que auxiliam nesse processo.

2.1.4 Ferramentas de apoio ao ensino de programação

O ensino de programação envolve a escrita de código, que pode acontecer dentro de ferramentas desenvolvidas para esse fim. Infelizmente grande parte destes ambientes são

elaborados mais para fins profissionais do que pedagógicos. Assim, um dos caminhos adotados por professores é buscar ferramentas desenvolvidas com o intuito de ajudar estudantes. Nesta subseção será apresentado o resultado de um mapeamento sistemático realizado com objetivo de identificar e mapear as ferramentas utilizadas, bem como o objetivo de utiliza-las.

Para esse mapeamento foram elaboradas cinco questões de pesquisa que levam em consideração a evolução histórica de trabalhos na literatura, as principais ferramentas utilizadas no apoio a metodologia, a acessibilidade destas ferramentas (softwares livres ou pagos), qual o tipo de metodologia utilizada (ex: games, interdisciplinar), e se as mesmas são utilizados na educação básica ou superior.

Num olhar histórico na literatura sobre novas experiências no uso de aplicações encontramos 34 trabalhos de pesquisa no recorte temporal de três anos.

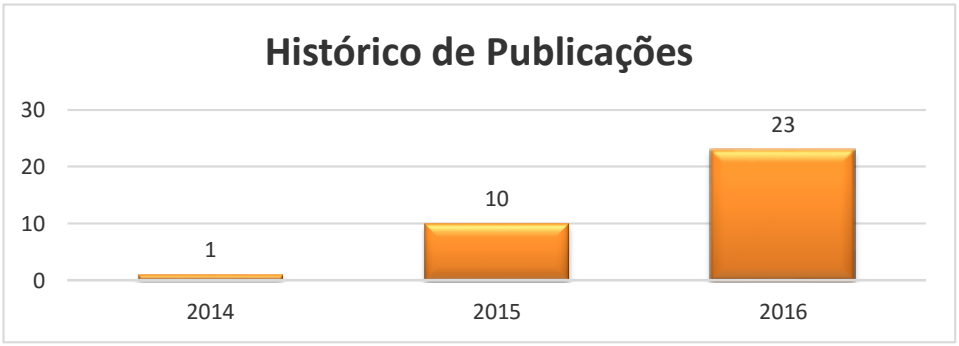


Gráfico 1 -Evolução histórica de publicações

Observando a figura acima, podemos perceber que e evolução histórica de publicações sobre metodologias e ferramentas sobre o ensino de programação nos principais eventos e periódicos brasileiros tem apresentado um grande salto. Ao olharmos um pouco para o ensino básico, podemos observar que o ensino normalmente é focando na utilização das tecnologias com o intuito de alfabetização digital, ou seja, fornecer o acesso as tecnologias. Para mudar esse cenário, precisamos pensar em um ensino básico com proficiência digital, incluindo o pensamento computacional e a programação. Segundo Wangenheim (2014), isso atualmente é uma tendência mundial e existem diversos ambientes para ensinar computação para cada faixa etária.

O segundo ponto levantado é a relação de quais ferramentas são utilizadas no processo metodológico de ensino. Pensar numa metodologia que atraia a atenção do estudante nos momentos iniciais do ensino é de fundamental importância no processo de ensino aprendizagem e em se tratando de ensino de programação o uso de ferramentas ou aplicativos para a execução de tarefas é indispensável.

Porém, o fato de utilizar alguma ferramenta não garante que os objetivos do processo de ensino aprendizagem sejam alcançados. Diante disso, é fato de que o conhecimento e o domínio do saber são de responsabilidade do professor, entretanto, o uso de aplicações de forma lúdica e interdisciplinar, poderá ser uma ferramenta didática quando na transposição didática desse saber. Através da técnica, o conhecimento poderá melhor se adequar ao perfil dos nossos alunos na contemporaneidade.

Todos os artigos abordam metodologias de ensino de programação, mas alguns deles não citam a utilização de ferramentas ou aplicativos. O quadro abaixo mostra as principais ferramentas utilizadas no apoio as metodologias de ensino de programação.

Ferramentas utilizadas como apoio a metodologia de ensino.		
Ferramentas	**Autores**	**Descrição**
SCRATCH	(BATISTA; CASTRO JR.; BOGARIM; LARREA, 2015); (BELCHIOR; BONIFÁCIO; FERREIRA, 2015); (CARDOSO, ANTONELLO, 2015); (GALDINO; NETO; COSTA, 2015); (GOMES; MELO; TEDESCO, 2016); (LOPES; DUARTE; NOGUEIRA; LOPES; FERREIRA, 2016); (MATTOS; FERREIRA; ANACLETO, 2016); (OLIVEIRA; RODRIGUES; QUEIROGA, 2016); (RAFALSKI; SANTOS, 2016); (SALAZAR, ODAKURA, BARVINSKI, 2015); (SANTOS; MONTEIRO; MACHADO; LINS; RAMOS; BATISTA, 2015); (SILVA; GARCIA; OLIVEIRA; TRINDADE; SGARBI; NASCIMENTO, 2016); (SILVA; ARANHA; SANTOS, 2016); SOUZA; MOMBACH, 2016); (TEIXEIRA; ORO; BATISTELA; MARTINS; PAZINATO; 2015); (VERA; MIRANDA; COSTA; MATOS, 2016);	Ferramenta desenvolvida dentro do mesmo contexto lúdico proposto pelo LOGO, nesta utiliza-se a metáfora na qual as instruções de um programa são elaboradas a partir de blocos de comandos que se encaixam uns nos outros. Pode ser usada para elaboração de jogos de computador, histórias interativas, obras de arte gráfica e animação por computador, e outros projetos de multimídia.

	(WANGENHEIM; NUNES; SANTOS, 2014)	
APP INVENTOR	(BATISTA; CASTRO JR.; CANTERO; BOGARIM; LARREA, 2016); (RIBEIRO; MANSO; BORGES, 2016)	A plataforma fornece a possibilidade dos estudantes praticarem conceitos de algoritmos para elaborarem aplicativos que serão utilizados em seus dispositivos móveis.
ARDUINO	(ALBUQUERQUE, BREMGARTNER, LIMA, SALGADO, 2016); (MARINHO; MOREIRA; COUTINHO; PAILLARD; NETO, 2016)	Plataforma de prototipagem eletrônica de hardware livre e de placa única, projetada com um microcontrolador e suporte de entrada/saída embutido e possui uma linguagem de programação padrão baseada em C/C++.
SOCRATIVE	(IZEKI; NAGAI; DIAS, 2016)	Ferramenta que permite ao professor engajar e avaliar os alunos enquanto a aprendizagem ocorre. O professor pode criar um questionário e aplicá-lo de diferentes formas, de acordo com a metodologia de ensino empregada. O aluno deverá responder o questionário em um tempo total determinado pelo professor, que pode visualizar as respostas de cada aluno ao mesmo tempo em que eles respondem as perguntas.
CODE BAYMAX	(MARTINS; BRELAZ; NASCIMENTO; ALFAIA; MARTINS, 2016)	O Code Baymax incorpora conceitos básicos de pensamento lógico como algoritmos (sequência de instruções para resolver um problema), funções, ciclos, sub-rotinas e loops. Isso oferece às crianças a satisfação de solucionar desafios cada vez mais complexos por meio de novas formas de jogar e pensar.
CODE STUDIO	(JESUS; SILVA, 2016); (LOPES; DUARTE; SOUSA; SOUZA; PEREIRA, 2016)	O Code Studio é um editor de código-fonte e suporta um número de linguagens de programação e um conjunto de recursos que podem ou não estarem disponíveis para uma dada linguagem.
LEGO	(CAMBRUZZI; SOUZA, 2015)	Lego Mindstorms é uma linha do brinquedo LEGO voltada para a educação tecnológica e

MINDSTORMS		que possui uma plataforma própria para programação.
LORD OF CODE	(NAGANO; DIRENE, 2016); (SOUZA; COSTA; SILVA; TERRA, 2016)	Inserindo o aspecto lúdico dos jogos a desafios envolvendo lógica de programação, a ferramenta Lord of Code (LoC) visa auxiliar na aprendizagem de programação por meio de um jogo que estimula a interpretação e resolução de problemas em códigos escritos em Java.
PYSCHOOL.NET	(OLIVEIRA; NOGUEIRA; MOTTA; MEIRELES, 2015)	Uma plataforma de ensino que disponibiliza um editor de programas e um campo de provas onde se pode testar o programa e construir jogos.
ROBOCODE	(AMARAL; SILVA, PANTALEÃO, 2015); (RODRIGUES; QUEIROGA; OLIVEIRA; MORE, 2016); SANTIAGO; KRONBAUER, 2016	É um jogo de programação, onde o objetivo é desenvolver um robô para lutar contra outros robôs. Utilizando as linguagens de programação .NET (C#) ou JAVA, podem ser implementadas as classes e objetos que definirão a inteligência artificial por trás da estratégia de batalha do seu robô.
SPYDER	(CARVALHO; OLIVEIRA; GADELHA, 2016); (MARTINS, REIS, MARQUES, 2016)	É uma IDE[1] para a linguagem Python e possui recursos que auxiliam o aprendizado do programador iniciante, tais como destacador de sintaxe, enumerador de linhas, visualizador do estado de variáveis, bibliotecas Numpy e MathPlotLib

Quadro 3 - Ferramentas de apoio ao ensino de programação encontradas no mapeamento

O SCRATCH possibilita que as crianças elaborem animações, histórias interativas ou jogos, tornando fácil a combinação de gráficos, imagens, fotos, música e som. Com o SCRATCH é possível criar personagens que dancem, cantem e interagem uns com os outros.

[1] IDE - *Integrated Development Environment* ou Ambiente de Desenvolvimento Integrado, é um programa de computador que reúne características e ferramentas de apoio ao desenvolvimento de software.

Permite também integrar imagens com efeitos de som e clipes musicais para criar um cartão interativo de aniversário para um amigo ou para criar um mapa interativo. Segundo Resnick (2009), diversas pesquisas mostram que o uso do SCRATCH contribui positivamente no ensino de computação em escolas. Criando programas de software com o SCRATCH, crianças aprendem a pensar criativamente, a trabalhar de forma colaborativa e a pensar de forma sistemática na solução de problemas.

Quanto a acessibilidade as ferramentas livres no âmbito educacional tornaram-se importantes no combate à exclusão social, uma vez que propiciam a redução de custos na elaboração de laboratórios e de projetos voltados para educação e inclusão digital. Em geral, eles apresentam uma boa qualidade e funcionam em máquinas com configurações de hardware mais antigas, o que ajuda a reduzir os custos.

As publicações encontradas no mapeamento sistemático trazem 11 ferramentas que são utilizadas para o ensino de programação, todas são livre ou possuem versões livres. Algumas aplicações têm versões básicas livre e versões completas que necessitam de aquisição para que funcionem. Como a todos os trabalhos tratam de experimentos com as ferramentas, foram utilizadas as versões básicas.

Quanto ao emprego do software dentro da metodologia de ensino temos o gráfico abaixo.

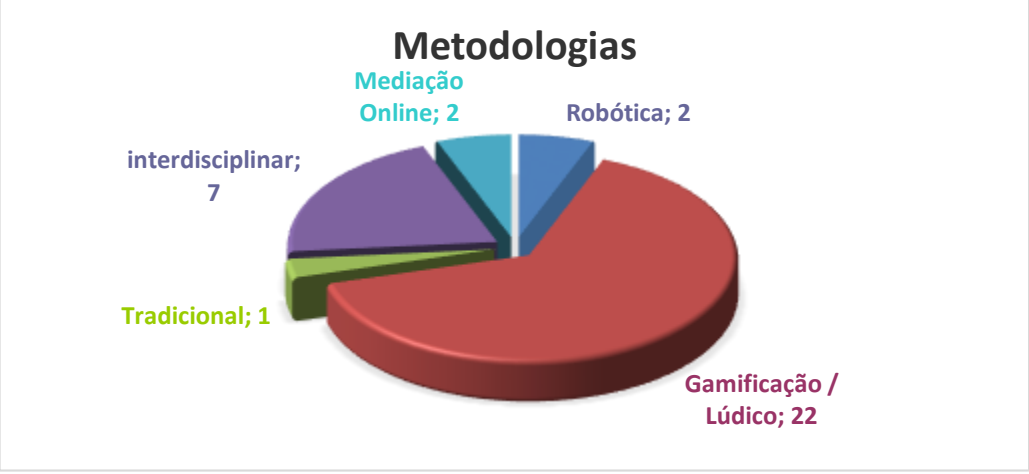


Gráfico 2 - Metodologias utilizadas para o ensino de programação

Como demonstra o gráfico acima, na grande maioria dos estudos, a Gamificação/Lúdico é a metodologia mais utilizada no processo de ensino aprendizagem de programação. O termo gamificação, segundo Kapp (2012), significa utilizar os mecanismos, a estética e o pensamento dos jogos com o objetivo de encorajar as pessoas, motivar as ações, promover aprendizado e resolver problemas. O termo teve origem na indústria de mídias digitais em 2008, mas foi popularizado somente na segunda metade de 2010 (DETERDING et al, 2011).

De acordo com sua definição, a gamificação vem sendo utilizada em sites e aplicativos móveis voltados ao consumidor com a finalidade de convencer as pessoas a utilizarem determinada plataforma. Muntean (2011) relata que os princípios da gamificação podem ser aplicados em processos de jogos educacionais associando a motivação intrínseca à extrínseca objetivando melhorar o engajamento e interesse do aprendiz. A gamificação é sustentada a partir dos princípios do design de jogos. Esses princípios definem o que são jogos e como eles funcionam, bem como os elementos do design que o compõem. Os jogos educacionais destacam-se como ferramentas que facilitam no ensino e aprendizagem dos usuários, no entanto esta deve estar interligada à outros recursos de aprendizagem (MUNTEAN, 2011). Depois aparecem trabalhos que utilizam uma abordagem interdisciplinar, tendo como princípio a ligação com outras áreas do conhecimento, como matemática, física e música.

Outro ponto observado no mapeamento foi quanto aos espaços ou instituições onde as ferramentas são utilizadas. A introdução de conceitos de Computação, enquanto ciência, na educação básica é de fundamental importância e relevância (CSTA, 2011). No entanto, seu ensino, no Brasil, por muitos anos tem tido enfoque nos âmbitos da graduação e pós-graduação. Atualmente vemos uma grande mudança nesse sentido, pois com o advento da tecnologia, a facilidade de acesso e o incentivo do poder público tem mudado esse paradigma. O ACM Model Curriculum for K-12 Computer Science – (CSTA, 2011) defende que é necessário o desenvolvimento de habilidades computacionais na Educação Básica, no sentido da Ciência da Computação ser importante intelectualmente, promover múltiplos caminhos profissionais futuros, desenvolver a capacidade de resolver problemas, apoiar e relacionar-se com outras ciências e motivar os estudantes.

Neste mapeamento constatamos que grande parte dos trabalhos de pesquisa se desenvolvem na educação básica, visando a introdução dos elementos computacionais de forma lúdica, com os mais diversos objetivos, como ajudar na aprendizagem de conteúdos de matemática.

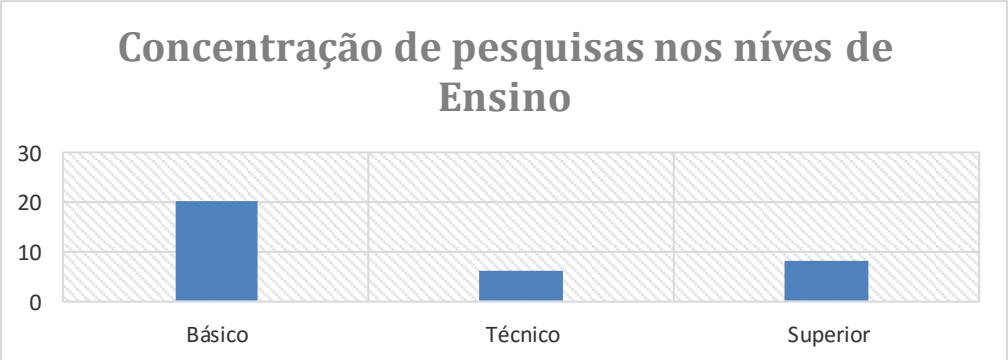


Gráfico 3 - Campo de desenvolvimento das pesquisas sobre o ensino de programação.

Desta forma, observa-se que as pesquisas brasileiras com relação as metodologias de ensino utilizadas no ensino da programação e quais ferramentas dão apoio a essa metodologia, constatou-se o aumento no número de publicações relacionadas a ambientes de ensino de programação com o passar dos anos e a preferência dos professores em utilizar o SCRATCH, com ferramenta de apoio, pois tem um caráter mais lúdico para o ensino de programação. Tal fato está diretamente ligado a tendência de Gamificação das ferramentas, situação que prende mais a atenção do estudando, imergindo o mesmo, dentro de um jogo onde ele acaba aprendendo a programação.

Os espaços onde os estudos são desenvolvidos também chamam a atenção, pois durante muitos anos o ensino de programação estava restrito as universidades e com o advento das tecnologias, o barateamento da mesma e os incentivos governamentais é possível perceber que atualmente o ensino de programação está presente também na educação básica.

Portanto, nesta subseção apresentamos um breve panorama sobre as principais dificuldades encontradas no ensino de programação bem como uma das propostas da literatura para minimizar estas dificuldades. Neste caso, temos as ferramentas de apoio pedagógico, que tendem a ser mais lúdicas do que os softwares tradicionais.

Na próxima seção será apresentado um panorama sobre a relação da tecnologia e computação com a música que historicamente estão interligadas, seja na produção musical ou no processo de ensino/aprendizagem.

2.2 Histórico da tecnologia musical

A música como expressão artística, tem o objetivos de expressar o sentimento da forma mais pura. Como o pintor usa arte para demonstrar seus sentimentos, o escultor usa sua obra para transparecer suas impressões interiores, o músico irá utilizar suas ferramentas para mostrar seus sentimentos e devaneios, suas opiniões ou mesmo a ausência delas sobre determinado assunto. Desta maneira, a música necessariamente não necessita um computador para existir, porém, com a advento da computação aliada a ferramentas e meios eletrônicos tem produzido ao longo dos anos, principalmente a partir da década de 70, alterações na maneira de fazer, interagir e divulgar música. O avanço da computação na área da música é marcante a tal ponto de utilizar inteligência artificial, redes neurais e realidade virtual na construção de sistemas inovadores e que permitem uma grande flexibilidade ao usuário.

A invenção do telefone por Alexander Graham Bell (1876), estabeleceu que o som podia ser convertido em sinal elétrico e vice-versa. Por sua vez, a invenção do gramofone, que rapidamente se seguiu, estabeleceu as possibilidades de armazenamento e alteração do som. Por volta de 1906, Thaddeus Cahill criou o Dinamofone, primeiro instrumento que produzia som por meios elétricos. A geração do som era feita a partir de dínamos e a transmissão por meio de cabos telefônicos.

Segundo Gohn (2009), a invenção do oscilador a válvula por Lee De Forest, representa outro momento histórico de grande importância. O oscilador, que representa a base para a geração do som eletrônico, tornava possível a geração de frequência a partir de sinais elétricos e, consequentemente, a construção de instrumentos eletrônicos mais fáceis de manejar. O primeiro desses foi desenvolvido pelo russo Lev Termen (Leon Theremin), em 1919/1920 e foi posteriormente melhorado por volta da década de trinta. Este instrumento, o “Theremin”, usava dois osciladores controlados pelo movimento das mãos do executante em torno de duas antenas verticais, sem nunca tocá-las. Outros instrumentos eletrônicos rapidamente o seguiram (GOHN, 2009). O inventor alemão Jörg Mager introduziu alguns deles na década de trinta. O “Ondas Martenot” foi criado pelo francês Maurice Martenot e o

"Trautonium" pelo alemão Friedrich Trautwein, ambos em 1928. Neste mesmo ano o americano Lores Hammond produziu o primeiro órgão elétrico.

Os compositores de música eletroacústica geram suas composições em estúdios e laboratórios. As audições são realizadas através da reprodução das composições em fita. Este tipo de técnica foi chamada de "tape-music", ou música para fita magnética. Tal música, independente do processo de composição (acústico ou eletrônico), tem como particularidade o fato de que a sua execução em público só poderia ser feita por meio de reprodutores de fita e amplificação eletrônica. A sua execução era exatamente a criação do compositor, eliminando com isso a ideia do instrumento/intérprete. O compositor fazia os dois papéis utilizando-se de audições para mostrar suas obras, já que essas eram concebidas em laboratórios musicais.

Ainda na década de 50, a partir da antiga idéia de criar sons usando a eletricidade, Herbert Belar e Harry Olsen inventaram o Mark II RCA Music Synthesizer, o primeiro sintetizador controlado por voltagem. Deste instrumento somente um modelo foi fabricado.

Max Mathews, desenvolveu no Bell Laboratories, em Nova Jersey, o primeiro programa de computador para a música em 1957, num computador de grande porte. O programa chamado Music I tinha uma única voz, uma forma de onda triangular, não possuía ADS[2]R e só controlava a afinação, intensidade e duração dos sons. O Music I deu origem a uma série de programas musicais como Music II, Music III e Music IV. Estes programas abriram espaço para uma "avalanche" de novos programas musicais de todas as categorias que foram criados a partir de 1976 com a difusão dos microcomputadores e a utilização de linguagens de programação de alto nível e grande portabilidade.

Na década de 50, Robert Moog era um engenheiro que montava e vendia Theremins. Na década de 60 ele iniciou a construção de novos equipamentos que resolveu chamar de sintetizadores. Ele foi o responsável pela invenção dos primeiros sintetizadores modulares Moog, que foram mundialmente difundidos através dos primeiros tecladistas que utilizaram

[2] ADSR – (Attack – Decay – Sustain – Release)- Sigla referente aos termos Ataque, Decaimento, Sustentação e Repouso e é uma das formas mais comumente utilizadas para aplicar um envelope de amplitude a um som para produzir um timbre característico de um instrumento musical.

sintetizadores em suas gravações e performances, como Keith Emerson e Wendy Carlos. A chegada do sintetizador Moog e de outros instrumentos eletrônicos semelhantes ao mercado, em 1964, representou uma revolução nas técnicas da música eletrônica.

Os sintetizadores possibilitaram o surgimento de novos gêneros musicais baseados em suas possibilidades tímbricas e de controle eletrônico dos parâmetros musicais. Os músicos, então, utilizavam a síntese para criar seus próprios sons exigidos em suas composições e que até então não eram possíveis de serem produzidos pelos instrumentos convencionais.

A música popular começou a assimilar a música eletrônica de vanguarda, e grupos ingleses de rock progressivo da década de 70 como ELP, Yes e Genesis começaram a utilizar sintetizadores em suas performances. A utilização da orquestra sinfônica, coral e sintetizadores na música popular foi difundida principalmente por Rick Wakeman com a obra Viagem ao Centro da Terra obtendo uma sonoridade ímpar até então na música popular. A utilização só de equipamentos eletrônicos originou também a música eletrônica popular, tendo como expoentes Kraftwerk, JeanMichel Jarre e Vangelis. Este último recebendo um Oscar nos EUA pela trilha sonora do filme Chariots of Fire, realizada apenas com piano e sintetizadores, abrindo ainda mais o caminho para a música eletrônica. Até então, apenas compositores eruditos com trilhas para orquestra haviam alcançado esse prêmio.

Os sintetizadores evoluíram e deixaram de ser teclados analógicos com osciladores e potenciômetros para alteração do som para se tornarem uma incrível linha de produtos que vão desde módulos de som até computadores com software que os tornam sintetizadores. Grandes empresas fabricantes de sintetizadores ganharam o mercado como a Roland, Yamaha e a Korg. Com base nos conceitos herdados dos sintetizadores analógicos modulares, vários tipos de síntese como FM, aditiva, subtrativa, linear e modelagem física foram desenvolvidas para os sintetizadores digitais modernos.

Na década de 70 já existiam ligações entre os instrumentos musicais, mas foi só em 1983 que a MIDI - Musical Instrument Digital Interface - surgiu para padronizar as comunicações entre instrumentos eletrônicos. Como o computador já havia sido usado para a música, mas sem uma ligação efetiva com instrumentos do meio externo, ele foi rapidamente empregado na ligação com sintetizadores através do interfaceamento MIDI. Essa utilização da MIDI originou uma revolução no campo da música.

A criação dos microcomputadores pessoais e sua difusão nos anos 80 facilitou a configuração de um sistema MIDI completo pois barateou e simplificou a aquisição do hardware básico para Introdução à Computação Musical aplicações em sistemas musicais. Logo após o padrão MIDI ser difundido, fabricantes de software iniciaram uma produção em grande escala dos mais diversos aplicativos musicais que utilizavam a comunicação entre instrumentos e o computador. No final de 1986, uma grande linha de produtos já existia. Os computadores PC multimídia tornaram o padrão MIDI e a gravação digital em computador ainda mais difundida por já apresentarem esses recursos inclusos no preço final do computador.

Além das incontáveis contribuições da computação e tecnologia à música no que diz respeito a produção sonora, também vemos inúmeras iniciativas no tocante a educação musical, onde a tecnologia tem desempenhado um papel importante. A criação de aplicativos e ferramentas que auxiliam os professores no processo de ensino aprendizagem vem crescendo com o passar dos anos.

Modelos tradicionais de educação musical tem sido aperfeiçoados com o auxílio da tecnologia por professores e pesquisadores que buscam aproximar a metodologia de ensino ao cotidiano do aluno.

Henderson Filho (2007) aponta que a tecnologia contribui significativamente para o desenvolvimento educacional, visto que além serem empregadas para dinamizar as aulas, também servem de complemento às atividades. As aulas se tornam mais interessantes, participativas, menos cansativas, mais produtivas e condizentes com a realidade atual das tecnologias digitais.

É perceptível o interesse das diversas áreas de conhecimento pelo uso da tecnologia como um recurso inovador e eficiente, no que diz respeito aos aspectos educacionais. Gohn (2009) destaca que na área da música não é diferente, “as tecnologias atuais estão sendo cada vez mais aproveitadas na educação, especialmente na área de música”.

Os caminhos da educação musical estão tomando um rumo diferente, uma vez que os educadores estão sendo cada vez mais estimulados a propor alternativas no âmbito tecnológico para melhorar o ensino de música, a partir de estratégias didáticas com o uso do computador e das multimídias disponíveis (ONOFRIO, 2011).

Com a evolução da informática, consequentemente ocorreu um progresso expressivo dos programas de computadores (softwares). Os softwares trouxeram para o computador funções que antes só eram possíveis fazer de outras maneiras e com outras ferramentas, por exemplo, os programas específicos de gravação musical e criação e edição de partituras substituíram respectivamente os registros de fita magnéticas e a escrita à mão no papel (GOHN, 2009).

> ...na medida em que o acesso às novas tecnologias – especialmente as tecnologias digitais – vai se tornando mais amplo, o uso desses recursos deve passar a integrar o leque de ferramentas de ensino usadas para os mais variados tipos de situações de aprendizagem musical, da musicalização infantil à prática de instrumentos, da composição aos estudos de análise, servindo tanto ao aluno interessado em música como uma atividade lúdica ou de lazer quanto ao estudante comprometido com uma prática mais regular e eventualmente almejando tornar-se um profissional (GOHN, 2003, p. 11).

Sendo assim, proporcionar aos músicos e professores de música, conhecimentos sobre programação e computação permite que esses profissionais busquem implementação de ferramentas que os possam auxiliar na solução de seus problemas. Construir tal ponte pode ser algo difícil de fazer, pois aparentemente, música e computação são áreas do conhecimento bem distintas.

No capítulo 4 será descrita a proposta elaborada para esse trabalho de pesquisa, e nela podemos observar que a música está bem próxima de vários conceitos de programação. Segundo SANTOS 2015, a compreensão de que a música, além de seus aspectos sociais e culturais, também tem seu aspecto matemático, mostra que é possível construir essa relação com a computação que é compreendida também pela matemática. Já na próxima seção será descrita a teoria da aprendizagem significativa.

2.3 Teoria da aprendizagem significativa

Aprendizagem significativa é o conceito central da teoria da aprendizagem de David Ausubel, proposta na década de 60. Nesta teoria o autor destaca que a aprendizagem significativa é o mecanismo humano para adquirir e armazenar a vasta quantidade de ideias e informações representadas em qualquer campo de conhecimento.

Ausubel (2003), destaca que:

> O conhecimento é significativo por definição. É o produto significativo de um processo psicológico cognitivo (“saber”) que envolve a interação entre ideias

> "logicamente" (culturalmente) significativas, ideias anteriores ("ancoradas") relevantes da estrutura cognitiva particular do aprendiz (ou estrutura dos conhecimentos deste) e o "mecanismo" mental do mesmo para aprender de forma significativa ou para adquirir e reter conhecimentos (AUSUBEL, 2003, p. 6).

Berssanette (2016) destaca que essa teoria preconiza que o conhecimento se organiza em estruturas cognitivas, que são conjuntos de conhecimentos que o indivíduo possui sobre um determinado assunto. A aprendizagem torna-se significativa quando os conhecimentos anteriores são inter-relacionados ao novo conteúdo a ser estudado o qual passa a ser incorporado às estruturas de conhecimento, adquirindo significado especial.

Este conceito não consiste numa simples associação e sim uma interação relevante entre os conhecimentos que se possui e os que serão conhecidos (MOREIRA, 1999). Desta forma, a aprendizagem significativa é um processo de mudança do conhecimento alterando a estrutura cognitiva do aprendiz, modificando os conceitos pré-existentes e criando novas conexões.

Nessa perspectiva, novos conhecimentos são construídos à medida que o aprendiz se movimenta no sentido de articular novos saberes aos que já possui, assim, aprendizagem precisa ser ancorada a uma outra já existente na estrutura cognitiva do sujeito para que possa ser então assimilada.

Portanto, um dos pontos fundamentais desta teoria consiste em determinar aquilo que o aprendiz já sabe ou conhece, para que a proposta de ensino seja baseada nestes conhecimentos. A aprendizagem mecânica é o contrário da aprendizagem significativa: a aprendizagem de novas informações ocorre com pouca ou nenhuma associação com conceitos relevantes existentes na estrutura cognitiva (MASINI; MOREIRA, 2001).

Para evitar isso, Ausubel sugere deliberadamente manipular a estrutura cognitiva usando organizadores prévios.

> Organizadores prévios tem a função de ponte entre o que o aprendiz sabe e o que deve saber. Eles visam estabelecer relações entre ideias, proposições e conceitos já existentes na estrutura cognitiva, a fim de que a aprendizagem possa ser significativa. Além disso, também podem ser usados para "reativar" significados obliterados, para "buscar" na estrutura cognitiva do aluno significados que já existiam, mas que não eram usados há algum tempo (BERSSANETTE, 2016, p.40)

Ainda, segundo Ausubel, para que a aprendizagem seja significativa há três condições: predisposição do indivíduo; material potencialmente significativo e estrutura cognitiva capaz

de assimilar a nova informação. A predisposição para aprender está intimamente relacionada com a experiência afetiva que o aprendiz tem no evento educativo. Além disso, a aprendizagem significativa propõe a participação ativa do aluno na aquisição de conhecimento, de maneira a evitar-se uma mera reprodução de conceitos formulados pelo professor ou pelo livro-texto, mas uma reelaboração do aluno (PELIZZARI; KRIEGL, 2002).

Para Ausubel, cada disciplina tem uma estrutura articulada e hierarquicamente organizada de conceitos (MASINI; MOREIRA, 2001). No entanto, a ordem em que os principais conceitos e ideias da matéria de ensino são apresentadas muitas vezes não é a mais adequada para facilitar a interação com o conhecimento prévio do aluno.

> ..., é essencial uma análise crítica da matéria de ensino a ser apresentada ao estudante; o conteúdo precisa ter boa organização lógica, cronológica e epistemológica, mas além disso é indispensável uma análise com foco no estudante e em particular seu conhecimento prévio. Além disso, é importante também não sobrecarregar o aluno de informações desnecessárias, dificultando a organização cognitiva. É preciso buscar a melhor maneira de relacionar, explicitamente, os aspectos mais importantes do conteúdo da matéria de ensino aos aspectos especificamente relevantes de estrutura cognitiva do aprendiz. Este relacionamento é imprescindível para a aprendizagem significativa (BERSSANETTE, 2016, p.41).

O adiamento da experiência de aprendizagem para além da maturidade do estudante desperdiça oportunidades de aprendizagens valiosas e, muitas vezes, reduzindo de forma desnecessária, a quantidade e complexidade dos conteúdos que se pode dominar num determinado período da aprendizagem escolar. Por outro lado, quando um aluno é exposto, prematuramente, a uma tarefa de aprendizagem, antes de estar preparado de forma adequada para a mesma, não só não aprende a tarefa em questão (ou aprende-a com muitas dificuldades), como também aprende com esta experiência a temer, desgostar e evitar a tarefa (AUSUBEL, 2003).

Neste sentido, se faz necessário determinar continuamente o que o aprendiz conhece para ensiná-lo de acordo. Para tal, faz-se necessário mecanismos que visem identificar o estado mental de cada aprendiz, relativo ao domínio de conhecimento em questão, para que possam auxiliar os professores nesta tarefa. Ausubel enfoca a linguagem como um facilitador importante para a ocorrência da aprendizagem significativa (MOREIRA, 1999). Desta forma, é possível observar que os conceitos abordados serão realmente assimilados, se eles forem apresentados numa linguagem coerente e que faça sentido para o aprendiz.

Destaca-se que embora tenha contribuído para o avanço da teoria da construção do conhecimento, Ausubel não proporcionou aos educadores instrumentos simples e funcionais para ajudá-los a averiguar “o que o aluno já sabe”. Tal processo deve ser reinventado por cada educador, que precisa avaliar constantemente os resultados da sua metodologia e realizar as correções.

De maneira resumida, nesta subseção apresentamos os significados atribuídos por Ausubel ao conceito de aprendizagem significativa. Neste Capítulo apresentamos as principais referências que servirão de base para essa pesquisa.

3 METODOLOGIA

No capítulo 3, apresenta-se o processo e fundamentação metodológica do trabalho, iniciando pela definição do campo da pesquisa, a população que será atingida, local onde será desenvolvido o trabalho, etapas da pesquisa, e de que forma a coleta do dados foi realizada.

3.1 Delineamento geral da pesquisa

Traçada a estratégia de se fazer uma pesquisa interdisciplinar que envolva o campo da computação e da arte, este trabalho objetiva introduzir práticas com o software livre Frescobaldi como ferramenta de apoio ao ensino de linguagem de programação para músicos através de uma metodologia baseada na Teoria da Aprendizagem Significativa, além de proporcionar ao músico a possibilidade de interação com uma área distinta além de despertar seu interesse em aprofundar os conhecimentos sobre computação. A hipótese que orienta este trabalho, é a de que o ensino de programação com a valorização do conhecimento prévio sobre música possa utilizado como meio para atrair novos estudantes de programação.

Com relação à natureza, esta pesquisa classifica-se como aplicada, pois objetiva gerar novos conhecimentos relacionados ao ensino e aprendizagem de programação de computadores a partir da implementação da proposta de ensino.

Do ponto de vista dos objetivos, trata-se de uma pesquisa-ação, que é um método ou uma estratégia de pesquisa que agrega vários métodos ou técnicas de pesquisa social, com os quais se estabelece uma estrutura coletiva, participativa e ativa a nível de busca de informação. Já como estratégia, a pesquisa-ação pode ser entendida como um modo de conceber e de organizar uma pesquisa social de ordem prática e que esteja de acordo com as exigências da ação e da participação dos atores envolvidos no problema.

> A pesquisa-ação diferencia-se dos demais métodos qualitativos por oferecer bases e procedimentos para o pesquisador fazer intervenções num ambiente real e encontrar soluções práticas relevantes. O alto grau de envolvimento do pesquisador, tanto por intervir quanto por colaborar com os participantes (quando ele mesmo não é um deles), possibilita que ele obtenha mais informações sobre o ambiente e as utilize na sua pesquisa. Já a característica iterativa do método possibilita que o pesquisador reveja criticamente as ações já realizadas e redirecione o rumo da pesquisa conforme os resultados obtidos (FILIPPO, 2008, p. 31).

Em seu andamento, os pesquisadores recorrem a métodos e técnicas de grupo para lidar com a dimensão coletiva e interativa da investigação, técnicas de registro,

processamento, exposição de resultados, assim como, eventualmente, questionários e técnicas de entrevista individual como meio de informação complementar (THIOLLENT, 2005). Como método, atesta Brandão (1999), a pesquisa-ação, não se preocupa com a explicação dos fenômenos sociais após seu acontecimento, mas busca o caminho inverso e procura a aquisição do conhecimento durante o processo tido como de "transformação".

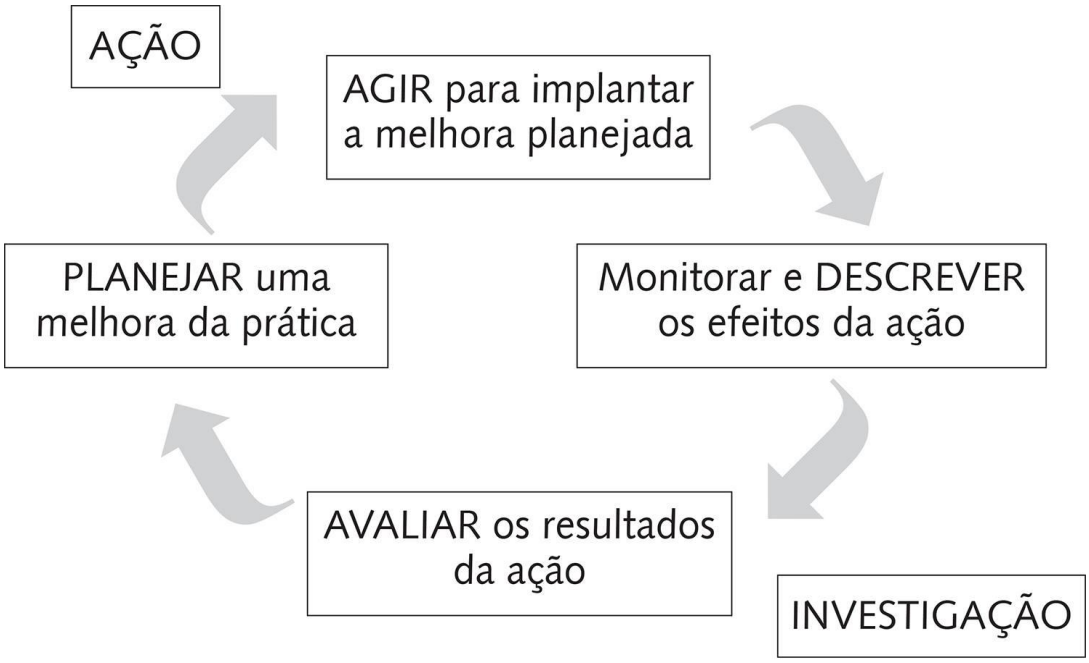


Figura 1 - Representação do ciclo básico de investigação-ação (TRIPP,2005)

Apesar de principiante nos ambientes tecnológicos e organizacionais, esta forma de pesquisa já possui relativa tradição nestes meios como uma forma de obtenção de informações, negociação de soluções para problemas de ordem técnico-organizativa e possibilidade de uma maior participação de atores sociais em processos de tomada de decisão (THIOLLENT, 2005). Tripp (2005) caracteriza a pesquisa-ação como um dos inúmeros tipos de investigação-ação (termo genérico para todo o processo que siga um ciclo no qual se aprimora a prática pela oscilação sistemática entre agir no campo da prática e investigar a

respeito da mesma), como por exemplo a aprendizagem-ação, a prática reflexiva, aprendizagem experimental, ciclo PDCA[3] dentre outras conforme a Figura1.

De fato, como demonstra Thiollent (1997), um dos fundamentos da pesquisa-ação está no constante *feedback* da informação produzida pela pesquisa. O autor apresenta um roteiro de quatro etapas conforme mostra a Figura 2 que, segundo ele, é um dos possíveis caminhos para este tipo de pesquisa.

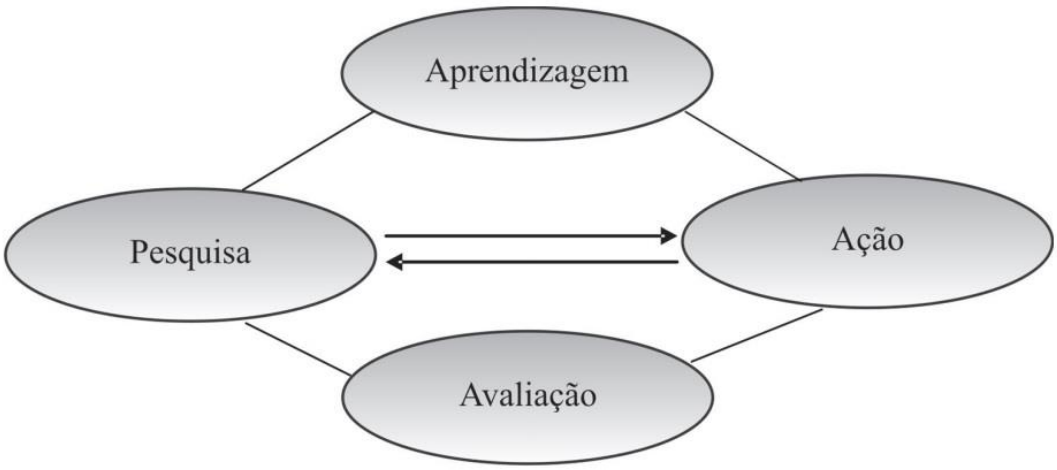


Figura 2 - Relações entre pesquisa, ação, aprendizagem e avaliação (THIOLLENT, 1997)

Entre os objetivos de conhecimento alcançáveis em pesquisa-ação, Thiollent (2005) aponta para:

a) A coleta de informação original acerca de situações ou atores em movimento;

b) A concretização de conhecimentos teóricos, obtida por meio do diálogo entre pesquisadores e participantes;

c) A produção de regras práticas ou direcionadores para solucionar problemas;

d) Possíveis generalizações estabelecidas a partir de pesquisas semelhantes e desta forma aprimora a experiência dos pesquisadores.

---

[3] PDCA - Também chamado de Ciclo de Deming ou Ciclo de Shewhart — é uma ferramenta de gestão que tem como objetivo promover a melhoria contínua dos processos por meio de um circuito de quatro ações: planejar (plan), fazer (do), checar (check) e agir (act).

Já para Tripp (2005) existem cinco modalidades de pesquisa-ação, ao se pensar sobre a natureza de um projeto desta característica:

1 – Pesquisa-ação técnica – constitui uma abordagem pontual na qual o pesquisador faz uso de uma prática existente e a implementa em sua própria esfera prática para realizar uma melhoria;

2 – Pesquisa-ação prática – o pesquisador escolhe ou projeta as mudanças;

3 – Pesquisa-ação política – o pesquisador exerce o "poder" trabalhando com ou contra os outros para mudar o sistema;

4 – Pesquisa-ação socialmente política – passa a existir quando se acredita que o modo de ver e agir "dominante" do sistema, dado como certo relativamente a tais coisas, é realmente injusto de várias maneiras e precisa ser mudado;

5 – Pesquisa-ação emancipatória – tem como meta explícita mudar o status quo não só para si mesmo e para seus companheiros mais próximos, mas de mudá-lo numa escala mais ampla, do grupo social como um todo.

Desta forma, a proposta metodológica elaborada nesta pesquisa, segue predominantemente a linha de pensamento de Brandão (1999) e procura a aquisição de novos conhecimentos durante o processo tido como de "transformação", onde certos aspectos de música e programação podem estar contidos de forma reciproca (alguns aspectos de música e programação podem ser relacionados, compartilhar similaridades em seus conceitos).

3.2 Lugar e público alvo da pesquisa

O processo exploratório foi executado no Departamento de Música da Universidade Federal da Paraíba, Campus I situada no bairro do Castelo Branco em João Pessoa. A Universidade Federal da Paraíba, anteriormente Universidade da Paraíba, é uma Instituição autárquica de regime especial de ensino, pesquisa e extensão, vinculada ao Ministério da Educação, com estrutura multi-campi e atuação nas cidades de João Pessoa, Areia, Rio Tinto e Mamanguape, e Bananeiras É voltada a educação superior, especializada na oferta gratuita de educação superior nas diferentes modalidades de ensino.

A cidade de João Pessoa tem aproximadamente 723.515 habitantes conforme o senso do IBGE realizado em 2010. O campus de UFPB iniciou suas atividades em João Pessoa no ano de 1947, com a fundação da Faculdade de Ciências Econômicas. O Departamento de Música da Universidade Federal da Paraíba (DeMús) foi fundado em 1978 quando também foi criado o Curso de Bacharelado em Música (BelMús).

O público alvo para o desenvolvimento e aplicação da proposta são alunos voluntários do curso sequencial superior de música com habilitação em regência de bandas marciais e instrumentos musical popular. A amostra foi de 11 alunos que participaram do curso de editoração de partituras com introdução a programação oferecido durante um semestre letivo.

3.3 Etapas da pesquisa

Nesta subseção serão apresentadas as etapas desta pesquisa.

Seguindo a metodologia adotada para esse trabalho, foi estruturado um roteiro para o desenvolvimento desta pesquisa. O roteiro elaborado é constituído por oito etapas e tem como base o esquema de McKay e Marshall (2001) apresentado na Figura 3.

A primeira etapa do trabalho consistiu em identificar o problema de interesse e partiu de uma inquietação pessoal e experiência vivida. Foi levantada uma hipótese que posteriormente tornou-se um projeto de pesquisa com intuito de verificar a possibilidade de ensinar introdução a programação a partir de uma abordagem interdisciplinar, proporcionando a músicos uma experiência de conhecer conceitos sobre programação e com isso, incentivá-los a aprofundar o conhecimento.

Seguindo com os passos elencados na Figura 3, realizou-se a busca alinhada com fatos que forneceram suporte à solução do problema identificado na primeira etapa. A partir de uma revisão bibliográfica acerca do tema, onde são apresentados problemas e possíveis soluções, buscou-se conhecer melhor como se dá o processo de ensino e aprendizagem dos conceitos iniciais de programação de computadores. Dentre as soluções propostas pela literatura, estão o uso de ferramentas e softwares que possibilitem maior interatividade por parte dos alunos. Desta forma, como a proposta desta pesquisa utiliza uma IDE como apoio, realizou-se um mapeamento sistemático para verificar quais as ferramentas utilizadas no ensino de programação.

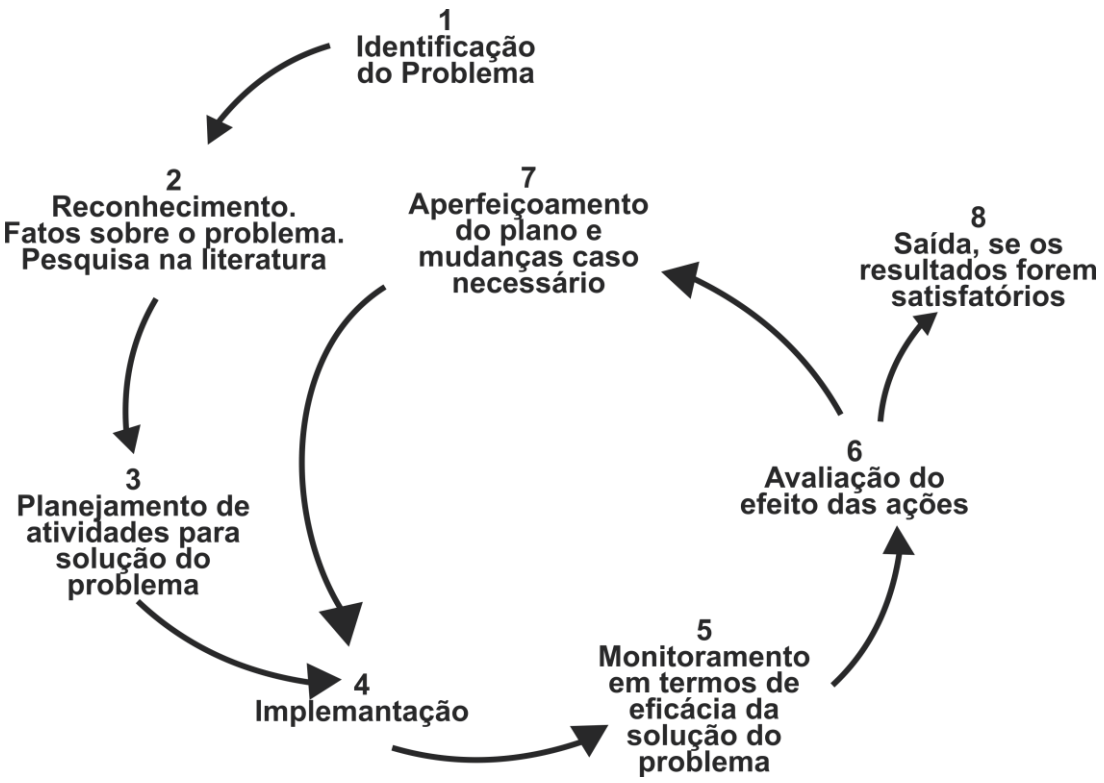


Figura 3 - Os passos de um projeto de Pesquisa-ação (Adaptado de McKay e Marshall 2001)

Segundo Petersen (2008) e Kitchenhan (2007), o processo de Mapeamento Sistemático da Literatura (MSL) tem como objetivo fazer uma pesquisa em largura, e não em profundidade. Por se basear em um protocolo de pesquisa, este mapeamento pode ser reproduzido por outros pesquisadores. Os itens que constituem este protocolo são: objetivos de busca, palavras-chave, filtros, mecanismos de busca acadêmica, critérios de exclusão/inclusão, e dados a serem levantados. A definição dos objetivos de busca institui o que se pretende encontrar na literatura através do mapeamento.

A terceira etapa do trabalho é composta por desenvolver um plano de ações para a solução do problema apresentado na primeira etapa. Sendo assim, elaborou-se uma proposta de ensino baseada na teoria da aprendizagem significativa, que tem como objetivo, o ensino introdutório de programação para músicos.

Seguindo o diagrama da Figura 3, a quarta etapa do processo de pesquisa foi a implementação em sala de aula, com o público alvo descrito no subitem 3.2 desta seção. No decorrer do período em que ela foi aplicada, observou-se a quinta etapa que consiste em

monitorar as ações implementadas para saber se os resultados encontrados estão de acordo com o que se esperava para a solução do problema.

Realizadas as devidas observações e avaliações do processo que se referem a sexta etapa, alguma ajustes foram realizados (etapa 7) e implementados (etapa 4) até que os objetivos traçados fossem alcançados, que se refere a oitava etapa. Durante a implementação da proposta foi realizada a coleta de dados que posteriormente serão descritos e analisados.

Na subseção seguinte serão descritos os instrumentos de coleta de dados, organização e análise dos dados obtidos durante a implementação da proposta.

3.4 Execução do projeto

Finalizada a elaboração da proposta, a execução prática da mesma aconteceu com o propósito de verificar os resultados.

As aulas aconteceram nas dependências do Centro de Comunicação, Turismo e Artes (CCTA), onde estão localizados os cursos de Bacharelado e Licenciatura em música, além do curso Sequencial em Regência de Bandas.

A proposta tem carga horária de 30 horas de aula (considerando a hora aula de 50 minutos). Para efetivação da mesma foram realizados 15 encontros com 2 horas cada, e início na terceira semana de julho de 2018.

A disciplina será ministrada pelo pesquisador sob a supervisão e auxilio do orientador da pesquisa.

3.4.1 Coleta e análise de dados

A coleta de dados para esta pesquisa foi realizada a partir de documentos utilizados no curso como o plano de curso e avaliações elaborados especificamente para a pesquisa, o desempenho dos estudantes na disciplina, com foco na análise qualitativa deste material. Além disso serão registradas pelo pesquisador observações da sala de aula e diálogos sem estruturas ocasionais com o professor orientador da disciplina durante a aplicação.

Os dados obtidos foram analisados para verificar se os objetivos de ensinar conceitos introdutórios de programação para músicos foi atingido.

Para realizarmos a análise dos dados obtidos, foram elaborados alguns critérios:

I. Cumprimento da ementa e objetivos da disciplina, observando se os tópicos previstos foram exercidos de forma:

   a. Parcial;

   b. Parcial com inserção de tópicos não contemplados na ementa.

   c. Plena;

   d. Plena com inserção de tópicos não contemplados na ementa.

II. Verificação dos conteúdos abordados em exercícios de fixação e trabalhos referentes da disciplina, observando se:

   a. Não foram incluídos;

   b. Incluídos parcialmente;

   c. Incluídos plenamente;

III. Observar do desempenho dos alunos, quanto a:

   a. Evolução semanal;

   b. Índice de aprovação da disciplina.

IV. Observações qualitativas do pesquisador a respeito do engajamento dos alunos e sua participação;

V. Observações do professor orientador da disciplina a respeito do desempenho dos estudantes.

Todos os critérios observados e os respectivos resultados serão descritos e analisados na seção seguinte.

Outro método utilizado durante a pesquisa é a observação participante. Trata-se de uma observação etnográfica no qual o observador participa ativamente nas atividades de recolha de dados, sendo requerida a capacidade do investigador se adaptar à situação (PAWLOWSKI, ANDERSEN, TROELSEN, & SCHIPPERIJN, 2016).

É um método que nos permite checar e obter dados de situações e eventos comuns que não são possíveis de captar través de entrevistas ou através de instrumentos de auto-avaliação. Segundo Vogt (1999), a Observação Participante é um tipo de investigação no qual:

> (...)um pesquisador participa como membro do grupo que ele ou ela é estudando. Às vezes o pesquisador informa ao grupo que ele ou ela é um observador, bem como um participante, e às vezes o pesquisador finge ser um membro comum. (VOGT, 1999, p. 208)

Enquanto método de observação direta, destaca-se a participante, a não participante e a geográfica ou exploração psicossociológica no terreno (Deshaies, 1997). Os investigadores são levados a partilhar papéis e hábitos dos grupos observados, estando assim em condições favoráveis para observar fatos, situações e comportamentos que não ocorreriam, ou que seriam alterados, na presença de estranhos (Deshaies, 1997).

> A Observação Participante é realizada em contato direto, frequente e prolongado do investigador, com os atores sociais, nos seus contextos culturais, sendo o próprio investigador instrumento de pesquisa. Requer a necessidade de eliminar deformações subjetivas para que possa haver a compreensão de fatos e de interações entre sujeitos em observação, no seu contexto. É por isso desejável que o investigador possa ter adquirido treino nas suas habilidades e capacidades para utilizar a técnica. (CORREIA, 1999, p. 31)

Inserida no conjunto das metodologias denominadas de qualitativas, a Observação Participante é utilizada em estudos ditos exploratórios, descritivos ou ainda, em estudos que visam a generalização de teorias interpretativas.

Portanto, definidos os procedimentos metodológicos e o capítulo seguinte descreve a proposta e a abordagem interdisciplinar das duas áreas, música e computação.

4 PROPOSTA

Neste capítulo será descrita a proposta, partindo da relação entre música e computação e posteriormente a descrição a ferramenta Frescobaldi que servirá de ambiente de aplicação dos conceitos.

4.1 A música e programação de computadores

Segundo Drapkin (2017), músicos parecem ter um talento natural para programar computadores. Ambas as habilidades lidam com conceitos abstratos, além disso, músicos e programadores conceituam suas mídias em uma organização hierárquica. Quando os músicos pensam sobre a Quinta Sinfonia de Beethoven, eles a veem como um trabalho singular que é dividido em diferentes movimentos, cada um dos quais é dividido em seções, frases, barras e notas únicas.

Os programadores pensam em seus programas de maneira semelhante. O software certamente é hierárquico, começando no topo com um sistema como "Contabilidade". Esse sistema é dividido por funções, como contas a pagar, contas a receber, razão geral, etc. Cada função pode ser dividida em subprogramas, como aquisição de dados, que são divididos em loops, blocos e linhas de código de programa. As palavras em uma linha individual de código de programa correspondem às notas em uma página de música. Os programas até têm repetições e primeiro e segundo finais na forma de loops e casos.

Historicamente, podemos observar que músicos já eram convidados a trabalharem empresas que desenvolviam softwares. Podemos observar isso em um cartaz da IBM - International Business Machines divulgado em 1968 (Figura 4), onde há um chamamento de profissionais de diversas áreas são convidados a trabalhar como programadores na empresa.

A interação entre o mundo da computação, programação e música continuou a se fortalecer e se mantem até hoje, promovendo mudanças profundas no mundo da música que vão desde a produção musical até linguagens de programação específicas para música.

Figura 4 - Cartaz de 1968 da IBM

No campo da indústria fonográfica o uso de computadores para gerar som evoluiu e formatos, mídias, performance e distribuição de música forçaram a indústria musical a seguir essa tendência crescente e se adaptar. Estas novas tecnologias na manipulação de áudio não são as únicas mudanças que a indústria da música viu, e vão muito além da gravação de áudio. A produção musical também evoluiu e artistas independentes não vinculados a grandes gravadoras estão em um campo de jogo mais nivelado. O custo com ferramentas de gravação e distribuição de música fez com que qualquer pessoa possa alcançar grandes públicos.

No que diz respeito a distribuição, algoritmos foram desenvolvidos por programadores que se especializaram em engenharia de áudio, demandas de usuários, IHC, entre outras para estudar os gostos musicais dos usuários e entregar músicas personalizadas

sob demanda. Essa personalização baseada em algo como preferências de escuta, gênero, idade e localização desempenham um papel importante na entrega da música. Usuários também podem fazer suas próprias personalizações quando se trata do que ouvem on-line. Embora as listas de reprodução personalizadas sejam recomendadas a elas por meio de algoritmos complexos, os usuários não são impedidos de sair da bolha, conectar-se a novos artistas e desenvolver um novo gosto pela música.

O *softwares* presentes no meio musical tem as mais distintas finalidades e agrupados conforme a sua utilidades. Nesta pesquisa focamos a utilização de uma ferramenta pertencente ao grupo de editores de partitura, pois tem como saída principal uma partitura musical alta qualidade gráfica. Podemos destacar outros grupos e *softwares* como gravadores e editores de áudio, *softwares* sintetizadores, *softwares* voltados para produção sonora, distribuição, entre outros.

Na área da programação algumas linguagens de programação também surgiram com a finalidade de ajudar compositores, designers de software, pesquisadores e artistas para criar gravações, performances, editorações de partituras. Diversas são as formas de interação com essas linguagens. Como exemplo, podemos citar o Max, que é uma linguagem de programação visual para as necessidades especializadas de artistas, educadores e pesquisadores que trabalham com áudio, mídia visual e computação física e tem como base o sistema de fluxo de dados. Programas Max (*patches* nomeados) são feitos organizando e conectando blocos de construção de objetos dentro de um *patcher*, ou blocos visuais. Esses objetos atuam como programas autocontidos (na verdade, são bibliotecas vinculadas dinamicamente), cada uma das quais pode receber entrada (por meio de uma ou mais entradas visuais), gerar saída (por meio de saídas visuais) ou ambos. Objetos passam mensagens de suas saídas para as entradas de objetos conectados.

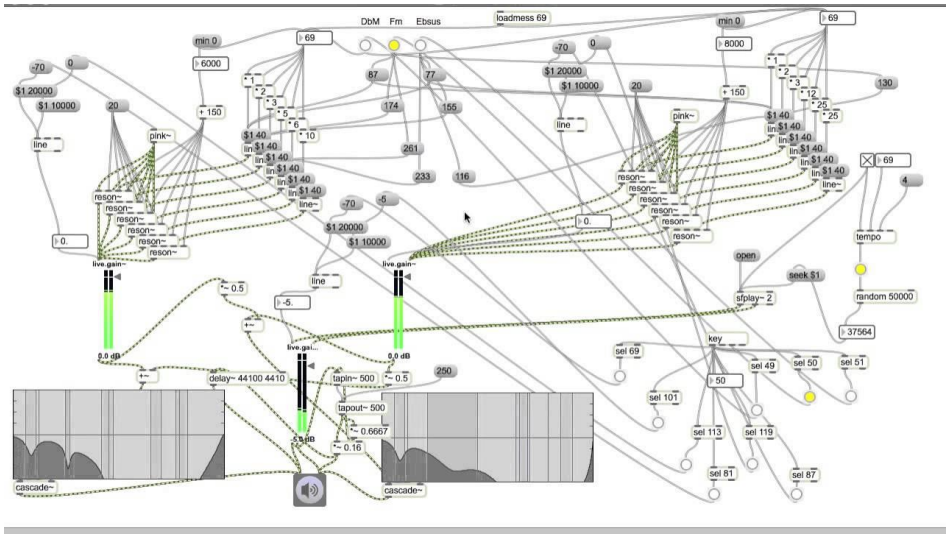


Figura 5 - Código de uma música criada com o MAX

Outra linguagem é o Pure Data ou simplesmente Pd, é um ambiente de programação gráfica para áudio e vídeo usado como ambiente de composição interativo e como estação de síntese e processamento de áudio em tempo real. O Pure Data é comumente utilizado para performances musicais ao vivo, VeeJaying, efeitos sonoros, composição, análise de áudio, interação com sensores e câmeras, controlar robôs e até interagir com websites. A mais básica unidade de funcionalidade é uma caixa e o programa é criado a partir da conexão destas caixas juntas em diagramas de modo que ambos representam o fluxo dos dados enquanto realizam atualmente as operações mapeadas nos diagramas.

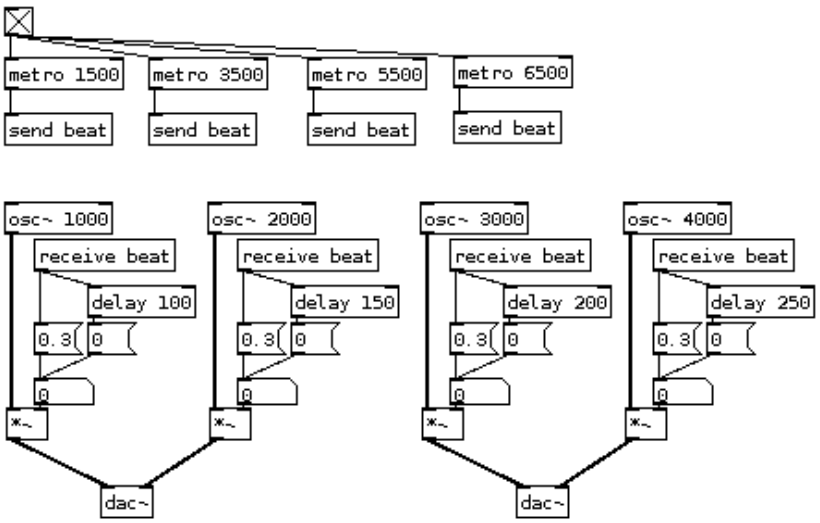


Figura 6 - Código desenvolvido com o Pd.

Além de linguagens de programação visuais para a área da música, também temos a linguagem de programação LilyPond que é a baseada em TeX, que permite a escrita de partituras musicais com base numa filosofia de linguagem de marcação. O LilyPond é a linguagem que usaremos nesta proposta e será detalhada na seção 4.3. A escolha do LilyPond se deu com base nos conhecimentos sobre conceitos elementares de teoria musical. Além disso, como a proposta sugere uma comparação com os cursos tradicionais que fazem o uso de linguagens como C, C++, Java, entre outros, julgamos importante a utilização do LilyPond pois tem uma paradigma semelhante. Para utilizar o Pure Data é necessário que o interessado conheça conceitos básicos de síntese sonora, *midi*, entre outros conhecimentos que normalmente só músicos com mais experiência na área de composição e produção musical possuem.

4.1.1 Relação entre conteúdo de música e programação de computadores

Para começarmos a relacionar os termos das duas áreas, olhamos para a fundamentação teórica do trabalho e temos que um programa de computador é uma sequência de processos realizados a partir da entrada de dados até a produção do resultado desejado. Essa sequência de passos é denominada de algoritmo.

Podemos observar que a construção de um algoritmo leva em consideração três blocos de instruções que são:

I. Sequência – para a execução de uma tarefa, precisamos seguir uma sequência de instruções que nos levarão a um resultado, exatamente como as instruções do programa;

II. Repetições – num algoritmo temos instruções que se repetem até que uma condição seja satisfeita. Da mesma forma acontece em um programa, onde um grupo de declarações são executadas N número de vezes até que a sentença seja verdadeira ou falsa.

III. Seção – dentro de um algoritmo existem seções que podem ou não ser executadas, dependendo dos desvios que possam existir. Dentro de uma programa, se uma condição ocorrer em tempo de execução, pula-se para uma parte diferente identificada do programa.

Partindo para a música temos os mesmo elementos relacionados acima e que se apresentam de forma semelhante:

I. Sequência – quando o músico começa a tocar uma partitura ele deve seguir a sequência, nota após nota; compasso seguido de outro compasso e assim sucessivamente. Quando uma orquestra for tocar, não faz sentido um determinado músico pular compassos ou notas, pois deixará de existir coerência na música.

II. Repetições – Na música também existem repetições de determinados trechos a depender das condições colocadas pelo compositor ou arranjador. Na música popular é comum encontrarmos seções ou frases que se repetem por várias vezes até que a partitura indique uma próxima seção ou o maestro assim o faça.

III. Seção – dentro da música também encontramos seções e que são executadas conforme determinação do compositor e isso acontece quando símbolos colocados nas partituras indicam esses desvios. Indicações como D.S. al Coda ou D.S. al Fine são exemplos de desvios para próximas seções.

Podemos assim então fazer analogia das palavras reservadas das linguagens de programação com os símbolos encontrados nas partituras.

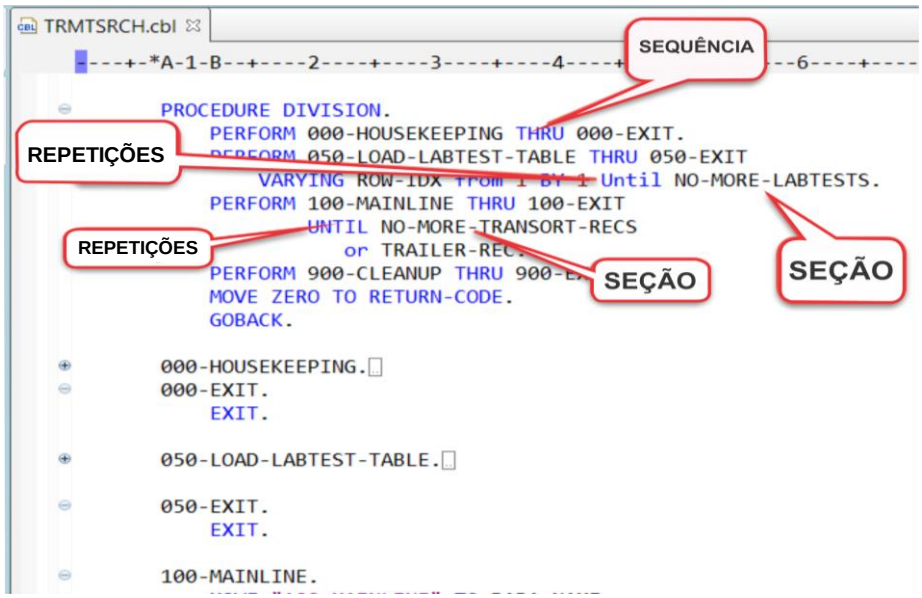


Figura 7 - Código com as referências similares a música

Seguindo com as relações entre música e computação podemos citar os vetores. Vetores são variáveis compostas que podem armazenar um conjunto de valores. Todos estes valores são referenciados através do nome do vetor (o mesmo para todo o conjunto de valores) e de um índice (distinto para cada valor). Relacionando com a música temos a escala, que podemos considerar dessa forma um vetor, onde cada nota torna-se um índice. Se tomarmos a escala[4] de DÓ MAIOR, temos um vetor com 7 valor de índice 6 onde podemos acessar o valor de cada índice de forma aleatória ou sequenciada. Neste vetor os índices ficariam conforme a figura abaixo:

VALOR ⇨	DO	RE	MI	FA	SOL	LA	SI
INDICE ⇨	0	1	2	3	4	5	6

Quadro 4 - Vetor ou array com variáveis

Outra possível relação que podemos fazer com vetor são as teclas de um piano, onde neste caso, cada tecla do piano corresponderia a um valor de determinado índice.

Em programação é comum usarmos o MACRO, ou seja, definir algumas funções no início do nosso programa para que o código seja mais limpo e de fácil manutenção. Quando usamos o *#define* na linguagem C por exemplo, estamos definindo alguma função ou atribuindo um valor a alguma variável. Quando você está escrevendo instruções de um programa que fazem referência a uma macro, as instruções agrupadas na macro são chamadas. Mas como desenvolvedor ou programador de software você deve poder expandir visualmente a Macro (pelo menos até certo ponto) para entender sua função.

Se lançarmos esse olhar sobre a música, temos algo semelhante acontecendo. Ao considerar um acorde de guitarra, por exemplo C, o guitarrista segue a partitura e interpretam mentalmente o símbolo C como sendo a seguinte sequência de notas em ordem ascendente: C – E – G, exatamente da mesma maneira que um desenvolvedor analisa e interpreta um macro.

[4] Escala - Sequência ordenada de tons pela frequência vibratória de sons, que consiste na manutenção de determinados intervalos entre as suas notas.

De posse destas relações estabelecidas, a teoria da aprendizagem significativa pode ser aplicável a essa proposta, pois existe a possibilidade de músicos estudarem programação de forma mais significativa, por já possuírem o conhecimento de música estabelecido. De forma semelhante, podemos fazer o caminho inverso, onde programadores podem aprender teoria musical através do uso de cógido ou em Inglês americano "through coding".

4.2 Lilypond / Frescobaldi

Para compreendermos melhor o software Frescobaldi, precisamos entender o sistema que serve de alicerce para o mesmo que é denominado LilyPond. O LilyPond, por sua vez é um projeto livre e de código aberto e está publicado sob a Licença Pública Geral do projeto GNU e sua documentação possui aproximadamente 8000 páginas e está traduzida para 7 idiomas.

O LilyPond não é uma aplicação fácil de se usar, pois sua interatividade é baseada em um documento de texto, a partir do qual, depois de compilado, o resultado é a produção de arquivos pdf, midi, png, svg, entre outros. Para utilizar a ferramenta LilyPond, o primeiro requisito é a instalação da aplicação em qualquer sistema operacional, executando-se um arquivo.

O LilyPond é um programa que tem uso e aplicação de terminal. Isso significa que o programa não possui uma seção gráfica. Trabalhar com a linha de comando não é uma tarefa muito simples, pois requer do usuário certo conhecimento de códigos e sintaxes. A maioria das operações, principalmente em Linux podem ser feitas usando a linha de comando. Embora haja ferramentas gráficas para a maioria dos programas, às vezes eles não são suficientes. O terminal é chamado frequentemente de linha de comando ou shell. Até pouco tempo, esta era a maneira que o usuário interagia com o computador; entretanto, muitos usuários viram que o uso do shell ou linha de comando pode ser mais rápido do que um método gráfico.

Para usar o LilyPond, o usuário escreve um arquivo de texto que é O programa. O arquivo contém uma descrição formal de uma música, similar a um linguagem de programação, contendo bibliotecas, estruturas, funções, variáveis, entre outras. Fora essa descrição, o programa pode criar arquivos pdf, midi, png, svg, entre outros que o usuário pode então imprimir ou compartilhar.

Posteriormente serão apresentadas algumas interfaces do LilyPond que nos ajudarão compreender melhor o funcionamento do mesmo. As interfaces apresentadas são executadas na plataforma MacOS e funcionam de forma similar em outros sistemas operacionais.

Na imagem abaixo podemos observar a interface gráfica do LilyPond que baseia-se em um editor de texto. Nesse editor de texto são inseridas as informações necessárias e presentes em uma partitura musical, como o tipo de fonte, o título, o compositor, o instrumento, as notas, etc.

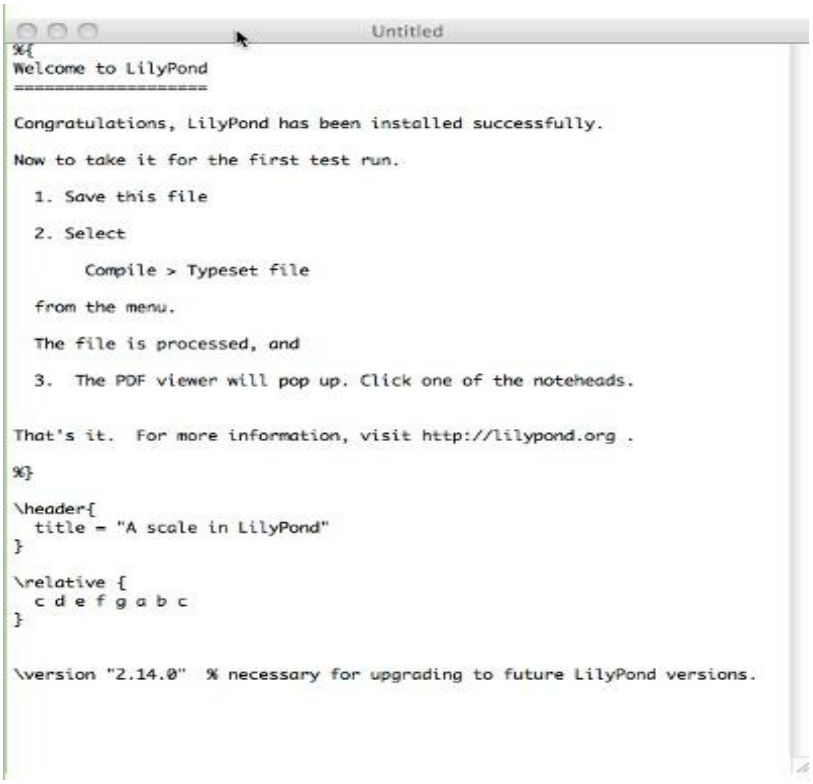


Figura 8 - Editor de texto do LilyPond

Partindo desse ponto a partitura está pronta para ser salva, porém não temos neste momento a partitura em uma linguagem que os músicas a compreendam. O arquivo que temos é um código de programação, contendo todas as sintaxes que um programa qualquer teria, como variáveis, laços de repetição, estruturas de controle, funções, entre outros e é salvo com uma extensão ly.

A saída gerada pelo processo de compilação pode ser um documento em formato pdf que poderá ser impresso, ou então arquivos midi, png, svg que é a compilação do código da

Figura 8. Analisando a interface apresentada na Figura 8, podemos observar que a utilização do LilyPond não é muito intuitiva e conceitos importantes de IHC, como usabilidade, não são contemplados neste aplicativo, portanto requer do usuário um conhecimento avançado sobre programação.

Para que possamos ter a partitura propriamente dita, esse código precisa ser compilado, processo esse mostrado pela Figura 9. A Figura 10 mostra a saída gerada pelo LilyPond, que nesse caso foi um PDF.

Figura 9 - Interface de LilyPond

Figura 10 - PDF gerado como saída gerada pelo LilyPond

Para a proposta da pesquisa que objetiva introduzir práticas com o software livre Frescobaldi como ferramenta de apoio ao ensino de linguagem de programação para músicos através de uma metodologia baseada na Teoria da Aprendizagem Significativa, a ferramenta

LilyPond traz algumas barreiras que acabam desestimulando o aluno a estudar programação, pois ao levarmos em consideração os estudos em IHC, interfaces pouco amigáveis são aspectos que afastam a audiência.

Com o propósito de facilitar o uso desta ferramenta, desenvolvedores criaram um interface gráfica para o LilyPond, chamada de Frescobaldi. Essa nova aplicação segue os mesmos princípios do LilyPond e requer do usuário a criação de um código de programação que irá gerar um arquivo pdf, midi, png, svg, entre outros. Os requisitos necessários para rodar esta aplicação são similares ao LilyPond.

A Figura 11 mostra um panorama geral da interface gráfica do Frescobaldi onde é possível observar uma série de recursos além do simples editor de texto utilizado pelo LilyPond. O recursos disponíveis no Frescobaldi são recursos que estão disponíveis em qualquer outro editor comercial de partituras, o que facilita o uso e atende a várias premissas de IHC e usabilidade.

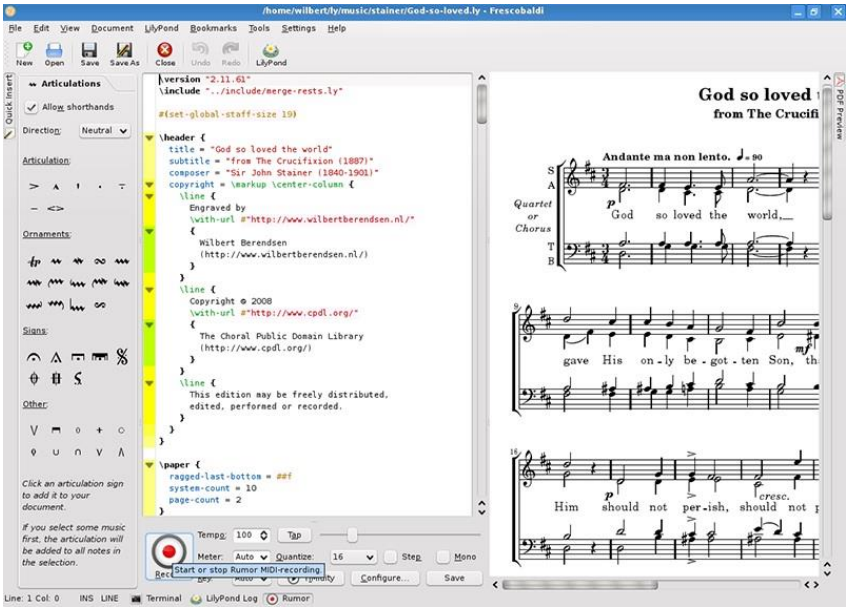


Figura 11 - Interface gráfica do Frescobaldi

Podemos destacar alguns pontos importantes do Frescobaldi que tem um editor de texto com destaque de sintaxe e preenchimento automático, visualização da música com recurso de apontar e clicar avançado, tocador MIDI para prova de escuta dos arquivos MIDI gerados pelo LilyPond, assistente de notas para configurar uma partitura musical, gerenciador

de recortes para armazenar e aplicar fragmentos de texto, de modelos ou de scripts, navegador da documentação e ajuda do LilyPond embutidos, interface de usuário com cores, fontes e atalhos de teclado configuráveis.

O software também possui um outro ponto a ser destacado e que facilita o uso da ferramenta que é a tradução para vários idiomas incluindo o Holandês, Inglês, Francês, Alemão, Italiano, Tcheco, Russo, Espanhol, Galício, Turco, Polonês, Português do Brasil e Ucraniano. Além das configurações de interface que o Frescobaldi possui, inúmeras características musicais também podem ser editadas como:

I. Mudar o tom da música;

II. Mudar a música do relativo para o absoluto e vice-versa;

III. Mudar o idioma utilizado para o nome das notas;

IV. Mudar o ritmo (dobrar, reduzir à metade, adicionar ou remover pontos, colar) etc;

V. Hifenizar a letra usando dicionários de hifenização de processadores de texto;

VI. Adicionar facilmente chaves, dinâmica, articulação usando o painel Quick Insert;

VII. Atualizar a sintaxe do LilyPond usando o convert-ly, com exibição das diferenças, entre outros.

Figura 12 - Assistente MIDI

As figuras 12, 13 e 14 mostram alguns recortes da interface do Frescobaldi e demonstram a melhora nos componentes gráficos e de interação do software. Outro ponto importante a ser observado é que existe uma característica interessante nesse ambiente de desenvolvimento, chamado "Ponto e clique", que consiste na vinculação do código fonte com

a pré-visualização do PDF, de modo que quando o cursor é movido sobre o texto, são destacados os elementos gráficos no PDF e, clicando no PDF, os elementos de texto do código-fonte que produziu esse elemento são destacados.

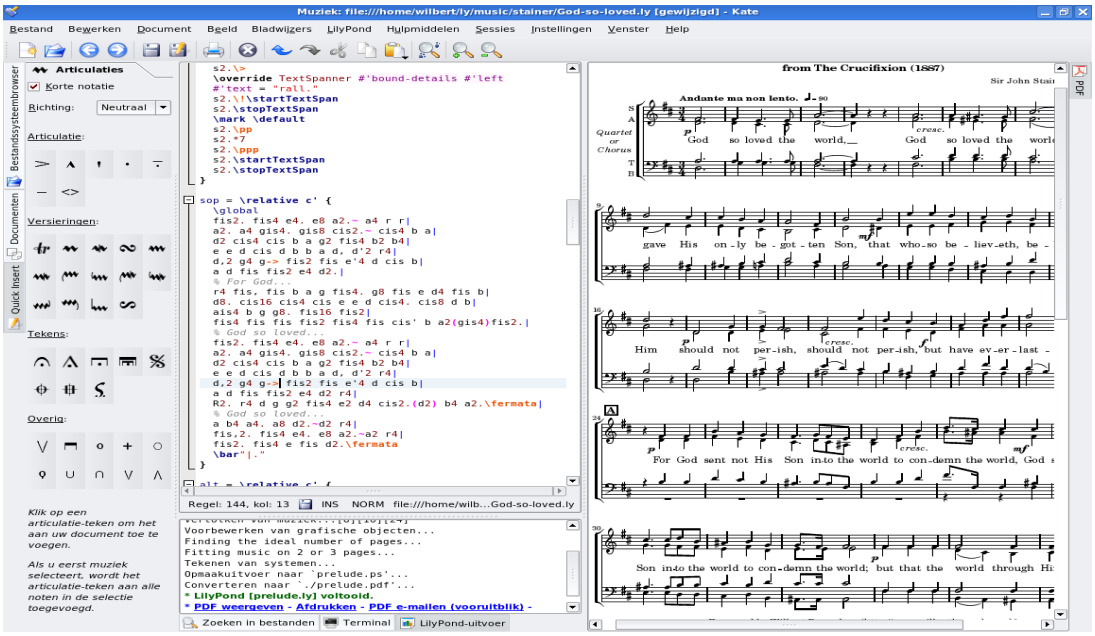


Figura 13 - Quick Insert com assistente de notas e articulações

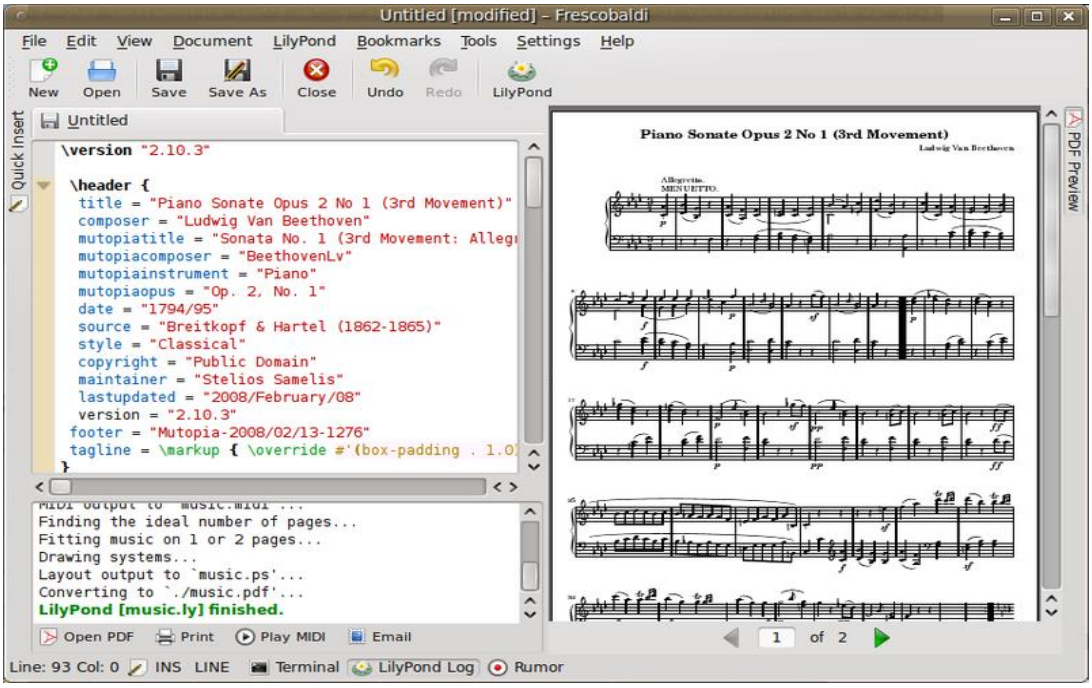


Figura 14 - Coloração de sintaxe e informações de debugging

4.3 Roteiro de curso e de aulas

O plano de curso e os planos de aula que serão apresentados no decorrer desta seção, foram desenvolvidos tendo como base a proposta objeto deste estudo. Para cada aula tem uma carga horária de duas horas aula preferencialmente em um laboratório de informática. O espaço onde a pesquisa aconteceu, em um centro de artes e comunicação, não oferece um laboratório de informática, sendo assim, necessário que os alunos levem para as aulas um computador pessoal onde irão praticar os conteúdos repassados.

A estrutura dos roteiros foi inspirada no trabalho de (FELDER; BRENT, 2003) e contém os objetivos da aula, conteúdos e atividades a serem realizadas. Cada aula não trata de um assunto específico devido a integração dos conteúdos, também não se estabelece rigidamente o tempo que o professor irá usar para sua aplicação, pois depende de uma série de fatores como por exemplo a turma, a discussão dos assuntos, entre outros.

Os conteúdos abordados nestes roteiros com a proposta são: História da computação; Software x Hardware; Algoritmos; Entrada e Saída de dados; Variáveis; Estrutura de decisão; Estrutura de repetição; Implementação de problemas na linguagem de programação LilyPond.

Segundo a estrutura curricular constante nas Diretrizes Curriculares do MEC (BRASIL, 2001), a matéria de Programação faz parte da área de formação básica em Ciência da Computação, juntamente com as matérias Computação e Algoritmos e Arquitetura de Computadores. Seu conteúdo abrange, além do ensino de linguagens de programação propriamente ditas, conceitos, princípios e os modelos de programação; estudo de estruturas de dados e de métodos de classificação e pesquisa de dados.

A seguir é apresentado o plano de curso e na sequencia os roteiros de quinze aulas com a proposta.

PLANO DE CURSO	
Disciplina: Algoritmos e Introdução a Programação – Relação entre programação e música. Monitor: Lucas Blatt Professor Orientador: Alexandre Magno Ferreira e Silva Carga Horária: 30 horas Semestre: 2018-2	
Ementa Geral História do Computador; Introdução aos algoritmos; Diferença entre Software e Hardware; Tipos básicos, variáveis e constantes; Comandos de atribuição, entrada e saída de dados; Estruturas de controle: sequencial, condicional e de repetição; Inclusão de arquivos esternos.	
Objetivos Apresentar ao aluno as noções mais básicas de programação de computadores relacionados à música através do ensino de algoritmos com o uso de uma linguagem e ambiente de programação. O Aluno deverá desenvolver as habilidades de programação com estruturas de controle a partir do uso de uma linguagem de programação moderna.	
Bloco	Conteúdo
Introdução	• Porque aprender a programar? • História do Computador • Introdução a algoritmos • Software x Hardware • O que significa programar um computador • Código fonte e código objeto • Programas são escritos em linguagens formais • Como obter um ambiente de programação
Primeiros programas	• Escrevendo seu primeiro programa • Identificadores • Identificadores formados por várias palavras

	• Espaçamento e endentação • Comandos • Palavra chave
Variáveis	• Significado de uma variável • Atribuição • Declaração de variáveis
Estruturas	• Estrutura de repetição • Aninhamento • Formas alternativas de repetição
Classes	• Objetos e classes • Atributos e associações
Metodologia Os alunos receberão material para estudar previamente. Durante as aulas haverá exposição e discussão sobre os assuntos previstos para o dia bem como a realização de exercícios em sala.	
Avaliação: A avaliação do aluno será feita continuamente pela realização exercícios de fixação e avaliações de verificação compostas por uma prova e um trabalho onde serão observados todos os tópicos trabalhados em sala de aula.	

Quadro 5 - Plano de curso

<table>
<tr><th colspan="2">AULA 01</th></tr>
<tr><td colspan="2">Objetivos:
Apresentar a disciplina, seus objetivos e ementa.
Apresentar as formas de avaliação que serão desenvolvidas durante a disciplina.
Apresentar a importância de software livre
Apresentar alguns pontos importantes sobre a importância do estudo de programação.
Apresentar superficialmente o aplicativo Frescobaldi que será utilizado na disciplina.</td></tr>
<tr><th>Conteúdo</th><th>Atividades</th></tr>
<tr><td>Apresentação</td><td>Iniciar a aula apresentando o professor e em slide, a disciplina, a ementa e os objetivos desta disciplina.</td></tr>
<tr><td>Métodos de Avaliação</td><td>Apresentação dos slides onde estão dispostas as ferramentas de avaliação, que serão:
Uma prova; Um trabalho; Participação das aulas; Entrega de exercícios de revisão.</td></tr>
<tr><td rowspan="2">Software Livre / Pirata</td><td>Falar sobre o risco de se utilizar uma cópia pirata ou não registrada de uma ferramenta para realizar seus trabalhos.</td></tr>
<tr><td>Falar sobre a vantagem e as possibilidades de se utilizar softwares livres e abertos no seu dia a dia.</td></tr>
<tr><td>Por que Programar?</td><td>Com o avanço da tecnologia, aprender a programar possibilita um leque de opções para qualquer profissional de qualquer área, além de possibilitar uma série de ações pontuais no dia a dia que demandam algum conhecimento de programação.</td></tr>
<tr><td>Frescobaldi</td><td>Apresentar de forma rápida a ferramenta Frescobaldi que será utilizada como IDE em nosso curso, mostrando alguns exemplos. (Observe que os códigos tem comandos familiares aos músicos)</td></tr>
<tr><td>Encaminhamentos</td><td>Na próxima aula vamos falar um pouco sobre a história do computador, de como surgiu até os dias atuais.</td></tr>
</table>

Quadro 6 - Plano da 1ª aula

<table>
<tr><th colspan="2">AULA 02</th></tr>
<tr><td colspan="2">Objetivos:
➢ Apresentar o histórico e evolução dos Computadores;
➢ Apresentar a definição de computadores.</td></tr>
<tr><th>Conteúdo</th><th>Atividades</th></tr>
<tr><td>Conhecimentos prévios</td><td>Questionar aos alunos quais conhecimentos que eles possuem sobre a computação ou programação de computadores.</td></tr>
<tr><td rowspan="5">História do computador</td><td>Como surgiram os computadores</td></tr>
<tr><td>As primeiras máquinas de calcular</td></tr>
<tr><td>Computadores mecânicos</td></tr>
<tr><td>Computadores eletromecânicos</td></tr>
<tr><td>Computadores eletrônicos</td></tr>
<tr><td>O que é Computador?</td><td>Apresentar a definição de computadores e como funcionam hoje</td></tr>
</table>

Quadro 7 - Plano da 2ª aula

<table>
<tr><th colspan="2">AULA 03</th></tr>
<tr><td colspan="2">Objetivos:
➢ Aplicação dos computadores;
➢ Apresentar os conceitos de tecnologia.</td></tr>
<tr><th>Conteúdo</th><th>Atividades</th></tr>
<tr><td>Conhecimentos prévios</td><td>Questionar aos alunos quais conhecimentos foram apresentados até o momento e anotar em local visível para os alunos.</td></tr>
<tr><td rowspan="2">Aplicação dos computadores</td><td>O que faziam os primeiros computadores?</td></tr>
<tr><td>Quantas tarefas faziam os primeiros computadores?</td></tr>
</table>

	Por que os computadores atuais são multitarefa?
Tecnologia	O que é tecnologia?
	Onde ela é empregada?
	A tecnologia na música.

Quadro 8 - Plano da 3ª aula

AULA 04	
Objetivos: ➢ Apresentar os conceitos básicos sobre algoritmo; ➢ Para que serve o algoritmo; ➢ Como os algoritmos podem ser apresentados;	
Conteúdo	Atividades
Algoritmo	O que é algoritmo?
	Para que serve o algoritmo?
	Como o algoritmo pode ser apresentado?
	Construir um algoritmo. Abrir espaço para que cada aluno possa construir um algoritmo sobre um determinado tema, preferencialmente um algoritmo que envolva a música. (rotina de estudo de instrumento, manutenção de um instrumento musical, estudo de grade, estudo de uma partitura, estudos de aquecimento,...)

Quadro 9 - Plano da 4ª aula

AULA 05
Objetivos: ➢ O que é refinamento de algoritmo.

Conteúdo	Atividades
Algoritmo	Refinamento de algoritmo
	Construir um algoritmo Construir uma algoritmo com o tema lubrificação de instrumento e prever possíveis situações que possam ocorrer no decorrer da lubrificação dos pistons de um trompete.
	Refinar um algoritmo.
	Refinar o algoritmo construído na aula anterior. Abrir espaço para que cada aluno possa refinar o algoritmo construído na aula anterior sobre um determinado tema, preferencialmente um algoritmo que envolva a música.

Quadro 10 - Plano da 5ª aula

AULA 06	
Objetivos: ➢ Apresentar os conceitos sobre *software* e sua função ➢ Apresentar os conceitos sobre *hardware* e sua função ➢ Entender como *software* e *hardware* se relacionam	
Conteúdo	Atividades
Sistema	O que é software?
	Que tipo de software existe?
	Software de Sistema
	Software de Programação
	Software de Aplicação
	Livre, gratuito, pago e demo

Hardware	O que é hardware?
	Como funciona.
	Diferentes tipos de Hardware
Software e Hardware	Como se relacionam

Quadro 11 - Plano da 6ª aula

AULA 07	
Objetivos: ➢ Realizar o download e instalação do pacote *LilyPond* ➢ Realizar o download e instalação da IDE *Frescobaldi* ➢ Configuração de preferências do *LilyPond* ➢ Apresentar a interface gráfica do *Frescobaldi* ➢ Apresentar alguns comando básicos iniciais do *Frescobaldi*	
Conteúdo	Atividades
LilyPond / Frescobaldi	Acessar a *Homepage* do *Lilypond* e realizar o download da versão estável
	Acessar a *Homepage* do *Frescobaldi* e realizar o download da versão estável
	Instalar o pacote *LilyPond* e o *Frescobaldi*
	Configurar algumas preferências do *Frescobaldi*
	Apresentar alguns aspectos importantes da interface gráfica do *Frescobaldi*. (Janelas, Atalhos, Documentação, Snippets)
	Coloração de Sintaxe; Endentação; Numeração das linhas de Códigos.
	Alguns Códigos básicos da linguagem Notação

	c d e f g a b Duração *1, 2, 4, 8, 16, 32,* *ponto de aumento* Pausas *r* Acidentes *s – sustenido (sharp)* *b – bemol (flat)* *ss – dobrado sustenido* *bb – dobrado bemol*

Quadro 12 - Plano da 7ª aula

AULA 08	
Objetivos: ➢ Apresentar a linguagem LilyPond; ➢ Apresentar os casos sensíveis; ➢ Apresentar as palavras – chave; ➢ Apresentar sua sintaxe.	
Conteúdo	Atividades
Linguagem de Programação	O que é e como funciona?
Linguagem LilyPond	O que é linguagem LilyPond
	Quando surgiu
	Como é a sintaxe
	Casos sensíveis *Nota* — *Entrada Lilypond (caso sensível)* *C* — *c*

	D	*d*
	E	*e*
	F	*f*
	G	*g*
	A	*a*
	B	*b*
	Uso da Versão *\version "2.18.2"*	
	Realizar exemplos. *\version "2.18.2"* *{* *c d e f g a b* *}*	
	Comparar a estrutura de um programa com a estrutura de uma partitura, respeitando as regras e hierarquias. Comparar a sintaxe de uma linguagem de programação qualquer com a sintaxe da linguagem musical.	

Quadro 13 - Plano da 8ª aula

AULA 09	
Objetivos: ➢ Apresentar palavras – chave ➢ Apresentar funções	
Conteúdo	Atividades

Palavra Chave / Função	Tempo - Para definir o tempo usamos o comando time. \time 4/4, \time ¾; Andamento - Indicação de tempo e marca de metrônomo podem ser definidas com o comando tempo. \time “andante”; \tempo “presto” 4=120; Clave - Indicação de clave pode ser definida usando o comando clef. \clef “bass”; Armadura de Clave - Indicação de tonalidade é definida usando o comando de chave seguido por um campo e maior ou menor. \key d \major; Modo Relativo - A oitava é escolhida automaticamente, assumindo-se que a nota seguinte seja sempre colocada mais próxima da nota anterior.
	Exemplo: *Ctrl + shift + v;* *\language “english”* *\relative c’* *{* *cs4 d8 ef f g a4 f16 g a e a a4 d f* *\time 3/2* *e2 b f* *\clef alto\numericTimeSignature \time 4/4* *c1}*

Quadro 14 - Plano da 9ª aula

AULA 10
Objetivos: ➢ Apresentar palavras – chave; ➢ Apresentar a forma de inserção articulações, dinâmicas e marcas de ensaio; ➢ Apresentar funções.

<table>
<tr><th>Conteúdo</th><th>Atividades</th></tr>
<tr><td rowspan="2">Função</td><td>• \header {...}
Introduzir bloco que contenha informações de cabeçalho da partitura como compositor, arranjador, título e outras informações pertinentes</td></tr>
<tr><td>Exemplo:
Ctrl + shift + v;
\language “english”
\header { title = “XXXXX”
composer = “XXXXXXX”
arranger = “XXXXXXX”
}</td></tr>
<tr><td rowspan="2">Sintaxe / Palavra chave</td><td>• Articulações;
• Dinâmicas;
• Marcas de ensaio;
• Conhecendo o QUICK INSERT;
• Ligaduras;
• Acordes;
• Texto na Partitura.</td></tr>
<tr><td>Exemplo:
Ctrl + shift + v;
\language “english”
\header {
title = “XXXXX”
composer = “XXXXXXX”
arranger = “XXXXXXX”
}
\relative c’ { cs4 d8 ef f g a4 f16 g a e a a4 d f
\time 3/2
e2 b f \clef alto\numericTimeSignature \time 4/4 c1}</td></tr>
</table>

Quadro 15 - Plano da 10ª aula

<table>
<tr><th colspan="2">AULA 11</th></tr>
<tr><td colspan="2">Objetivos:
➢ Apresentar uma forma para criação rápida de partituras (SETUP WIZARD);
➢ Apresentar os tipos de saída do Frescoballdi;
➢ Criação de quiálteras;
➢ Apresentar a criação de atalhos.</td></tr>
<tr><th>Conteúdo</th><th>Atividades</th></tr>
<tr><td rowspan="2">Saída de Dados</td><td>Elaborar alguns exemplos e explorar as várias saídas que o programa oferece (midi, pdf, svg, png, xml)</td></tr>
<tr><td>Exemplo:
Ctrl + shift + v;
\language “english”
\header {title = “XXXXX”
composer = “XXXXXXX”
arranger = “XXXXXXX”}
{ a2 \tuplet 3/2 { b4 b b } c4 c \tuplet 3/2 { b4 a g }
c4 \tuplet 5/4 { f8 e f \tuplet 3/2 { e[f g] } } f4}</td></tr>
<tr><td rowspan="2">Funcionalidades do Frescobaldi</td><td>Criar partituras utilizando o SETUP WIZARD
Conhecer o banco de atalhos que o programa oferece;
Criar novos atalhos
Código para inserir quiálteras na partitura;</td></tr>
<tr><td>Exemplo:
Ctrl + shift + v;
\language “english”
\header { title = “XXXXX”
composer = “XXXXXXX”
arranger = “XXXXXXX”}
{ a2 \tuplet 3/2 { b4 b b } c4 c \tuplet 3/2 { b4 a g }
c4 \tuplet 5/4 { f8 e f \tuplet 3/2 { e[f g] } } f4 }</td></tr>
</table>

Quadro 16 - Plano da 11ª aula

<table>
<tr><th colspan="2">AULA 12</th></tr>
<tr><td colspan="2">Objetivos:
➢ Apresentar as instruções referentes a variáveis
➢ Apresentar estruturas de controle
➢ Apresentar laços de repetição
➢ Apresentar o que desvios condicionais</td></tr>
<tr><th>Conteúdo</th><th>Atividades</th></tr>
<tr><td rowspan="2">Estruturas de Controle
Laços de Repetição
Desvios condicionais</td><td>Apresentar as estruturas de controle, laços de repetição e desvios condicionais presentas na Linguagem LilyPnd (ritornelos, casas de primeira e segunda vez, pulos de coda e $)</td></tr>
<tr><td>Exemplo:
Ctrl + shift + v;
\language “english”
\header { title = “XXXXX”
composer = “XXXXXXX”
arranger = “XXXXXXX”}
{\ repet volta 2 { c4 d e f }
\alternative { {c2 e} {f2 g} } }</td></tr>
<tr><td rowspan="2">Variáveis</td><td>Variáveis: Como devem ser declaradas; Onde podem ser chamadas; Como podem ser chamadas;</td></tr>
<tr><td>Ctrl + shift + v;
\language “english”
\header {title = “XXXXX”
composer = “XXXXXXX”
arranger = “XXXXXXX” }
violin = \new Staff {
\relative { a'4 b c b } }
cello = \new Staff { \relative { \clef "bass" e2 d } } {
<<
\violin
\cello
>> }</td></tr>
</table>

Quadro 17 - Plano da 12ª aula

<table>
<tr><th colspan="2">AULA 13</th></tr>
<tr><td colspan="2">Objetivos:
➢ Apresentar mais funcionalidade do Frescobaldi;
➢ Criar pautas com nomes de instrumentos
➢ Apresentar o \paper {...}
➢ Apresentar o \score {...}</td></tr>
<tr><th>Conteúdo</th><th>Atividades</th></tr>
<tr><td>Funcionalidades do Frescobaldi</td><td>Apresentar alguns recursos que todo usuário deveria saber no Frescobaldi e que podem facilitar a utilização para o usuário;</td></tr>
<tr><td rowspan="2">Variáveis / Função / Escopo</td><td>Mostrar o comando para que sejam inseridos nomes de instrumentos nas pautas
\new Staff \with {instrumentName”.....”}
Utilizar a função \paper {...} e realizar exemplos com definições distintas
Utilizar a função \score{...} e demonstrar as diferentes funções internas.</td></tr>
<tr><td>Exemplo:
Ctrl + shift + v;
\language “english”
\paper {
top-margin = 10
botton-margin = 10
#(set-default-paper-size "a3" 'landscape)
}
\header { title = “XXXXX” }
violin = \new Staff {
\relative {
a'4 b c b
} }
\score {
\new Staff \with {instrumentName”.....”}\violin \midi{...}
}</td></tr>
</table>

Quadro 18 -Plano da 13ª aula

<table>
<tr><th colspan="2">AULA 14</th></tr>
<tr><td colspan="2">Objetivos:
➢ Apresentar a inserção de notas de percussão
➢ Revisar todo o conteúdo</td></tr>
<tr><th>Conteúdo</th><th>Atividades</th></tr>
<tr><td rowspan="2">Variáveis / Função / Estruturas de controle e repetição</td><td>Mostrar o modo \drums {...} aos alunos relativo a inserção de notas relacionadas a percussão.</td></tr>
<tr><td>Exemplo 1:
Ctrl + shift + v;
\language “english”
\paper { top-margin = 10
botton-margin = 10
#(set-default-paper-size "a3" 'landscape) }
\drums {
\clef percussion
bd4 bd bd bd
\clef treble
hh4 hh hh hh }
Exemplo 2:
Ctrl + shift + v;
\language “english”
\paper { top-margin = 10
botton-margin = 10
#(set-default-paper-size "a3" 'landscape)}
\drums {
\repeat unfold 2 {
sn16^"L" sn^"R" sn^"L" sn^"L" sn^"R" sn^"L" sn^"R" sn^"R"
\stemUp
sn16_"L" sn_"R" sn_"L" sn_"L" sn_"R" sn_"L" sn_"R" sn_"R"
} }</td></tr>
</table>

Quadro 19 - Plano da 14ª aula

<table>
<tr><th colspan="2">AULA 15</th></tr>
<tr><td colspan="2">Objetivos:
➢ Apresentar a inserção de bibliotecas ou arquivos externos
➢ Encaminhar o trabalho final</td></tr>
<tr><th>Conteúdo</th><th>Atividades</th></tr>
<tr><td rowspan="2">Variáveis / Funções / Bibliotecas e Arquivos Externos</td><td>Apresentar a inserção de arquivos e bibliotecas externas,
Chamar as funções e variáveis presentes nesses arquivos</td></tr>
<tr><td>Exemplo Arquivo:
\version "2.18.2"
\language "english"
trompeteum = \relative c'
{ cs4\mf d8[(ef\< f) g-.] a4|
f16-. g(a e) a4 d->-. f->-. |
\time 3/2
e2\ff^Intense b f |
\clef alto
\numericTimeSignature \time 4/4
<c a>1\fermata \bar "|."| }
trompetedois = \relative c'
{\clef "bass" g2 <af c> 2|
\tuplet 5/4 {a16 g f e f} g2 e4|
fs2 d2 c2 |
\numericTimeSignature <d ef>1 \fermata| }
trompetetres = \relative c'
{ cs4\mf d8[(ef\< f) g-.] a4|
f16-. g(a e) a4 d->-. f->-. |
\time 3/2
e2\ff^Intense b f |
\clef alto
\numericTimeSignature \time 4/4
<c a>1\fermata \bar "|."|
}</td></tr>
</table>

	trompetequatro = \relative c' *{\clef "bass" g2 <af c> 2|* *\tuplet 5/4 {a16 g f e f} g2 e4|* *fs2 d2 c2 |* *\numericTimeSignature <d ef>1 \fermata|* *}* Exemplo *\include*: *\version "2.18.2"* *\language "english"* *\include "definicao.ly"* *\header {* *title = "Vamos colocar o titulo aqui"* *composer = "O Alunos" }* *\score {* << *\new Staff \with {instrumentName = "Tpt 1"} \trompeteum* *% \new Staff \with {instrumentName = "Tpt 2"} \trompetedois* *% \new Staff \with {instrumentName = "Tpt 3"} \trompetetres* *% \new Staff \with {instrumentName = "Tpt 4"} \trompetequatro* >> *\layout {}* *\midi {}* *}*

Quadro 20 - Plano da 15ª aula

5 DESCRIÇÃO E ANÁLISE DE RESULTADOS

Neste capítulo serão apresentados os resultados obtidos com a conclusão do curso, bem como a análise dos dados colhidos no decorrer desta pesquisa.

Após a realização de pesquisa bibliográfica acerca das dificuldades apresentadas pelos alunos iniciantes em programação de computadores e das soluções para atenuar as dificuldades presentes na literatura, foi elaborada uma proposta de ensino que os exponha a conceitos de programação baseada na teoria da aprendizagem significativa de David Ausubel.

A proposta aborda os conteúdos introdutórios de programação de computadores para músicos que estão presentes na maioria das disciplinas de algoritmos, programação e disciplinas equivalentes, como lógica de programação, linguagem de programação, técnicas de programação, entre outras. Dentre os conteúdos abordados pela proposta estão variáveis, estrutura, repetição, declaração, escopo, inclusão de bibliotecas externas, além do uso de uma linguagem de programação para criação de partituras.

A aplicação da proposta ocorreu dentro da disciplina editoração de partituras, com uma turma do Curso Sequencial em Música com habilitação em Regência de Bandas de Departamento de Música da Universidade Federal da Paraíba – UFPB – Campus de João Pessoa. A disciplina é optativa na grade curricular e tem como objetivo o ensino de ferramentas de editoração de partituras. A ferramenta utilizada neste curso foi o Frescobaldi/LilyPond que, como descrito no capítulo anterior é um editor de partituras que tem como entrada, uma linguagem de programação.

As aulas iniciaram com uma turma de onze alunos e terminou com dez alunos, pois houve uma desistência. Uma investigação acerca da desistência apontou que o aluno não conseguiu conciliar o horário da disciplina com demandas particulares.

Para abordar os conteúdos pretendidos na proposta, foi definida uma carga horária de trabalho de 30 (trinta) horas aula, sendo divididas em 15 (quinze) encontros de 2 (duas) horas aula cada.

Abaixo será descrita de forma geral como aconteceram as aulas e os conteúdos aplicados.

5.1 Resultados

A aplicação da proposta em sala de aula ocorreu nos meses de julho, agosto, setembro e outubro de 2018 seguindo o cronograma elaborado e descrito no plano de curso. No decorrer do curso, além das perguntas propostas nos procedimentos metodológicos no subitem de coleta e análise de dados, também foi realizada uma observação constante das aulas, buscando compreender de fato a aplicabilidade da proposta.

Ementa
Histórico das linguagens de programação. Descrição e construção de algoritmos. Metodologia de programação. Introdução à uma linguagem de programação moderna. Variáveis. Estrutura. Repetição. Declaração. Escopo. Inclusão de bibliotecas externas. Vetores.

O primeiro ponto a ser analisado é o cumprimento ou não da ementa da disciplina. Conforme os registros no caderno de anotações de campo e nos planos de aula, observa-se que os elementos dispostos na ementa do curso foram contemplados de forma parcial. O conteúdo referente a vetores não foi aplicado devido ao tempo disponível para o curso. Desta forma, baseado nos procedimentos metodológicos, consideramos que a ementa foi cumprida parcialmente por não contemplar todo o assunto.

Porém, no primeiro contato com os alunos, constatou-se que havia a necessidade de abordar alguns conteúdos que serviriam de base para o curso. Optou-se em incluir um breve histórico sobre a computação, desde o surgimento até os dias atuais e uma abordagem sobre *Software* e *Hardware*. Sendo assim, com a inserção de novos tópicos, considera-se que o cumprimento da ementa foi parcial com a inserção de tópicos.

Cumprimento da ementa e objetivos da disciplina
Parcial com inserção de novos conteúdos.

Quadro 21 - Execução da ementa

A segunda verificação tem como base os conteúdos abordados em exercícios de fixação e trabalhos referentes da disciplina. Durante as aulas, foram encaminhados alguns exercícios de fixação de conteúdo bem como uma trabalho final. Os conteúdos abordados dentro dos exercícios foram avaliados conforme a inclusão do conteúdo estudado, ou seja, se foram incluídos nos exercícios e trabalhos ou não. O quadro abaixo mostra o que os alunos utilizaram na realização dos exercícios.

	SINTAXE / INTRUÇÕES DE VARIÁVEIS	**FUNÇÃO / TIPOS DE SAÍDA**	**ESTRUTURA DE CONTROLE**	**INCLUSÃO DE BIBLIOTECAS EXTERNAS**
Exercício I	*Incluídos Plenamente*			
Exercício II	*Incluídos Plenamente*	*Incluídos Plenamente*		
Exercício III	*Incluídos Parcialmente*	*Incluídos Parcialmente*	*Incluídos Parcialmente*	
Trabalho Final	*Incluídos Parcialmente*	*Incluídos Parcialmente*	*Incluídos Parcialmente*	*Não Incluídos*

Quadro 22 - Conteúdos utilizados nas atividades

Observando o quadro acima, podemos constatar que a turma incluiu em seus exercícios e trabalhos, de modo geral, os conteúdos abordados em sala de aula. O trabalho final teve por objetivo a elaboração de uma partitura musical em formato de grade[5] com a inclusão de todos os conteúdos abordados em sala de aula e o conteúdo que trata da inserção de arquivos externos em seu código fonte não foi contemplado.

A elaboração de uma partitura em formato de grade com a inclusão de vários instrumentos pode ser obtida de formas distintas e entre elas, a inclusão dos instrumentos

[5] Grade – A grade musical é uma partitura que contém todas as informações que de uma música, um compilado de todos os instrumentos e serve de guia para o maestro.

criados separadamente em forma de arquivos. No código fonte que gera a grade também é possível criar os instrumentos, dando assim a opção do editor musical de escolher de qual maneira vai organizar seu código.

Na semana que antecedeu o trabalho final, muitas dúvidas relacionadas a inserção de arquivos externos surgiram em um grupo de aplicativo de celular criado pelos alunos para informes e eventuais dúvidas. O professor orientador da disciplina e o pesquisador esclareceram dúvidas que surgiram, mas na análise do trabalho final observou-se que os alunos não incluíram o conteúdo relacionado a inserção de arquivos externos (*include)*. Sendo assim, constatou-se que a não utilização desse conteúdo deve-se a dificuldade dos alunos em compreender esse assunto. Portanto foi considerado que os conteúdos abordados em sala de aula, foram incluídos parcialmente nas atividades.

O terceiro dado coletado e analisado definido nos procedimentos metodológicos foi avaliação do desempenho dos alunos quanto a evolução frente aos conteúdos e pelo índice de aprovação na disciplina.

A avaliação de desempenho foi realizada por blocos e tem por objetivo verificar de maneira precisa a evolução dos alunos frente ao conteúdo. Esta avaliação teve como base as atividades que os alunos executaram durante a disciplina, sendo apenas utilizada para efeitos da pesquisa. A tabela abaixo mostra a divisão e o marco de avaliação de desempenho.

História da computação; Software x Hardware; Algoritmos	1º Avaliação de desempenho (prova)
Introdução a Linguagem LilyPond; Casos sensíveis, Sintaxe, Variáveis.	2ª Avaliação de desempenho (exercício)
Função; Tipos de Saída	3ª Avaliação de desempenho (exercício)
Estrutura de Controle; Laços de repetição; Desvios	4ª Avaliação de desempenho (exercício)
Inclusão de arquivos e bibliotecas externas	5ª Avaliação de desempenho (trabalho)

Quadro 23 - Marcos de avaliação de desempenho

Para categorizar os dados obtidos, foram adotados os conceitos A, B, C e D onde: o conceito A significa “Aprendizagem Plena” com nota 10; B - “Aprendizagem Parcialmente Plena” com notas entre 7 e 9,9; C - “Aprendizagem Suficiente” com notas entre 5 e 6,9; D - “Aprendizagem insuficiente” com notas abaixo de 5. A tabela abaixo apresenta os dados obtidos na primeira avaliação.

Desempenho dos alunos na primeira avaliação	
Conceitos	**Número de Alunos**
Conceito A (aprendizagem plena)	0
Conceito B (aprendizagem parcialmente plena)	0
Conceito C (aprendizagem suficiente)	4
Conceito D (aprendizagem insuficiente)	3
Não avaliados (ausentes)	3

Quadro 24 – Desempenho no primeiro marco avaliativo

A avaliação inicial teve como base a prova com conteúdo de história, algoritmos e conceitos sobre *Software* e *Hardware*. Observando o desempenho dos alunos na primeira verificação, constatamos dados pouco satisfatórios com notas abaixo de 7, mesmo com uma revisão de conteúdo com o objetivo de esclarecer possíveis dúvidas. Nesta análise, os alunos que não realizaram a prova foram desconsiderados, pois não é possível mensurar o desempenho dos alunos quanto a compreensão do conteúdo.

Na próximas tabelas serão expostos os resultados da segunda, terceira, quarta e quinta avaliações que tiveram como base os exercícios de fixação e trabalho final. As atividades das tabelas abaixo foram realizadas por todos os alunos.

Desempenho dos alunos na segunda avaliação	
Conceitos	**Número de Alunos**
Conceito A (aprendizagem plena)	2

Conceito B (aprendizagem parcialmente plena)	5
Conceito C (aprendizagem suficiente)	2
Conceito D (aprendizagem insuficiente)	1

Quadro 25 - Desempenho no segundo marco avaliativo

Desempenho dos alunos na terceira avaliação	
Conceitos	**Número de Alunos**
Conceito A (aprendizagem plena)	4
Conceito B (aprendizagem parcialmente plena)	4
Conceito C (aprendizagem suficiente)	2
Conceito D (aprendizagem insuficiente)	0

Quadro 26 - Desempenho no terceiro marco avaliativo

Desempenho dos alunos na quarta avaliação	
Conceitos	**Número de Alunos**
Conceito A (aprendizagem plena)	3
Conceito B (aprendizagem parcialmente plena)	6
Conceito C (aprendizagem suficiente)	1
Conceito D (aprendizagem insuficiente)	0

Quadro 27 - Desempenho no quarto marco avaliativo

Desempenho dos alunos na quinta avaliação	
Conceitos	**Número de Alunos**

Conceito A (aprendizagem plena)	0
Conceito B (aprendizagem parcialmente plena)	9
Conceito C (aprendizagem suficiente)	1
Conceito D (aprendizagem insuficiente)	0

Quadro 28 - Desempenho no quinto marco avaliativo

Além dos dados obtidos pelas cinco avaliações realizadas com base no material produzido pela disciplina, também foi observado o índice de aprovação da turma. As notas que compõem a média final foram obtidas pelo somatório de três notas e consistem em um prova, um trabalho e a participação dos alunos em sala de aulas.

Para estabelecer o índice de aprovação na disciplina foram consideradas todos os alunos que participaram efetivamente da disciplina, totalizando assim dez alunos. Conforme descrito no início deste capítulo, um aluno desistiu do curso que não foi contabilizado para avaliação do desempenho, e os demais conseguiram êxito na disciplina, obtendo assim, a aprovação.

Além dos resultados apresentados até o momento, cabe mencionar observações qualitativas do pesquisador sobre o engajamento dos alunos e sua participação durante o curso e percebeu-se indícios de que as aulas com uma abordagem significativa dos conteúdos introdutórios de programação de computadores podem produzir um interesse maior por parte dos alunos.

Alguns alunos ao se expor um determinado conceito como por exemplo *Se Então*, mencionavam que este conceito poderia ser comparado aos *ritornelos* existentes na música, comparando os conceitos. De forma semelhante aconteceu quando apresentado o conceito de variável. O aluno indagou que tal tipo de situação poderia também acontecer na música quando se trata de um ostinato (frase ou ritmo que se repete consecutivamente na música).

Em alguns casos foi possível acelerar o conteúdo, pois a relação estabelecida com a música possibilitou a compreensão mais rápida de alguns conceitos, como por exemplo, o

conteúdo sobre escopo, onde imediatamente os alunos fizeram analogia ao mudança de compasso ou tonalidade em determinado trecho de uma música.

Desta forma, do ponto de vista investigativo, conclui-se que esta aplicação obteve êxito apesar dos diversos fatores que foram identificados ao longo de sua execução. Dessa forma, podemos considerar que a execução pratica da proposta obteve diversos fatores positivos, conforme foi mencionado acima e colaboram na confirmação da hipótese sugerida por este estudo.

5.2 Análise e discussão dos resultados

Com base na revisão de literatura e nos resultados coletados na aplicação prática desta proposta, apresenta-se a seguir a análise e discussão dos dados desta pesquisa.

Outra hipótese que orientou este trabalho, é a de que a elaboração e desenvolvimento de uma proposta de ensino baseada na Aprendizagem Significativa iria possibilitar a aprendizagem de conceitos de programação para músicos. Sendo assim, facilitar o uso prático do computador, compreender o que é uma linguagem de programação, conhecer o processo de criação um *software* e despertar o interesse do músico para o estudo da programação. Desta forma ajudando-os a elaborar soluções para os seus problemas como a criação de uma *webpage* para divulgar seu trabalho, trabalhos colaborativos envolvendo hardware e software, criação de aplicativo que auxilie o professor em sala de aula, entre outros, foram algumas das contribuições esperadas.

Desta forma, o objetivo central foi verificar a possibilidade de ensino introdutório de programação de computadores baseada na Aprendizagem Significativa. Pretendeu-se apresentar aos alunos conceitos do mundo da programação que pudessem ser relacionados com a música, possibilitando um compreensão sólida do conteúdo. Para promover esta experiência, utilizou-se o software *Frescobaldi* como ferramenta de apoio para o ensino.

Preliminarmente, estabelecida a relação entre conceitos de programação e conceitos de música, podemos denotar que a possibilidade de ensinar programação para músicos sob uma abordagem significativa é real. Visando a validação da pesquisa buscou-se acompanhar e averiguar a aplicabilidade da proposta para o ensino dos conceitos de programação de computadores. Para isso elencou-se os seguintes critérios:

I. Cumprimento da ementa e objetivos da disciplina,

II. Verificação dos conteúdos abordados em exercícios de fixação e trabalhos referentes da disciplina,

III. Observar do desempenho dos alunos.

Para enriquecer os resultados foram adicionadas observações qualitativas do pesquisador a respeito do engajamento dos alunos e sua participação e comentários do professor titular da disciplina a respeito do desempenho dos estudantes.

Cabe pontuar que geralmente o escopo das disciplinas introdutórias em programação, no seu princípio gira em torno da definição de conceitos, entretanto, a não compreensão destes compromete o desempenho do aluno nos conteúdos seguintes. Desta forma, buscou-se fornecer aos alunos princípios que tivessem como base uma analogia na música, instigando o aluno a formular seus próprios conceitos.

Durante a aplicação da proposta, ao se inserir um novo tópico a ser estudado, foram trazidos para uma discussão prévia, conceitos musicais que possuem significado semelhante ao novo conteúdo. Podemos citar como exemplo a introdução a linguagem de programação. Antes de estudar o conteúdo foi trazido para a discussão a leitura de uma partitura musical. Não é encontrado na literatura uma linguagem igual e isso requer um conhecimento apropriado sobre a linguagem musical. Invariavelmente, a leitura e interpretação de uma partitura musical traz uma série de instruções que devem ser seguidas durante a execução de uma partitura. Sendo assim, todos os alunos compreenderam o papel da linguagem musical e as instruções nela contida.

Com conceito previamente definido, iniciou-se o estudo sobre a linguagem de programação e prontamente os alunos compreenderam o conteúdo, o que influenciou positivamente a capacidade dos estudantes assimilarem os conteúdos envolvidos em programação e também desenvolverem seus próprios códigos.

De acordo com a aprendizagem significativa, um dos pontos fundamentais consiste em determinar aquilo que o aprendiz já sabe ou conhece (MASINI; MOREIRA, 2001), ou seja, o estado atual da sua estrutura cognitiva (conjunto de ideias que, no aluno, preexistem à nova aprendizagem), para que a proposta de ensino seja baseada nestes conhecimentos.

No entanto, a programação de computadores é um assunto completamente novo para a maioria dos estudantes iniciantes que não possuem o embasamento necessário para que a aprendizagem ocorra de maneira significativa. Neste sentido, cabe ao professor explorar e elaborar uma metodologia, fazendo com que a aprendizagem de programação de computadores ocorra com maior sentido.

Quando o estudante não possui o conjunto de conhecimentos prévios necessários sobre o novo conceito a ser aprendido, Ausubel sugere como estratégia de manipular a estrutura cognitiva, com organizadores prévios (MOREIRA, 1999). Organizadores prévios, tem a função de ponte entre o que o aprendiz sabe e o que deve saber, visam estabelecer relações entre ideias, proposições e conceitos já existentes na estrutura cognitiva, a fim de que a aprendizagem possa ser significativa.

Como exemplo, a ideia de "repetição" já é conhecida dos músicos como ritornelo, e apresentar no computador uma instrução de repetição (*repeat*), a execução pela máquina adquire um caráter quase concreto. Desta forma, se um músico fosse apresentado a este tipo de conteúdo de maneira tradicional, poderia não compreender a instrução de forma correta, além de se frustrar por erros de sintaxe inesperados e se questionar se afinal entendeu ou não a matéria.

Na proposta implementada onde os conteúdos a serem aprendidos pelos estudantes se deram por meio da apresentação de pequenas estruturas de códigos simples, que ao longo do processo vão se tornando maiores e mais complexas. Por meio de atividades práticas, os estudantes puderam desenvolver seu próprio repertório de conhecimentos e habilidades para resolver problemas e representa-los no ambiente computacional.

Sendo assim, os conteúdos na abordagem proposta foram apresentados de maneira cíclica, onde a cada aula um novo assunto era introduzido sem deixar de lado o anterior, possibilitando ao estudante, o ganho de maturidade. Se novos conceitos fossem apresentados isoladamente como caixas, ou seja, separadamente e sem conexão, tal metodologia faria com que o aluno sentisse dificuldades para combinar esses conceitos na aplicação, quando necessário.

Alguns estudantes, demonstraram dificuldades na realização das atividades propostas durante a aplicação e se não for detectado e corrigido, este processo pode distanciar ainda

mais o aluno dos objetivos destas disciplinas. É crucial que o professor não forneça as respostas, mas desenvolva situações que visem oportunizar ao aluno maneiras de romper estes bloqueios.

Ainda com relação ao desempenho, foi possível perceber que alguns alunos apresentam um ganho de aprendizagem diferenciando a estes conteúdos significativos e avançam sem problemas durante as aulas. Outros, mesmo se esforçando não conseguem chegar a tal ganho, que corrobora com aquela tabela sobre nível de aprendizado.

A aprendizagem destes conteúdos não ocorre em tempo real, por isso, há necessidade de que um mesmo conteúdo seja revisto e aplicado por diversas vezes em momentos e situações diferentes. A abordagem deve ser feita na forma cíclica crescente, para que no correr do processo de aprendizagem o estudante adquira maturidade e não desenvolva repulsa ao conteúdo. Sendo assim, a motivação é outro fator essencial para que os alunos aprendam, e por isso conseguir o engajamento dos alunos e que estes estejam realmente envolvidos com o processo pode ser outro fator de sucesso para retenção destes conhecimentos. Neste sentido o investimento do professor no ambiente de sala aula e a utilização de uma metodologia que valorize o conhecimento prévio pode colaborar para o envolvimento dos alunos com estes conteúdos.

Observou-se indícios de que as aulas práticas dos conteúdos introdutórios de programação de computadores podem produzir um engajamento maior por parte dos alunos e, além disso, podem diminuir a lacuna no quesito abstração, um dos grandes problemas apontados por pesquisas sobre o ensino/aprendizagem de programação de computadores.

Por fim, cabe destacar alguns outros aspectos positivos referentes aos resultados presentes na aplicação prática da proposta. A definição de conceitos, geralmente presentes no escopo da maioria das disciplinas introdutórias de programação é feita no início da disciplina. Enquanto isso, na proposta, foram tratados no decorrer da disciplina e não influenciou a capacidade dos alunos na resolução de problemas e sua codificação. O único conceito que foi tratado separadamente e no início da disciplina foi a definição de algoritmo.

Percebeu-se indícios de que, como por exemplo na apresentação do comando de repetição *\repeat {...}*, quando o aluno pode ver e analisar a resposta do computador mediante as suas entradas de comandos, pode contribuir para diminuir o grau de abstração presente na

matéria de programação, favorecendo entre outros aspectos a motivação dos estudantes (PELIZZARI; KRIEGL, 2002).

> Durante o ensino/aprendizagem dos fundamentos de programação de computadores, percebe-se que os estudantes concentram seus esforços nas características específicas de sintaxe e semântica da linguagem, e não em como realmente acontece a programação (a resolução do problema e sua formalização numa linguagem de programação), sendo assim, a utilização de linguagens de programação com sintaxe mais simples e clara, pode favorecer o processo (BERSSANETTE, 2016, p. 99).

Alguns alunos podem ter dificuldades em identificar erros nos resultados produzidos pelos seus códigos inicialmente, estas dificuldades estão diretamente relacionadas a pouca experiência em programação, deste modo, a situação indicada pela proposta, possibilita aos alunos verem os conteúdos mais vezes, o que pode contribuir para a aquisição de experiência em programação (AURELIANO; TEDESCO, 2012; BYRNE; LYONS, 2001; HAGAN; MARKHAM, 2000).

De posse da análise e observando o desempenho da turma, podemos concluir que os resultados da aplicação colaboram na confirmação da hipótese sugerida por este estudo.

5.3 Considerações Finais

O processo de ensino/aprendizagem dos fundamentos de programação de computadores, tem se mostrado difícil para estudantes e professores. Isto é exposto pela literatura que referencia os elevados níveis de insucesso, as elevadas taxas de desistência e até mesmo abandono do curso.

As dificuldades envolvidas no processo de ensino/aprendizagem de computadores abrangem os quatro integrantes envolvidos no processo, sendo: a instituição, o professor, o aluno e os conteúdos, com diversas variáveis para cada um destes.

A literatura disponibiliza propostas que visam contribuir com o processo de aprendizagem de programação ou atenuar as dificuldades existentes no processo. Estas propostas geralmente se encaixam em três principais vertentes: ferramentas, estratégias e a combinação de ferramentas e estratégias.

Deste modo, esta pesquisa teve como objetivo a elaboração de uma proposta com vistas a verificar a possibilidade de ensinar conceitos introdutórios de programação de

computadores para músicos, baseada na teoria da aprendizagem significativa, enfatizando/valorizando a interação com a máquina.

Constatou-se então que a proposta abrange tópicos presentes na ementa da maioria das disciplinas de introdução a programação, mesmo tendo variadas linguagens de programação. Ementas que contenham conteúdos relacionados a sintaxe, variáveis, estruturas de controle, laços de repetição e bibliotecas estão presentes no escopo das disciplinas de introdução a programação.

Outro ponto importante a ser observado é a oportunidade dos estudantes verem os conteúdos mais vezes de maneira inter-relacionados e de maneira cíclica onde pode-se ganhar maturidade para assimilar estes conteúdos. Isso é possível devido ao universo onde essa proposta foi aplicada. A criação de partituras musicais na forma de grade com o auxílio do *Frescobaldi* exige dos estudantes o uso constante e cíclico de todos os conteúdos presentes no curso.

Considerado o aspecto mais importante da pesquisa, a utilização de uma metodologia que considere conhecimento prévio, concluímos que é possível proporcionar um engajamento maior por parte dos alunos podendo diminuir a lacuna no quesito abstração, um dos grandes problemas apontados por pesquisas sobre o ensino /aprendizagem de programação de computadores.

Em relação a flexibilidade na definição de termos e a sequência com que os conteúdos foram abordados, não foram identificados quedas no desempenho dos alunos no decorrer do curso. Neste sentido, recorremos a Ausubel, o qual indica que cada disciplina tem uma estrutura articulada e hierarquicamente organizada de conceitos. No entanto, a ordem em que os principais conceitos e ideias da matéria de ensino são apresentadas muitas vezes não é a mais adequada para facilitar a interação com o conhecimento prévio do aluno.

Deste modo, os resultados das aplicações da proposta, apresentam indícios que a estrutura articulada e hierarquicamente organizada de conceitos no ensino de programação pode não ser o mais adequado. Certa flexibilidade permite que os conteúdos sejam apresentados de forma inter-relacionada onde o estudante pode ganhar maturidade para assimilar.

Em se tratando da relação entre música e tecnologia, não podemos deixar de ressaltar a importância do ensino de programação para músicos. A literatura aponta que qualquer indivíduo relacionado à música é invariavelmente requisitado a um envolvimento com a tecnologia, travando contato com um linguajar que se estende desde o manuseio de um aparelho de som até ao funcionamento de complexos processadores digitais de áudio, ferramentas de edição de partituras, educação musical, entre outros. Desta maneira, diante da possibilidade confirmada do ensino de conceitos de programação de computadores para músicos, é possível despertar o seu interesse em aprofundar os conhecimentos de programação

Por fim, a impossibilidade de realizar o estudo com formação de um grupo de controle e aplicação da proposta por outro professor foi uma das limitações encontradas na realização da pesquisa. Entretanto, a hipótese foi confirmada, visto que a aplicação indica resultados positivos.

5.3.1 Recomendações para Estudos Futuros

Este trabalho, exploratório e descritivo em sua natureza, buscou não só tentar levantar possibilidades referentes ao ensino introdutório de programação para músicos, como também levantar questões adicionais a serem estudadas posteriormente. Dado este caráter, a falta de estudos sobre o tema e o tempo para a realização da pesquisa, o trabalho também deixa considerações sobre trabalhos posteriores.

Como trabalhos futuros sugere-se a replicação deste estudo utilizando a proposta elaborada em outras instituições, com o objetivo de verificar os resultados posteriores da aplicação da proposta. Replicar a proposta no sentido inverso, buscando ensinar conceitos musicais através da programação.

Agregar a esta pesquisa aspectos motivacionais e/ou cognitivos, analisar detalhadamente como certas habilidades e competências influenciam a aprendizagem de programação;

Sugere-se também a aplicação desta proposta em níveis distintos de escolaridade, pois a música está presente em todas as idades e formações. A adequação da proposta juntamente

com a utilização de outras ferramentas, a exemplo do *Pure Data (Pd)* pode gerar novas contribuições.

REFERÊNCIAS

ALBUQUERQUE, D.; BREMGARTNER, V.; LIMA, H.; SALGADO, N.; Uma Experiência do Uso Do Hardware Livre Arduino no Ensino De Programação De Computadores - In V Congresso Brasileiro de Informática na Educação (CBIE) 5, 2016, Uberlândia, MG. **Anais..., 2016.** Disponível em: <http://www.br-ie.org/pub/index.php/wie/article/view/6602 >. Acesso em: 16 mai. 2017.

ALMEIDA, E. S. et al. AMBAP: Um ambiente de apoio ao aprendizado de Programação**.** In: CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 22, 2002, Florianópolis. **Anais... Florianópolis: SBC, 2002.** Disponível em: <http://200.17.137.110:8080/licomp/Members/jeanemelo/plonelocalfolderng.2006-04-10.7475913377/PEP/Aulas23/2002SbcAmbap.pdf>. Acesso em: 24 Mai. 2018.

AMARAL, L. R. DO; SILVA, G. B. E; PANTALEÃO, E. Plataforma Robocode como Ferramenta Lúdica de Ensino de Programação de Computadores - Extensão Universitária em Escolas Públicas de Minas Gerais - In Anais do XXVI Simpósio Brasileiro de Informática na Educação (SBIE) 26, 2015, Maceió, AL. **Anais..., 2015**. Disponível em: http://www.br-ie.org/pub/index.php/sbie/article/view/5152 >. Acesso em: 3 ago. 2017.

AMBRÓSIO, A. P. L. et al. Programação de Computadores: compreender as dificuldades de aprendizagem dos alunos. **Revista Galego-Portuguesa de Psicoloxía e Educación**, v. 19, n. 1, ano 16, p. 185–197, 2011. Disponível em: <http://repositorium.sdum.uminho.pt/handle/1822/15554>. Acesso em: 14 ago. 2017.

AURELIANO, V. C. O. AND TEDESCO, P. C. A. R. Ensino e Aprendizagem de Programação para Iniciantes: uma Revisão Sistemática da Literatura focada no SBIE e WIE. In Anais do XXIII Simpósio Brasileiro de Informática na Educação (SBIE) 23, 2012, Rio de Janeiro, RJ. **Anais..., 2012**. Disponível em: < http://www.br-ie.org/pub/index.php/sbie/article/view/1718 >. Acesso em: 3 ago. 2017.

AUSUBEL, D.P. **Aquisição e retenção de conhecimentos: uma perspectiva cognitiva**. Lisboa: Plátano, 2003. Disponível em: <http://files.mestrado-em-ensino-deciencias.webnode.com/200000007-610f46208a/ausebel.pdf>. Acesso em: 14 Jun. 2018.

AUSUBEL, D.P. **The psychology of meaningful verbal learning**. New York: Grune and Stratton, 1963. Disponível em: <http://psycnet.apa.org/psycinfo/1964-10399-000>. Acesso em: 10 jun. 2018.

AUSUBEL, D.P; NOVAK, JD; HANESIAN, H. **Educational psychology: a cognitive view**. New York:, Holt, Rinehart and Winston. 1968. Disponível em: <http://www.spbkbd.com/english/art_english/art_51_030211.pdf>. Acesso em: 8 jun. 2018.

AUSUBEL, D.P; NOVAK, JD; HANESIAN, H. **Psicologia educacional**. Rio de Janeiro, Interamericana, 1980. Disponível em:

<https://scholar.google.com.br/scholar?q=Psicologia+educacional&btnG=&hl=pt-BR&as_sdt=0%2C5#3>. Acesso em: 29 jun. 2015.

BARANAUSKAS, M. C. C. **Procedimento, função, objeto ou lógica? Linguagens de Programação vistas pelos seus paradigmas**. In: José Armando Valente. (Org.). Computadores e Conhecimento - Repensando a Educação. Campinas, SP: Unicamp, 1994, p. 45-63. Disponível em: <http://www.nied.unicamp.br/publicacoes/arquivos/3XaQ1o8Nmk\nhttp://disciplinas.dcc.ufba.br/pub/MATA56/Exercicios/(Leitura_e_Resenha)_ArtigoDiscussaoParadigmas.pdf>. Acesso em: 9 jun. 2018.

BATISTA, E. J. S.; CASTRO JR., A. A.; BOGARIM, C. A. C.; LARREA, A. A. Utilizando o Scratch como ferramenta de apoio para desenvolver o raciocínio lógico das crianças do ensino básico de uma forma multidisciplinar - In Anais do XXI Workshop de Informática na Escola (WIE) 21, 2015, Maceió, AL. **Anais..., 2015**. Disponível em: < http://www.br-ie.org/pub/index.php/wie/article/view/5049>. Acesso em: 3 ago. 2017.

BATISTA, E. J. S.; CASTRO JR., A. A.; CANTERO, S. V.; BOGARIM, S. A. C.; LARREA, A. A. Uso do Scratch no ensino de programação em Ponta Porã: das séries inicias ao ensino superior - In Anais do XXII Workshop de Informática na Escola (WIE) 22, 2016, Uberlândia, MG. **Anais..., 2016.** Disponível em: http://www.br-ie.org/pub/index.php/wie/article/view/6863 >. Acesso em: 04 mai. 2017.

BELCHIOR, H.; BONIFÁCIO, B.; FERREIRA, R. Avaliando o Uso da Ferramenta Scratch para Ensino de Programação através de Análise Quantitativa e Qualitativa -In Anais do XXVI Simpósio Brasileiro de Informática na Educação (SBIE) 26, 2015, Maceió, AL. **Anais..., 2015**. Disponível em: < http://www.br-ie.org/pub/index.php/sbie/article/view/5396 >. Acesso em: 3 ago. 2017.

BENNEDSSEN, J.; CASPERSEN, M. E. **Abstraction ability as an indicator of success for learning computing science?** In: Proceedings of the ACM Workshop on International Computing Education Research, ICER 08. Sydney, Australia, 2008. Disponível em: <http://dl.acm.org/citation.cfm?id=1404523>. Acesso em: 8 jun. 2018.

BERSSANETTE, J.H. **Ensino de programação de computadores: uma proposta de abordagem prática baseada em Ausubel** - Ponta Grossa, 2016. Disponível em: <http://repositorio.utfpr.edu.br/jspui/bitstream/1/2487/1/PG_PPGECT_M_Berssanette%2C%20Jo%C3%A3o%20Henrique_2016.pdf>. Acesso em: 5 out. 2017.

BIGGS, J. **Aligning teaching and assessing to course objectives**. In: Teaching and Learning in Higher Education: New Trends and Innovations. University of Aveiro, 13-17 April, 2003. Disponível em: <https://www.dkit.ie/zhhans/system/files/Aligning_Reaching_and_Assessing_to_Course_Objectives_John_Biggs.pdf>. Acesso em: 5 jun. 2018.

BORGES, M. A. F. Avaliação de uma metodologia alternativa para a aprendizagem de programação. In: WORKSHOP DE EDUCAÇÃO EM COMPUTAÇÃO – WEI 2000, 8., 2000, Curitiba. **Anais... Curitiba: [s.n.], 2000.** Disponível em: <http://www.niee.ufrgs.br/eventos/SBC/2000/pdf/wei/relatos/selecionados/wei006.pdf> Acesso em: 15 jul. 2018.

BRANDÃO, Carlos Rodrigues. **Pesquisa participante**. São Paulo: Brasiliense, 1999.

BRASIL. Ministério da Educação. Secretaria de Educação Superior. **Diretrizes Curriculares de Cursos da área de Computação e Informática.** Brasília: MEC/SESU, 2001.

BRUSILOVSKY, PETER ET AL. **Teaching programming to novices: a review of approaches and tools**. In: Proceedings of ED-MEDIA'94 - World conference on educational multimedia and hypermedia. Vancouver, Canada, p. 103-110, 1994. Disponível em: <http://www.computer.org/portal/web/csdl/doi/10.1109/FIE.1999.839268>. Acesso em: 4 jun. 2018.

CAMBRUZZI, E.; SOUZA, R. M. DE. Robótica Educativa na aprendizagem de Lógica de Programação: Aplicação e análise - In Anais do XXI Workshop de Informática na Escola (WIE) 21, 2015, Maceió, AL. **Anais..., 2015**. Disponível em: < http://www.br-ie.org/pub/index.php/wie/article/view/4981 >. Acesso em: 3 ago. 2017.

CARDOSO, R.; ANTONELLO, S. L. Interdisciplinaridade, programação visual e robótica educacional: relato de experiência sobre o ensino inicial de programação - In Anais dos Workshops do IV Congresso Brasileiro de Informática na Educação (CBIE) 4, 2015, Maceió, AL. **Anais..., 2015**. Disponível em: < http://www.br-ie.org/pub/index.php/wcbie/article/view/6282 >. Acesso em: 7 ago. 2017.

CARVALHO, L. S. G.; OLIVEIRA, D. B. F.; GADELHA, B. F. Juiz online como ferramenta de apoio a uma metodologia de ensino híbrido em programação - In Anais do XXVII Simpósio Brasileiro de Informática na Educação (SBIE) 27, 2016, Uberlândia, MG. **Anais..., 2016.** Disponível em: < http://www.br-ie.org/pub/index.php/sbie/article/view/6694 >. Acesso em: 04 mai. 2017.

CASPERSEN, M. E.; BENNEDSEN, J. Instructional design of a programming course: a learning theoretic approach. In: Proceedings of the 3rd International Workshop on Computing Education Research - **ICER. Atlanta, GA: ACM, 2007**. Disponível em: <http://dl.acm.org/citation.cfm?id=1288595>. Acesso em: 28 jul. 2018.

CORREIA, M. C. **A Observação Participante enquanto técnica de investigação**. Pensar Enfermagem, 13(2), 30-36. 1999

CSTA - **Computer Science Teacher Association**. (2011) "CSTA K-12 Computer Science Standards". CSTA Standards Task Force. ACM - Association for Computing Machinery

DEREMER, D. Improving the learning environment in CS I: experiences with communication strategies. **ACM SIGCSE Bulletin, v.25, n.3, 1993.** Disponível em: <http://dl.acm.org/citation.cfm?id=165418>. Acesso em: 18 jun. 2018.

Deshaies, B. **Metodologia da investigação em ciências humanas**. Lisboa: Instituto Piaget. 1997.

DETERDING, S.; DIXON, D.; KHALED, R.; NACKE, L. E. Gamification: Toward a Definition. Conference on Human Factors in Computing Systems. **Anais.p.12–15, 2011. Vancouver: ACM Press.** Disponível em: <http://gamification-research.org/wpcontent/uploads/2011/04/02-Deterding-Khaled-Nacke-Dixon.pdf>. Acesso em: 28 maio. 2017.

DUNICAN, E. Making the analogy : alternative delivery techniques for first year programming courses. In: Proceedings from the 14º Workshop of the Psychology of Programming Interest Group. **Carlow: Brunel University, 2002.** Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.115.8440&rep=rep1&type=pdf>. Acesso em: 25 jul. 2018.

EDUCAÇÃO, M. DA. **Diretrizes Curriculares Nacionais para os cursos de graduação na área da Computação**. v. 2016, n. 1977, p. 1–9, 2012.

FALKEMBACH, G. A. M.; AMORETTI, M. S. M.; et al. Aprendizagem de algoritmos: uso da estratégia ascendente de resolução de problemas. In: TALLER INTERNACIONAL DE SOFTWARE EDUCATIVO, 8., 2003. Santiago. **Anais... Santiago, Chile, 2003**. Disponível em: <http://www.tise.cl/2010/archivos/tise2003/papers/aprendizagem_de_algoritmos.pdf> Acesso em: 6 jul. 2018.

FELDER, Richard M.; BRENT, Rebecca. Designing and teaching courses to satisfy the ABET Engineering Criteria. **Journal of Engineering Education**, v. 92, n. 1, p. 7- 25, 2003. Disponível em: <http://onlinelibrary.wiley.com/doi/10.1002/j.2168-9830.2003.tb00734.x/abstract>. Acesso em: 6 jun. 2018.

FILIPPO, D. D. R.. Suporte à Coordenação em Sistemas Colaborativos: uma pesquisa-ação com aprendizes e mediadores atuando em fóruns de discussão de um curso a distância. 2008. 281f. *Tese de Doutorado.* Pontifícia Universidade Católica do Rio de Janeiro, PUC RJ. Disponível em: < https://www.maxwell.vrac.puc-rio.br/11743/11743_3.PDF >. Acesso em: 22 set. 2017.

GALDINO, C. B. T.; NETO, S. R. S.; COSTA, E. B. KidCoder: Uma Proposta de Ensino de Programação de forma Lúdica - In Anais do XXVI Simpósio Brasileiro de Informática na Educação (SBIE),26, 2015, Maceió, AL. **Anais..., 2015**. Disponível em: < http://www.br-ie.org/pub/index.php/sbie/article/view/5340 >. Acesso em: 7 ago. 2017.

GOHN, Daniel M. **Educação musical a distância: propostas para o ensino e aprendizagem de percussão.** 2009. 191f. *Tese de Doutorado*. Escola de Comunicações e Artes, Universidade de São Paulo. São Paulo, 2009.

GOMES, A.; HENRIQUES, J.; MENDES, A. J. **Uma proposta para ajudar alunos com dificuldades na aprendizagem inicial de programação de computadores. Educação, Formação & Tecnologias**, v. 1, n. 1, p. 93–103, Maio, 2008. Disponível em: <http://www.eft.educom.pt/index.php/eft/article/view/23>. Acesso em: 14 jul. 2018.

GOMES, A.; MENDES, A. J. Learning to program-difficulties and solutions. In: International Conference on Engineering Education - **ICEE, 2007. Coimbra, Portugal: [s.n.].** Disponível em: <http://ineer.org/Events/ICEE2007/papers/411.pdf>. Acesso em: 7 jul. 2018.

GOMES, R. C. S.; MELO, J. C. B. DE; TEDESCO, P. C. A. R. Jogos Digitais no Ensino de Conceitos de Programação para Crianças - In Anais do XXVII Simpósio Brasileiro de Informática na Educação (SBIE) 27, 2016, Uberlândia, MG. **Anais..., 2016.** Disponível em: < http://br-ie.org/pub/index.php/sbie/article/view/6728 >. Acesso em: 18 mai. 2017.

GREENING, T. **Emerging Constructivist Forces in Computer Science Education: Shaping a New Future?** Computer Science Education in the 21st Century, p. 47–80, 2000.

GRIES, D. **What should we teach in an introductory programming course?.** ACM SIGCSE Bulletin, v. 6, n. 1, p. 81–89, 1974.

HABERMAN, B.; MULLER, O. **Teaching abstraction to novices: Pattern-based and ADT-based problem-solving processes.** In: 38th ASEE/IEEE Frontiersin Education <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4720415>. Acesso em: 8 jul. 2018.

HENDERSON, P. B. **Modern introductory computer science**. ACM SIGCSE Bulletin, v. 19, n. 1, p. 183–190, 1987. Disponível em: <http://dl.acm.org/citation.cfm?id=31756>. Acesso em: 6 ago. 2018.

HINTERHOLZ JUNIOR, O. Tepequém: uma nova ferramenta para o ensino de algoritmos nos cursos superiores em Computação. In: WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO (WEI), 7., 2009, , Bento Gonçalves, RS. **Anais...2009.** Disponível em: <http://csbc2009.inf.ufrgs.br/anais/pdf/wei/st02_04.pdf>. Acesso em: 6 ago. 2018.

IZEKI, C. A.; NAGAI, W. A.; DIAS, R. M. C. Experiência no Uso de Ferramentas Online Gamificadas na Introdução à Programação de Computadores - In Anais do XXII Workshop de Informática na Escola (WIE) 22, 2016, Uberlândia, MG. **Anais..., 2016.** Disponível em: < http://www.br-ie.org/pub/index.php/wie/article/view/6648>. Acesso em: 18 mai. 2017.

JENKINS, T. **On the difficulty of learning to program.** In: Proceedings of the 3rd Annual LTSN-ICS Conference. Loughborough University, United Kingdom, August 2002. Disponível em: <http://78.158.56.101/archive/ics/events/conf2002/tjenkins.pdf>. Acesso em: 24 abr. 2017.

JENKINS, T. The motivation of students of programming. In: ACM SIGCSE Bulletin, Proceedings of the 6th Annual Conference on Innovation and Technology In Computer Science Education, **ITiCSE 2001, v.33, n.3, 2001.** Disponível em: <http://dl.acm.org/citation.cfm?id=377472>. Acesso em: 18 mai. 2017.

JESUS, A. M. DE; SILVA, G. M. DA DOJO`GO: Uma Forma Divertida e Colaborativa de se Aprender através de Dojo de Programação de Jogos - In Anais do XXII Workshop de Informática na Escola (WIE) 22, 2016, Uberlândia, MG. **Anais..., 2016.** Disponível em: < http://www.br-ie.org/pub/index.php/wie/article/view/6890 >. Acesso em: 16 mai. 2017.

KAPP, K. M. **The Gamification of Learning and Instruction**. Pfeiffer Publishing, 2012.
Kitchenham, B. and Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering.

KOLIVER, C.; DORNELES, R. V.; CASA, M. E. Das (muitas) dúvidas e (poucas) certezas do ensino de algoritmos. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO (CSBC), 24., 2004, Salvador, BA. **Anais..., 2004**. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/wei/2004/008.pdf>. Acesso em: 6 jul. 2018.

LOPES, B. G.; DUARTE, W. S.; NOGUEIRA, R.C.; LOPES,R. F. F. ; FERREIRA, D.J. Método de Ensino de Programação Mediada por Simulação: Um Estudo de Caso no Curso Técnico Integrado em Informática - In Anais do XXVII Simpósio Brasileiro de Informática na Educação (SBIE) 27, 2016, Uberlândia, MG. **Anais..., 2016.** Disponível em: < http://www.br-ie.org/pub/index.php/sbie/article/view/6714 >. Acesso em: 09 mai. 2017.

LOPES, C. C.; DUARTE, M. S. S. G.; SOUSA, E. A. D.; SOUZA, E. P. DE, PEREIRA, I. B. O Ensino de Algoritmos e Lógica de Programação como uma Ferramenta Pedagógica para Auxiliar a Aprendizagem de Matemática: Um Relato de Experiência - In Anais dos Workshops do V Congresso Brasileiro de Informática na Educação (CBIE) 5, 2016, Uberlândia, MG. **Anais..., 2016.** Disponível em: < http://www.br-ie.org/pub/index.php/wie/article/view/6601 >. Acesso em: 16 mai. 2017.

MARINHO, C. S. S.; MOREIRA, L. O.; COUTINHO, E. F.; PAILLARD, G. A. L.; NETO, E. T. L. Experiências no Uso da Metodologia Coding Dojo nas Disciplinas Básicas de Programação de Computadores em um Curso Interdisciplinar do Ensino Superior - In Anais dos Workshops do V Congresso Brasileiro de Informática na Educação (CBIE) 5, 2016, Uberlândia, MG. **Anais..., 2016.** Disponível em: < http://www.br-ie.org/pub/index.php/wcbie/article/view/7035 >. Acesso em: 04 mai. 2017.

MARTINS, L. A. S.; BRELAZ, A. S.; NASCIMENTO, G. R.; ALFAIA, R. M.; MARTINS, T.S. Ensinando Lógica de Programação aplicada à Robótica para alunos do Ensino Fundamental - In Anais do XXVII Simpósio Brasileiro de Informática na Educação (SBIE)

27, 2016, Uberlândia, MG. **Anais..., 2016.** Disponível em: < http://www.br-ie.org/pub/index.php/sbie/article/view/6678 >. Acesso em: 09 mai. 2017.

MARTINS, R. S.; REIS, R. J. A. DOS; MARQUES, A. B. Inserção da programação no ensino fundamental: Uma análise do jogo Labirinto Clássico da Code.org através de um modelo de avaliação de jogos educacionais - In Anais do XXII Workshop de Informática na Escola (WIE) 22, 2016, Uberlândia, MG. **Anais..., 2016.** Disponível em: < http://br-ie.org/pub/index.php/wie/article/view/6609>. Acesso em: 09 mai. 2017.

MASINI, EFS; MOREIRA, MA. **Aprendizagem significativa: a teoria de David Ausubel**. São Paulo: Centauro, 2001.

MATTOS, F. DE; FERREIRA, V.; ANACLETO, J. O Ensino de Programação de SCRATCH E Seu Impacto na Opção Profissional para as Meninas - In Anais do XXVII Simpósio Brasileiro de Informática na Educação (SBIE) 27, 2016, Uberlândia, MG. **Anais..., 2016.** Disponível em: < http://www.br-ie.org/pub/index.php/sbie/article/view/6710/0 >. Acesso em: 09 ago. 2017.

McKAY, J.; MARSHALL, P. The Dual Imperatives of Action Research. **Information Technology & People**, v. 14, n. 1, p. 46-59, 2001. Disponível em: <http://dx.doi. org/10.1108/09593840110384771>. Acesso em: 07 jun; 2018.

MOREIRA, M.A. **Teorias de aprendizagem**. São Paulo: EPU, 1999.

MOREIRA, M.A. **Uma abordagem cognitivista ao ensino da Física.** Porto Alegre: Editora de Universidade, 1983. Moreira, M.A., SOUSA, C.M.S.G. Organizadores prévios como recurso didático. Porto Alegre, RS: Instituto de Física, 1996.

MOTA, M. P. et al. Ambiente integrado à Plataforma Moodle para apoio ao desenvolvimento das habilidades iniciais de programação. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO (SBIE), 20., 2009, Florianópolis, Sc. **Anais..., 2009**. Disponível em: <http://www.proativa.virtual.ufc.br/sbie2009/conteudo/artigos/completos/61591_1.pdf > Acesso em: 6 ago. 2018.

MUNTEAN, C. I. Raising engagement in e-learning through gamification. The 6th International Conference on Virtual Learning ICVL 2011. **Anais. p.323–329, 2002**. Disponível em: <http://icvl.eu/2011/disc/icvl/documente/pdf/met/ICVL_ModelsAndMethodologies_paper42.pdf>. Acesso em: 28 maio. 2017.

NAGANO, L. H.; DIRENE, H. I. Ensino de lógica de programação baseado na indução-dedução através de exemplos - In Anais do XXVII Simpósio Brasileiro de Informática na Educação (SBIE) 27, 2016, Uberlândia, MG. **Anais..., 2016.** Disponível em: < http://www.br-ie.org/pub/index.php/sbie/article/view/6827>. Acesso em: 09 ago. 2017.

NOBRE, I. A. M.; MENEZES, C. S. Suporte à cooperação em um ambiente de aprendizagem para programação (SAmbA). In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO (SBIE), 13., 2002, SÃO LEOPOLDO, RS. **Anais..., 2002**. Disponível em: <http://br-ie.org/pub/index.php/sbie/article/view/195>. Acesso em: 7 jul. 2018.

NOVAK, J. D. **Aprender criar e utilizar o conhecimento: mapas conceituais como ferramenta de facilitação em escolas e empresas.** Lisboa: Plátano. 2000.

OLIVEIRA, C. E. T. DE; NOGUEIRA, E. C.; MOTTA, C. L. R. DA; MEIRELES, L. B. Relação estudante-professor: Educação Baseada na Construção de Jogos - In Anais do XXVI Simpósio Brasileiro de Informática na Educação (SBIE),26, 2015, Maceió, AL. **Anais..., 2015**. Disponível em: < http://www.br-ie.org/pub/index.php/sbie/article/view/5330 >. Acesso em: 7 ago. 2017.

OLIVEIRA, M. V. DE; RODRIGUES, L. C.; QUEIROGA, A. P. G. DE Material didático lúdico: uso da ferramenta Scratch para auxílio no aprendizado de lógica da programação - In Anais do XXII Workshop de Informática na Escola (WIE),22, 2016, Uberlândia, MG. **Anais..., 2016.** Disponível em: < http://www.br-ie.org/pub/index.php/wie/article/view/6842/0 >. Acesso em: 09 ago. 2017.

ONOFRIO, Roberto M. Gomes de. A web como interface no ensino musical. 2011. 144f. *Dissertação de Mestrado*. Instituto de Artes, Universidade Estadual de Campinas. Campinas, 2011.

PAWLOWSKI, C. S., ANDERSEN, H. B., TROELSEN, J., & SCHIPPERIJN, J. (2016). Children's physical activity behavior during school recess: a pilot study using gps, accelerometer, participant observation, and go-along interview. Disponível em: < https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0148786> Acesso em: 24 mai. 2018.

PELIZZARI, A; ET AL. Teoria da aprendizagem significativa segundo Ausubel. **Revista PEC, Curitiba,** v.2, n.1, p.37-42, jul. 2001/2002. Disponível em: <http://files.gpeceausp.webnode.com.br/200000393-74efd75e9b/MEQII-2013-TEXTOSCOMPLEMENTARES- AULA 5.pdf >. Acesso em: 29 jul. 2018.

PETERSEN, K., FELDT, R., MUJTABA, S., AND MATTSSON, M. (2008). Systematic mapping studies in software engineering. In Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering, **EASE'08,** pages 68–77, Swinton, UK, UK. British Computer Society.

PIMENTEL, E. P.; FRANÇA, V. F.; OMAR, N. A caminho de um ambiente de avaliação e acompanhamento contínuo da aprendizagem em Programação de Computadores. In: WORKSHOP DE EDUCAÇÃO EM COMPUTAÇÃO E 115 INFORMÁTICA DO ESTADO DE MINAS GERAIS, 2., 2003, Poços de Caldas, MG. **Anais..., 2003**. Disponível em: <http://200.17.137.110:8080/licomp/Members/jeanemelo/plonelocalfolderng.2006-04-

10.7475913377/PEP/Aulas19/WEIMIG2003EdsonPimentelArtigo1-aula19.pdf> .Acesso em: 6 ago. 2018.

PIVA JR, D.; FREITAS, R. L. Estratégias para melhorar os processos de abstração na disciplina de Algoritmos. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO (SBIE), 21., 2010. João Pessoa, PB. **Anais..., 2010**. Disponível em: <http://www.br-ie.org/pub/index.php/sbie/article/view/1464>. Acesso em: 18 jan. 2018.

RAABE, A. L. A.; SILVA, J. M. C. Um Ambiente para atendimento as dificuldades de aprendizagem de algoritmos. In: WORKSHOP DE EDUCAÇÃO EM COMPUTAÇÃO (WEI). 13., 2005, São Leopoldo, RS. **Anais..., 2005.** Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/wei/2005/003.pdf>. Acesso em: 6 set. 2017.

RAFALSKI, J. P.; SANTOS, O. L. DOS Uma experiência com a Linguagem Scratch no Ensino de Programação com Alunos do Curso de Engenharia Elétrica - In Anais do XXII Workshop de Informática na Escola (WIE),22, 2016, Uberlândia, MG. **Anais..., 2016.** Disponível em: < http://www.br-ie.org/pub/index.php/wie/article/view/6868/0 >. Acesso em: 09 jul. 2017.

RAPKIEWICZ, C. E. et al. **Estratégias pedagógicas no ensino de algoritmos e programação associadas ao uso de jogos educacionais.** Novas Tecnologias na Educação, v. 4, n. 2, Dez., 2006. Disponível em: <http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:CINTEDUFRGS+Novas+Tecnologias+na+Educa??o#0>. Acesso em: 13 abr. 2018.

RESNICK M. ET AL. Scratch: programming for all. **Communications of the ACM,** 52(11), 2009, pp. 60-67.

RESNICK, M. et al. Scratch: programming for all. **Communications of the ACM**, v.52, n.11, p.60-67, 2009. Disponível em: <http://dl.acm.org/citation.cfm?id=1592779>. Acesso em: 8 jan. 2018.

REVISTA BRASILEIRA DE INFORMÁTICA NA EDUCAÇÃO – RBIE (http://brie.org/pub/index.php/rbie/index);

RIBEIRO, J. P.; MANSO, M. A.; BORGES, M. A. F. Dinâmicas com App Inventor no Apoio ao Aprendizado e no Ensino de Programação - In Anais do XXII Workshop de Informática na Escola (WIE),22, 2016, Uberlândia, MG. **Anais..., 2016.** Disponível em: < http://www.br-ie.org/pub/index.php/wie/article/view/6645 >. Acesso em: 09 ago. 2017.

ROBERTS, E. An overview of MiniJava. **ACM SIGCSE Bulletin**, v.33, n.1, 2001. Disponível em: <http://dl.acm.org/citation.cfm?id=364525>. Acesso em: 8 jan. 2018.

ROBERTS, E. Conserving the seed corn: reflections on the academic hiring crisis. **ACM SIGCSE Bulletin,** v.31, n.4, Dez.1999. Disponível em: <http://wwwcs. stanford.edu/people/eroberts/papers/ConservingSeedCorn.pdf>. Acesso em: 8 jul. 2018.

ROCHA, P. S. et al. Ensino e aprendizagem de programação: análise da aplicação de proposta metodológica baseada no Sistema Personalizado de Ensino. **Renote, v. 8, n. 3, p. 1-11, 2010**. Disponível em: <http://seer.ufrgs.br/renote/article/view/18061>. Acesso em: 10 ago. 2018.

RODRIGUES JÚNIOR, M. C. Experiências positivas para o ensino de algoritmos. In: WORKSHOP DE EDUCAÇÃO EM COMPUTAÇÃO, 4. Feira de Santana, BA. **Anais... Sergipe, Bahia: SBC, 2004.** Disponível em: <http://www.uefs.br/erbase2004/documentos/weibase/Weibase2004Artigo001.pdf>. Acesso em: 14 mai. 2018.

RODRIGUES, L.C.; QUEIROGA, A. P. G. DE; OLIVEIRA, M. V. DE; MORE, A. T. Relato de experiência: curso de introdução à programação para crianças do ensino fundamental no IFSP Votuporanga - In Anais do XXII Workshop de Informática na Escola (WIE), 22, 2016, Uberlândia, MG. **Anais..., 2016.** Disponível em: < http://www.br-ie.org/pub/index.php/wie/article/view/6841 >. Acesso em: 09 ago. 2017.

RODRIGUES, M. C. Como ensinar programação? Informática – **Boletim Informativo**, Ano 1, n.1, ULBRA, Canoas, RS, 2002. Disponível em: <https://scholar.google.com.br/scholar?q=Rodrigues%2C+M.+C.+Como+Ensinar+Programa %C3%A7%C3%A3o%3F.&btnG=&hl=pt-BR&as_sdt=0%2C5#0>. Acesso em: 6 jul. 2018.

SALAZAR, R.; ODAKURA, V.; BARVINSKI, C. Scratch no ensino superior: motivação - In Anais do XXVI Simpósio Brasileiro de Informática na Educação (SBIE),26, 2015, Maceió, AL. **Anais..., 2015**. Disponível em: < http://www.br-ie.org/pub/index.php/sbie/article/view/5470 >. Acesso em: 19 ago. 2017.

SALGADO, L. A. Z. Desprogramamdo Programado. **Anais do VII Simpósio Nacional da ABCiber**, n. X, p. 1–11, 2013, Curitiba, Paraná. Disponível em:< http://www.abciber.org.br/simposio2013/anais/pdf/Artistico/26059arq39410803934.pdf>. Acesso em: 21 jul.2017.

SANTIAGO, A. D. V.; KRONBAUER, A. H. Um Modelo Lúdico para o Ensino de Conceitos de Programação de Computadores- In Anais do XXVII Simpósio Brasileiro de Informática na Educação (SBIE),27, 2016, Uberlândia, MG. **Anais..., 2016.** Disponível em: < http://www.br-ie.org/pub/index.php/sbie/article/view/6723>. Acesso em: 09 ago. 2017.

SANTOS, A. C. T. DOS; MONTEIRO, J. A.; MACHADO, K. C. T.; LINS, P. R. B.; RAMOS, T. A. R.; BATISTA, L. V. Ensino de programação para Olimpíada Brasileira de Informática - In Anais do XXI Workshop de Informática na Escola (WIE),21, 2015, Maceió,

AL. **Anais..., 2015**. Disponível em: < http://www.br-ie.org/pub/index.php/wie/article/download/5004/3414 >. Acesso em: 19 ago. 2017.

SANTOS, R. P.; COSTA, H. A. X. Análise de metodologias e ambientes de ensino para algoritmos, estruturas de dados e programação aos iniciantes em computação e informática. **INFOCOMP - Journal of Computer Sciense, v. 5, n. 1, p. 41–50, 2006**. Disponível em: <http://www.dcc.ufla.br/infocomp/artigos/v5.1/art06.pdf\nhttp://www.academia.edu/download/30554771/art06.pdf>. Acesso em: 19 ago. 2017.

SHEARD, J. et al. Analysis of research into the teaching and learning of programming. In: **Proceedings of the fifth international workshop on Computing education research (ICER)** '09, Ago. 2009. Disponível em: <http://portal.acm.org/citation.cfm?doid=1584322.1584334>. Acesso em: 23 abr. 2018.

SILVA, D. O. DA; GARCIA, V. R.; OLIVEIRA, I. G. DE; TRINDADE, D. F. G.; SGARBI, E. M.; NASCIMENTO, L. F. L. Despertando Jovens Talentos com o Conhecimento da Computação - In Anais do XXII Workshop de Informática na Escola (WIE), 22, 2016, Uberlândia, MG. **Anais..., 2016.** Disponível em: <http://www.br-ie.org/pub/index.php/wie/article/view/6865 >. Acesso em: 09 ago. 2017.

SILVA, T. R. DA; ARANHA, E. H. S.; SANTOS, F. G.; TAVARES, K. F. Um Relato de Experiência da Aplicação de Videoaulas de Programação de Jogos Digitais para Alunos da Educação Básica - In Anais do XXII Workshop de Informática na Escola (WIE) 2016, Uberlândia, MG. **Anais..., 2016.** Disponível em: < http://www.br-ie.org/pub/index.php/wie/article/view/6611 >. Acesso em: 07 ago. 2017.

Simpósio Brasileiro de Informática na Educação – SBIE (http://brie.org/pub/index.php/sbie/)

SLEEMAN, D. The challenges of teaching computer programming. **Communications of the ACM**, v. 29, n. 9, p. 840–841, 1986. Disponível em: <http://dl.acm.org/citation.cfm?id=214913>. Acesso em: 6 abr. 2018.

SOLOWAY, E. et al. What do novices know about programming ? In: BRADE, A., SHNEIDERMAN, B (Eds.). **Directions in human-computer interactions, Norwood, NJ: Ablex, p. 27–54, 1983.** Disponível em: <https://scholar.google.com.br/scholar?q=What+do+novices+know+about+programming%3F&btnG=&hl=pt-BR&as_sdt=0%2C5#0>. Acesso em: 7 mar. 2018.

SOUZA, M. S. C. DE; COSTA, F. A. M.; SILVA, V. L.; TERRA, D. C. Lord of Code: uma ferramenta de apoio ao ensino de programação In Anais do XXVII Simpósio Brasileiro de Informática na Educação (SBIE),27, 2016, Uberlândia, MG. **Anais..., 2016.** Disponível em: < http://www.br-ie.org/pub/index.php/sbie/article/view/6820 >. Acesso em: 01 ago. 2017.

SOUZA, N. S. DE. Uma Abordagem Dialética Para a Pesquisa Interdisciplinar em Interação Humano-Computador. 2016. 150f. *Dissertação de Mestrado*, Universidade Federal da Bahia, Salvador, 2016. Disponível em:< https://repositorio.ufba.br/ri/bitstream/ri/21704/1/Napoliana-msc-vsFinal.pdf >. Acesso em: 07 fev. 2018.

SOUZA, P. S. S. DE; MOMBACH, J. G. Ensino de Programação para Crianças através de Práticas Colaborativas nas Escolas - In Anais do XXII Workshop de Informática na Escola (WIE), 22, 2016, Uberlândia, MG. **Anais..., 2016.** Disponível em: < http://www.br-ie.org/pub/index.php/wie/article/view/6861 >. Acesso em: 01 ago. 2017.

TEIXEIRA, A. C.; ORO, N. T.; BATISTELA, F.; MARTINS, J. A. R.; PAZINATO, A. M. Programação de computadores para alunos do ensino fundamental: A Escola de Hackers - In Anais do XXI Workshop de Informática na Escola (WIE), 21, 2015, Maceió, AL. **Anais..., 2015**. Disponível em < http://www.br-ie.org/pub/index.php/wie/article/view/5002 >. Acesso em: 05 ago. 2017

THIOLLENT, M. **Pesquisa-Ação nas Organizações**. São Paulo: Atlas, 1997.

TOBAR, C. M. et al. Uma arquitetura de ambiente colaborativo para o aprendizado de programação. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO (SBIE), 12, 2001, Vitória, ES. **Anais..., 2001.** Disponível em: <https://www.icmc.usp.br/~joaoluis/sbie-01.pdf>. Acesso em: 13 ago. 2017.

TRIPP, D. **Pesquisa-ação: uma introdução metodológica.** Educação e Pesquisa, São Paulo, v. 31, n. 3, p. 443-466, set./dez. 2005. Disponível em: < http://www.scielo.br/pdf/ep/v31n3/a09v31n3.pdf >. Acesso em: 07 jun. 2018.

VERA, W. F,. M.; MIRANDA, A. M.; COSTA, W. N.; MATOS, E. S. Dó, Ré, Mergesort: um relato de experiência interdisciplinar de ensino de computação com matemática e música - In Anais dos Workshops do V Congresso Brasileiro de Informática na Educação (CBIE), 27, 2016, Uberlândia, MG. **Anais..., 2016.** Disponível em: < http://www.br-ie.org/pub/index.php/wcbie/article/viewFile/7053/4927 >. Acesso em: 01 ago. 2017.

Vogt, W. P. (1999). **Dictionary of statistics & methodology: A nontechnical guide for the social sciences (2nd ed.)**. Thousand Oaks, London, New Delhi: Sage.

WANGENHEIM, C. G. VON; NUNES, V. N.; SANTOS, G. D. DOS. Ensino de Computação com SCRATCH no Ensino Fundamental – Um Estudo de Caso. Revista Brasileira de Informática na Educação - **RBIE V.22 N.3 – 2014.** Disponível em:< http://www.gqs.ufsc.br/wp-content/uploads/2011/11/2885-5895-1-PB.pdf >. Acesso em: 12 ago. 2017.

WILSON, B. C.; SHROCK, S. Contributing to success in an introductory computer science course: a study of twelve factors**. ACM SIGCSE Bulletin**, v.33, n.1, p.184- 188, 2001. Disponível em: <http://dl.acm.org/citation.cfm?id=364581>. Acesso em: 7 mai. 2018.

Workshop de Informática na Escola - WIE (http://brie.org/pub/index.php/wie/index)

Workshops do Congresso Brasileiro de Informática na Educação – WCBIE (http://www.br-ie.org/pub/index.php/wcbie/index);

Zaina, L. A. M. **Projeto Multidisciplinar: uma Experiência Prática no Ensino de Programação em um Curso de Engenharia da Computação**. XXXIII Congresso Brasileiro de Ensino de Engenharia, 2005, Campina Grande. Disponível em: <http://www.abenge.org.br/cobenge/arquivos/14/artigos/SP-15-28545321805-1118683317399.pdf>. Acesso em: 11 mai. 2018.

APÊNDICE

APOSTILA

CONCEITOS BÁSICOS DE PROGRAMAÇÃO DE COMPUTADOR PARA MÚSICOS

LUCAS BLATT

lucas.blatt@lavid.ufpb.br

CONCEITOS PRELIMINARES

1 - O Computador

Um computador é uma máquina que manipula dados a partir de uma lista de instruções. Os computadores podem ser mecânicos (computador analógico) ou eletrônicos (computadores digitais).

1.1 Histórico dos Computadores

- MECÂNICOS
    - Ábaco 1000 A.C;
    - Ossos de Napier 1612;
    - Pascaline, Pascal 1642;
    - Tear automático, Jacquard 1801;
    - Máquina de diferenças, Babbage 1882;
    - Tabulador eletromecânico, Hollerith 1890.

- 1ª GERAÇÃO – ELETRO-ELETRÔNICOS
    - Z1, Z2, Z3 (relés), Konrad Zuse 1935;
    - ABC (válvulas), Atanosoff 1936;
    - MARK-1, 1941, 120 m2 - 10 multiplicações em 3 segundos;
    - ENIAC, 1946, 30 toneladas, 18000 válvulas, 5000 somas/s.

- 2ª GERAÇÃO – TRANSISTORES 1947
    - TX-0, 1957;
    - PDP-1, Digital, 1º computador comercial.

- 3ª GERAÇÃO – CIRCUITOS INTEGRADOS 1958
    - IBM 360, 1965;

- PDP-11, sucesso universitário.

- 4ª GERAÇÃO – MICROPROCESSADORES 1970
  - Intel 4004, 1971, 4 bits;
  - Intel 8008, 1972;
  - Altair 8800, 1974, montado em kits;
  - Apple, 1976, TV+Teclado, BASIC escrito por Bill Gates;
  - IBM-PC, 1981, computador pessoal, (projeto aberto, processador 8088 Intel, 16 bits, 4.77 MHz, 16 kb RAM, US$ 4400.).

1.2 Arquitetura Básica

Internamente os computadores modernos podem ser caracterizados por três partes distintas, a unidade central de processamento (UCP), a memória (MEM) e os dispositivos de entrada e saída (E/S), conforme esquema na Figura 1.1.

1.2.1 Unidade Central de Processamento (UCP).

A UCP (ou CPU da sigla em inglês, Central Processing Unit) é um conjunto de dispositivos eletrônicos responsável pelas operações de processamento referentes aos cálculos lógicos e matemáticos. Para execução das operações de processamento citadas, a UCP realiza sempre as seguintes tarefas[1]:

1. busca de uma instrução na memória;
2. interpretação de uma instrução;
3. execução de uma operação representada na instrução;
4. gravação de eventuais resultados do processamento;
5. reinício de todo o processo (caso necessário)

Fazem parte da maioria das UCPs as seguintes unidades:

[1] A sequência de tarefas descritas aqui constitui a base mínima de um algoritmo: entrada de dados, processamento de dados e saída de dados.

Unidade Aritmética e Lógica (UAL): responsável por realizar cálculos matemáticos mais complexos de maneira mais rápida.

Registradores: Memória temporária para armazenar dados a serem processados.

Unidade de Controle (UC): Controla o fluxo de dados na UCP: busca na memória, chamadas da UAL, controle geral das tarefas da UCP.

Relógio Gerador de pulsos que determinam um ciclo de tarefas da UCP. Em cada ciclo (ou pulso) a UCP realiza uma tarefa, assim quanto maior a frequência do relógio da UCP, mais tarefas esta pode realizar num mesmo intervalo de tempo.

1.2.2 Memória

A memória é o dispositivo responsável por armazenar dados. Os vários tipos de memória no computador são classificadas de maneira geral de acordo com a sua capacidade de leitura, escrita e volatilidade. São divididas em:

1. **RAM**: sigla para memória de acesso aleatório, é uma memória em que se pode ler e escrever, mas cujo conteúdo é perdido uma vez que o computador é desligado. É a memória principal do computador e a mais usada pelos aplicativos e sistema operacional.
2. **ROM** sigla para memória somente-leitura, como o nome diz só é possível ler seu conteúdo, mas não alterá-lo. Não se altera se o computador é desligado.
3. **Secundária** são dispositivos usados para armazenar grandes quantidades de informação em caráter não volátil. Na maioria das vezes é muito mais lenta que a RAM. Exemplo são os discos rígidos.

1.2.3 Dispositivos de Entrada e de Saída

Os dispositivos de entrada e saída de dados (E/S) são de suma importância pois qualquer informação que deva entrar ou sair do computador será feita através deles. Dentre os dispositivos de entrada podemos citar: teclado, mouse, câmera, digitalizador. Os dispositivos de saída podem ser: monitor[2], impressora, saída de som, por exemplo.

[2] Há monitores que são utilizados também como dispositivos de entrada, os chamados monitores

Os dispositivos de E/S se comunicam com o computador através de portas específicas de comunicação, como porta paralela, porta serial, porta USB, porta SCSI, porta Firewire, porta PS/2, e assim por diante. Cada porta compreende um tipo de conector específico, porém mais do que isso um protocolo de comunicação entre dispositivos.

O dispositivo de entrada padrão é o teclado, enquanto que a saída padrão é o monitor. Isto significa que sempre que não for explicitamente especificado, um programa tentará ler do teclado e escrever para o monitor.

2 - Algoritmos

2.1 Conceito de Algoritmo

Um algoritmo pode ser definido como uma sequência finita de passos (instruções) para resolver um determinado problema. Sempre que desenvolvemos um algoritmo estamos estabelecendo um padrão de comportamento que deverá ser seguido (uma norma de execução de ações) para alcançar o resultado de um problema.

De forma semelhante acontece na música, quando é composta, escrita ou executada. Ao realizarmos estas ações na música, seguimos uma série de normas, padrões de execução de ações que geram uma saída (música) para ser executada ou apreciada pelo público.

Para o desenvolvimento de um algoritmo eficiente é necessário obedecermos algumas premissas básicas no momento de sua construção:

- Definir ações simples e sem ambiguidade;
- Organizar as ações de forma ordenada
- Estabelecer as ações dentro de uma sequência finita de passos.

Para a realização de uma composição ou arranjo na música também é interessante obedecermos algumas regras para a construção:

- Procurar inspirações musicais em áreas do seu interesse;
- Organizar estas inspirações em forma de melodia e ritmos;
- Estabelecer condições para esta melodia, com partes que se repetem, harmonia que acompanha a melodia, início e fim, estrutura, entre outros.

Quando tratamos de algoritmos, observa-se que são capazes de realizar tarefas como:

1. Ler e escrever dados;
2. Avaliar expressões algébricas, relacionais e lógicas;
3. Tomar decisões com base nos resultados das expressões avaliadas;
4. Repetir um conjunto de ações de acordo com uma condição.

Construindo uma ponte entre as ações possíveis que um algoritmo pode fazer com o que uma música pode fazer, temos alguns pontos em comum, como por exemplo:

1. Interpretar e produzir dados que na música consideramos como som;
2. Apresentar expressões com significados distintos (acidentes, expressão, articulação);
3. Repetir um conjunto de ações de acordo com uma condição (voltas dentro da música).

Portanto, de maneira geral, o conceito de algoritmo pode ser relacionado com a música, onde inúmeras regras podem ser aplicadas bilateralmente.

Abaixo são apresentados exemplos de algoritmos. O algoritmo 1 é um exemplo simples de algoritmo (sem condições ou repetições) para troca de um pneu.

1. Desligar o carro;
2. Pegar as ferramentas (chave e macaco);
3. Pegar o estepe;
4. Suspender o carro com o macaco;
5. Desenroscar os 4 parafusos do pneu furado;
6. Colocar o estepe;
7. Enroscar os 4 parafusos;
8. Baixar o carro com o macaco;
9. Guardar as ferramentas.

No algoritmo 2 estão ilustradas as tarefas anteriormente mencionadas. Nas linhas de 2 a 4 pode-se observar a repetição de uma ação enquanto uma dada condição seja verdadeira, neste caso em específico, o algoritmo está repetindo a ação “esperar ônibus” enquanto a condição “ônibus não chega” permanecer verdadeira, assim que essa condição se tornar falsa (quando o ônibus chegar) o algoritmo deixará de repetir a ação “esperar ônibus”, e irá executar a linha 5.

Já nas linhas de 7 a 9, é possível observar um exemplo da execução (ou não execução) de uma ação com base na avaliação de uma expressão. Nesse trecho, o algoritmo avalia se a expressão “não tenho passagem” é verdadeira e em caso positivo, executa a ação “pegar dinheiro”. Caso a expressão “não tenho passagem” seja falsa (ou seja, a pessoa tem passagem) então o algoritmo irá ignorar a ação “pegar dinheiro” e irá executar a linha 10.

1. Ir até a parada;
2. Enquanto ônibus não chega faça;
3. Esperar ônibus;
4. Fim-enquanto;
5. Subir no ônibus;
6. Pegar a passagem;
7. Se não há passagem então;
8. Pegar dinheiro;
9. Fim-se;
10. Pagar o cobrador;
11. Troco dinheiro – passagem;
12. Enquanto banco não está vazio faça;
13. Ir para o próximo;
14. Fim-enquanto;
15. Sentar;
16. ...

O algoritmo 3 apresenta um exemplo simples para a execução de uma música que está em uma partitura sem a previsão de situações que poderão acontecer no decorrer desta execução.

1. Montar a estante de partituras;

2. Colocar sobre a estante a partitura impressa;
3. Retirar o instrumento do case;
4. Afinar o instrumento;
5. Executar a música do início ao fim;
6. Guardar o instrumento no case;
7. Guardar a partitura;
8. Desmontar a estante.

2.2 Partes de Um Algoritmo

Um algoritmo quando programado num computador é constituído pelo menos das 3 partes, sendo elas: Entrada de dados; Processamento de dados; Saída de dados.

Na parte de entrada, são fornecidas as informações necessárias para que o algoritmo possa ser executado. Estas informações podem ser fornecidas no momento em que o programa está sendo executado ou podem estar embutidas dentro do mesmo.

Na parte do processamento são avaliadas todas as expressões algébricas, relacionais e lógicas, assim como todas as estruturas de controle existentes no algoritmo (condição e/ou repetição).

Na parte de saída, todos os resultados do processamento (ou parte deles) são enviados para um ou mais dispositivos de saída, como: monitor, impressora, ou até mesmo a própria memória do computador.

Ao estabelecermos uma relação com a música, também temos pontos fundamentais que fazem parte de uma música.

Na entrada de dados podemos considerar um improviso de algum instrumentista, onde são definidas tonalidades e estilos e ritmos. Sobre essas informações o músico desenvolve o seu improviso.

Quando tratamos de interpretação e processamento, seguindo o exemplo anterior, o músico avalia as informações indicadas na entrada e faz o devido processamento, criando o seu improviso. Outro exemplo é a leitura de uma partitura, onde o músico avalia todas as informações contidas nesta partitura e processa os dados.

E ao falarmos da saída, temos a música. O resultado da leitura, avaliação e processamento geram uma saída que pode ser apresentada de inumes maneiras, através de um instrumento musical, do canto, ou outras maneiras.

2.3 Representações de um Algoritmo

2.3.1 Fluxograma

Os fluxogramas são uma apresentação do algoritmo em formato gráfico. Cada ação ou situação é representada por uma caixa. Tomadas de decisões são indicadas por caixas especiais, possibilitando ao fluxo de ações tomar caminhos distintos. Esta representação é a mais utilizada.

A Figura 1 representa um algoritmo na forma de um fluxograma. O início e o fim do algoritmo são marcados com uma figura elíptica; as ações a serem executadas estão em retângulos; sendo que as estruturas de controle condicionais estão em losangos e indicam duas possibilidades de prosseguimento do algoritmo, uma para o caso da expressão avaliada (condição) ser verdadeira e outra para o caso de ser falsa.

No exemplo da Figura 1, a primeira ação é executada ('abrir forno') e então a segunda expressão é avaliada ('fogo aceso?') como verdadeira ou falsa; caso seja verdadeira, o algoritmo prossegue para a ação à esquerda ('botar lenha'); caso seja falsa, o algoritmo executa a ação à direita ('acender fogo'). Em seguida, para qualquer um dos casos, a próxima ação a ser executada é ('assar pão').

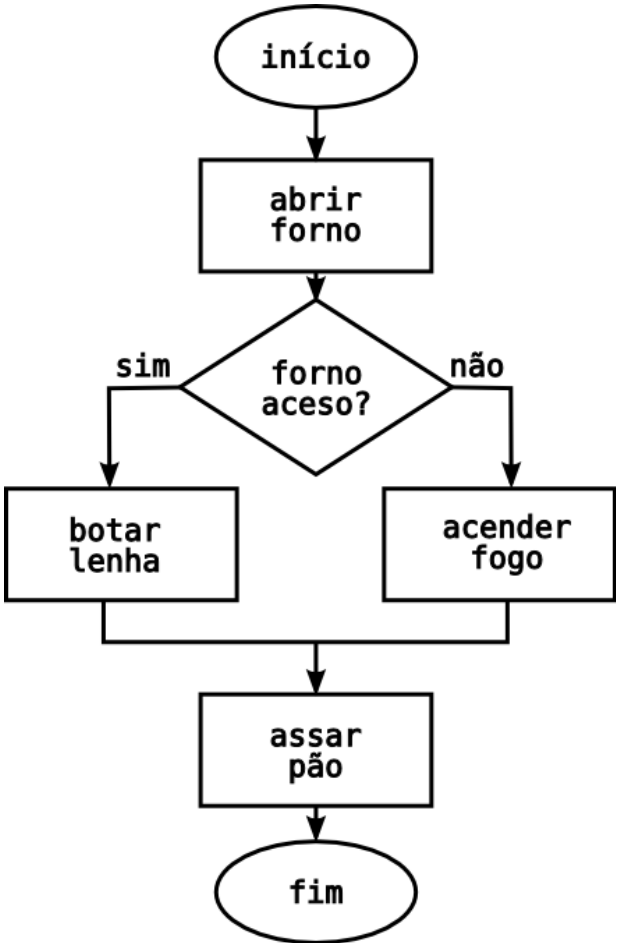


Figura 1 - Algoritmo representado em forma de um fluxograma.

2.3.2 Descrição Narrativa

Forma em que os algoritmos são expressos em linguagem natural, sem o uso de códigos ou desenhos para a representação.

Exemplo: Troca de um pneu furado.

1. Afrouxar ligeiramente as porcas;
2. Suspender o carro;
3. Retirar as porcas;
4. Retirar o pneu furado;
5. Colocar o pneu reserva;

6. Apertar as porcas;
7. Abaixar o carro;
8. Dar o aperto final nas porcas.

Esse é um exemplo de representação de um algoritmo de forma narrativa, onde os passos são descritos de maneira simples, sem exigir conhecimento aprofundado do leitor. A descrição narrativa também pode vir acompanhada de imagens que ilustram o que está sendo descrito e tem como finalidade ajudar o leitor. Abaixo é apresentado outro exemplo de um algoritmo narrado.

Exemplo: Cálculo da média de um aluno.

1. Obter as notas da primeira e da segunda prova;
2. Calcular a média aritmética entre as duas notas;
3. Se a média for igual ou maior que 6.0, o aluno foi aprovado, senão ele foi reprovado.

No exemplo acima já é possível observar que o algoritmo apresenta uma condição e com isso duas saídas possíveis. A saída vai depender da entrada apresentada pelo usuário.

2.3.2 Pseudocódigo

Forma de representação de algoritmos que se assemelha muito ao modo como os programas são escritos. Esta forma de representação permite que os algoritmos nela representados possam ser traduzidos, quase que diretamente, para uma linguagem de programação.

Exemplo de um algoritmo em pseudocódigo.

1. Algoritmo <nome_do_algoritmo>
2. <declaração de variáveis>
3. <subalgoritmos>
4. Início
5. <corpo do algoritmo>
6. Fim.

Exemplo: Cálculo da média de um aluno

1. Algoritmo Média
2. Var N1, N2, Média : real

3. Início
4. Leia N1, N2
5. Média ß (N1 + N2) / 2
6. Se Média >= 6.0
7. Então
8. Escreva “Aprovado”
9. Senão
10. Escreva “Reprovado”
11. Fim

Ao voltarmos o olhar para a música, temos possibilidades distintas de representação musical e que pode ser por pseudocódigo, fluxograma, narrativa, entre outros. A forma de se representar uma música vai depender do compositor ou arranjador, bem como a finalidade e espaço. Representação musical no âmbito popular normalmente é distinta de uma representação num espaço contemporâneo.

A letra de uma música normalmente é representada de forma narrativa. Uma composição de cunho contemporâneo pode ser representada através de fluxogramas e uma música popular é normalmente representada como um pseudocódigo.

A figura abaixo mostra o exemplo de um fluxograma de progressão harmônica.

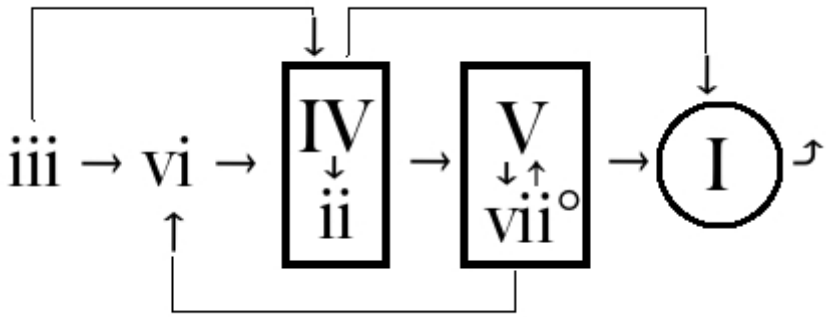


Figura 2 - Fluxograma de progressão harmônica

3 - Computador e Linguagem

3.1 Software x Hardware

Hardware é a parte física do computador, ou seja, o conjunto de aparatos eletrônicos, peças e equipamentos que fazem o computador funcionar. A palavra hardware pode se referir também como o conjunto de equipamentos acoplados em produtos que precisam de algum tipo de processamento computacional. A ciência que estuda o hardware é conhecida como arquitetura de computadores.

Entre alguns dos hardwares que podemos elencar estão:

- Monitor
- Memória
- Placas de som
- Placa de vídeo
- Chips
- Processadores
- Cooler (ventilador).

Esse termo se aplica para todo tipo de equipamento tecnológico, incluindo celulares, Smart TVs, dispositivos de leitura (como o Kindle) ou Wearables (relógios inteligentes, por exemplo).

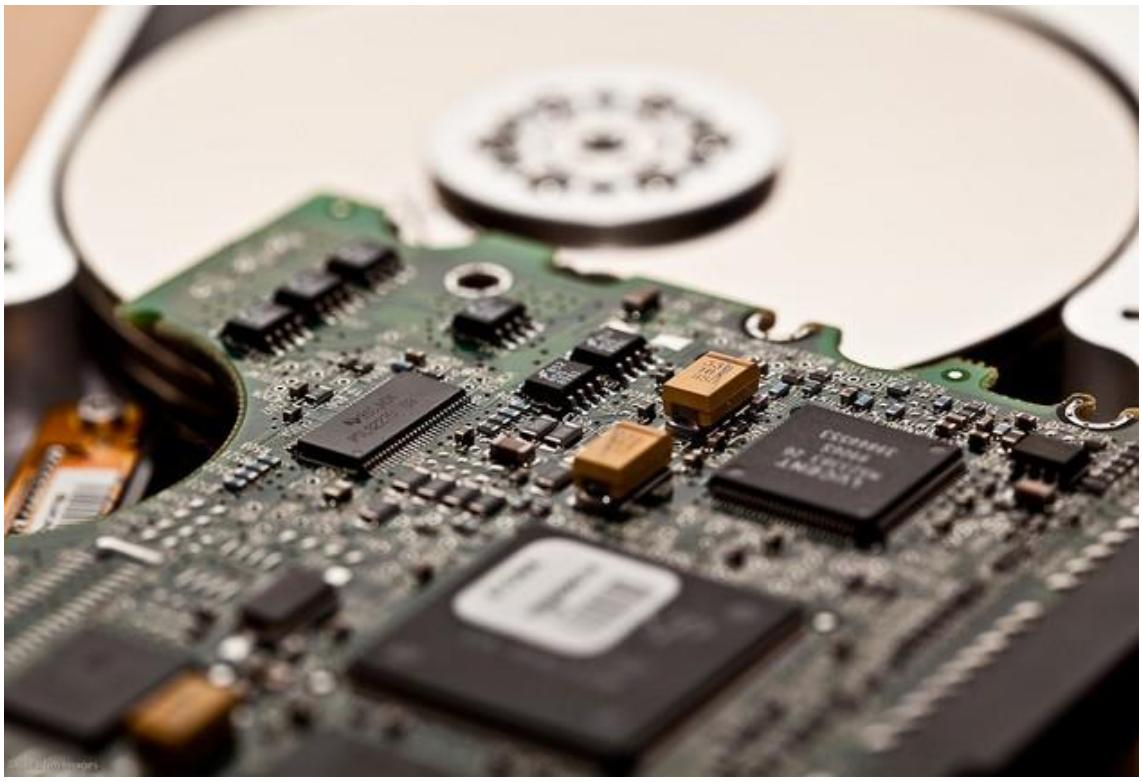


Figura 3 - Componentes que formam o Hardware.

Diferentemente do hardware, o software é a parte lógica do computador. Software é a manipulação, instrução de execução, redirecionamento e execução das atividades lógicas das máquinas. Os softwares são elaborados a partir de inúmeros códigos e customizados de acordo com as tecnologias dos diferentes fabricantes de computadores. Esses códigos que nada mais são do que instruções para o computador, são escritos em alguma linguagem de programação e será vista posteriormente.

Os sistemas operacionais são um exemplo disso. Sistemas como os famosos Windows, Linux, Android e iOS são configurados de diferentes maneiras.

Outros exemplos são:

- Navegadores
- Antivírus
- Programas de edição de textos e imagens
- Programas de reprodução de vídeos.

Qualquer outra aplicação no computador, é denominados de softwares.

Os softwares podem ainda ser classificados em:

- Softwares de Sistemas: permite que o usuário interaja com o computador e suas partes. Ex: firmware, drivers, etc.
- Softwares Aplicativos: permite que através de seu uso, o usuário faça uma tarefa específica. Ex: editores de texto, planilhas eletrônicas, etc.

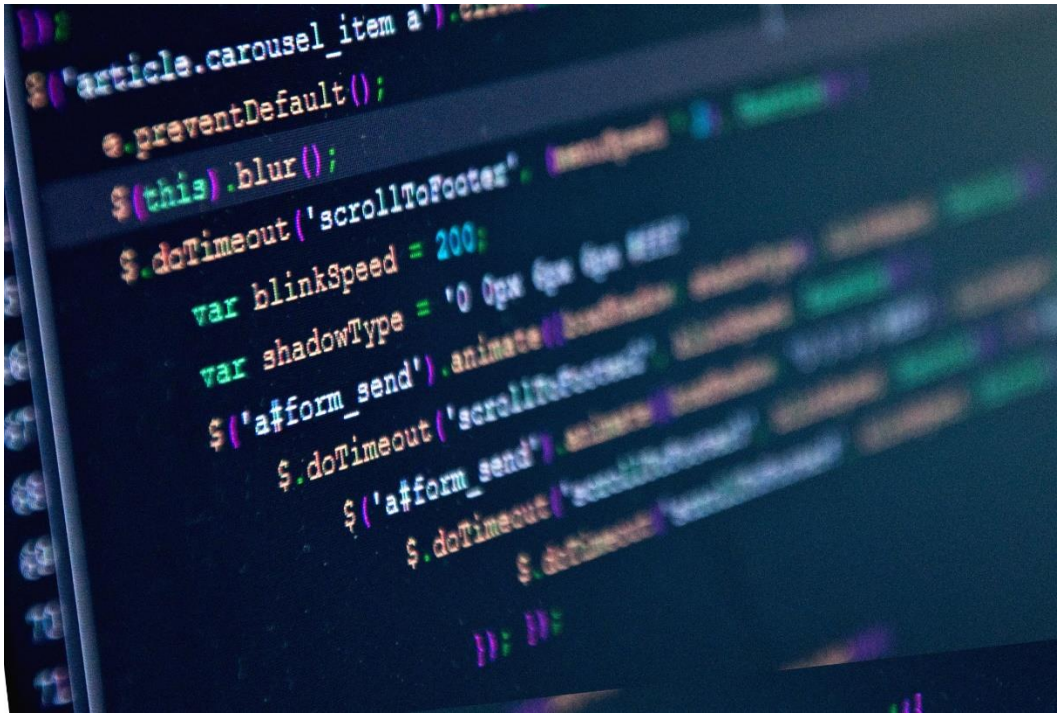


Figura 4 - Código de um Software

Na música podemos considerar que os conceitos de software e hardware são similares. A música possui uma linguagem própria com instruções e códigos que combinados e ordenados respeitando a sintaxe torna-se similar a um software. Sozinho, um código musical e suas instruções (partitura musical) não é capaz de produzir qualquer saída (som), semelhante a um software.

Para que a interpretação de um código musical ganhe vida, é necessário algum tipo de interface e processamento de dados. Essa ação ocorre através do músico e seu instrumento que podemos considerar com hardware.

Outro exemplo que podemos citar são os aparelhos eletrônicos, como sintetizadores sonoros. Um teclado é composto pelo hardware (teclas, display, saídas de áudio, autofalantes) e softwares (banco de sons, tranpositores, banco de ritmos).

3.2 Linguagens

Qualquer tipo de informação que deva ser transferida, processada ou armazenada deve estar na forma de uma linguagem. A linguagem é imprescindível para o processo de comunicação. Duas pessoas que se falam o fazem através de uma linguagem em comum, a linguagem natural. Da mesma forma, duas máquinas trocam informação por uma linguagem, que neste caso mais técnico e restrito, se chama protocolo. Do mesmo modo, um computador armazena suas instruções em código de máquina. Estas diferentes linguagens não podem ser traduzidas diretamente entre si, pois além de serem representadas de modos diferentes, também referem-se a coisas muito distintas. Para que um ser humano possa programar, armazenar e buscar informações num computador, é necessário que saiba instruí-lo na sua linguagem de máquina ou numa linguagem intermediária (uma linguagem de programação) que possa ser facilmente traduzida para o computador.

3.2.1 Linguagem Natural

A linguagem natural é a maneira como expressamos nosso raciocínio e trocamos informação. Como é a expressão da cultura de uma sociedade, desenvolvida através das gerações e em diferentes situações, raramente constitui um sistema de regras rígidas que possa ser implementada numa máquina ou que possa ser transcrita logicamente. Além da linguagem falada, fazem parte da nossa comunicação gestos e posturas, que não podem ser diretamente adaptados para compreensão de uma máquina. Por fim, toda a comunicação eficiente pressupõe um conhecimento prévio comum entre os interlocutores, por exemplo a mesma língua, a mesma bagagem cultural e assim por diante.

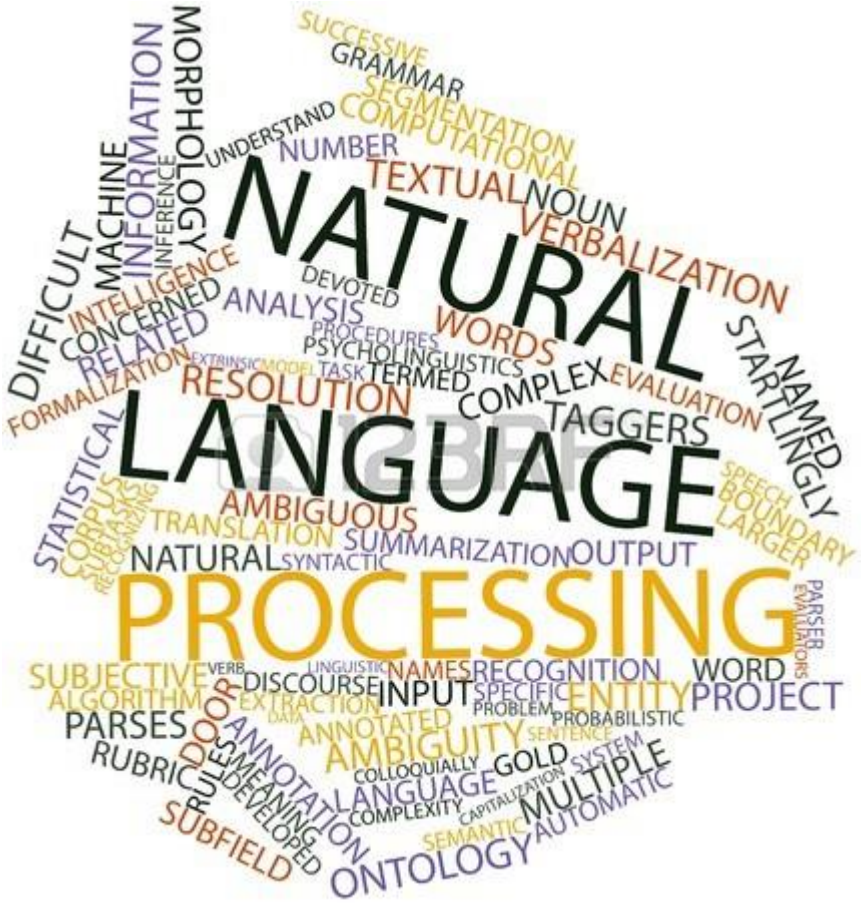


Figura 5 - Linguagem Natural é o que usamos como meio de comunicação todos os dias entre os humanos.

Ao contrário dos seres humanos, as máquinas (dentre elas os computadores) são projetados para executar tarefas bem determinadas a partir de determinadas instruções. Um computador não é por si só uma máquina inteligente no sentido que não pode aprender com a própria experiência para melhorar seu comportam ente futuro[3]. Ao contrário, um computador é somente capaz de realizar estritamente as tarefas que lhe forem delegadas e que façam parte do conjunto daquelas ações que ele pode executar. Neste sentido, é necessário compreender que tipo de instruções podem ser executadas pelos computadores para que possamos

[3] Diversos esforços vêm sendo despendidos dentro do meio científico para equipar computadores com esta capacidade, o campo de pesquisa que cuida desse tipo de tarefa é conhecido como Inteligência Artificial.

programá-los — instruí-los com a sequência de ações necessárias para resolver um determinado problema — de modo que realizem a tarefa do modo desejado.

Figura 6 - Linguagem de Máquina

3.2.2 Linguagem de Máquina e Assembler

Além do fato de o computador necessitar que lhe instruam com ações bem específicas, estas ações devem ser passadas para o computador numa linguagem que ele possa entendê-las, chamada linguagem de máquina. Esta linguagem é composta somente por números, representados de forma binária, que, sob o ponto de vista do computador, representam as operações e os operandos que serão usados no processamento do programa. Para um ser humano, a linguagem de máquina é dificílima de se compreender. Assim, existe uma linguagem representada por comandos mas que reproduz as tarefas que serão executadas dentro do computador, a linguagem de montagem (assembly). Entretanto, mesmo a linguagem de montagem é difícil de programar e os programas feitos para um determinado processador, por conterem instruções específicas deste, não funcionarão em um processador de outro tipo. Com ilustração, abaixo é mostrado o início de um programa que escreve a frase “Olá Mundo” no monitor. Na coluna da esquerda está o endereço relativo de memória, na coluna do centro o programa escrito em linguagem de máquina e na coluna da direita a representação em caracteres ASCII. Teoricamente, o programa poderia ser escrito diretamente em linguagem de máquina, como mostrado abaixo, entretanto a sintaxe do mesmo é muito pouco compreensível e a probabilidade de erro para o seu desenvolvimento seria muito grande.

```
00000000  7F 45 4C 46 01 01 01 00 00 00 00 00 00 00 00 00  .ELF............
00000010  02 00 03 00 01 00 00 00 D0 82 04 08 34 00 00 00  ............4...
00000020  BC 0C 00 00 00 00 00 00 34 00 20 00 07 00 28 00  ........4. ...(.
00000030  24 00 21 00 06 00 00 00 34 00 00 00 34 80 04 08  ..!.....4...4...
00000040  34 80 04 08 E0 00 00 00 E0 00 00 00 05 00 00 00  4...............
```

3.2.3 Linguagens de Programação

Para facilitar a tarefa de programar um computador, foram criadas várias linguagens de programação. Estas linguagens são um maneira de tentar escrever as tarefas que o computador vai realizar de maneira mais parecida com a linguagem natural. Embora ainda seja muitas vezes complexo em comparação com a linguagem natural, um programa escrito em uma linguagem de programação é muito mais fácil de ser implementado, compreendido e modificado.

As linguagens de programação são um meio termo entre a linguagem de máquina e a linguagem natural. Deste modo são classificadas de acordo com o nível entre a linguagem natural ou de máquina que ocupam. As linguagens muito parecidas com linguagem de máquina são chamadas de linguagens de baixo nível e suas instruções parecem-se muito com aquelas que serão executadas pelo processador. As linguagens de alto-nível são as que guardam mais semelhanças com a linguagem natural. Exemplo de linguagens de baixo nível é a linguagem de montagem (assembly). Exemplos de linguagens de alto-nível são: Pascal, C, Fortran, Java, Perl, Python, Lisp, PHP, LilyPond, entre outras.

Como o processador não pode executar o código numa linguagem de programação, esta deve ser traduzida em código de máquina antes de ser executada. Este processo é chamado de compilação (representado na Figura 3) e é responsável por converter os comandos da linguagem de programação nas instruções em código de máquina que o processador poderá utilizar.

Figura 7 - Compilação: o programa em linguagem de programação é transformado em instruções em linguagem de máquina (que o processador pode executar).

Por exemplo, o código de máquina da seção 3.2.2 foi gerado pelo programa a seguir, escrito na linguagem de programação C. Esse programa, depois de compilado, escreve frase "Olá Mundo" no monitor. A compilação, isto é, a tradução do programa em C para linguagem de máquina, produz algo parecido com o que foi é mostrado na seção 3.2.2, para o caso de um processador da família 80386, usados em PCs.

```
#include <stdio.h>

int main(){

        printf("Olá Mundo\n");

}
```

A primeira linha (#include) inclui algumas bibliotecas de instruções que facilitarão a programação. A linha seguinte indica que esta é a parte principal (main) do programa; o que estiver dentro do bloco delimitado por chaves { } será executado. Finalmente, a próxima linha imprime (printf) o argumento ("Olá Mundo") no monitor. Um programa escrito em linguagem de máquina, como contém instruções específicas de um processador, só poderá ser utilizado naquele processador ou em similares.

Em contrapartida, uma linguagem de programação, como contém somente instruções abstratas do que fazer, pode ser compilado para qualquer código de máquina. Em resumo, ao invés de escrever um programa em código de máquina para cada família de processadores, escreve-se o mesmo código numa linguagem de programação e este é compilada por um compilador específico daquela arquitetura.

3.2.4 Pseudocódigo

O pseudocódigo é uma maneira intermediária entre a linguagem natural e uma linguagem de programação de representar um algoritmo. Ela utiliza um conjunto restrito de palavras-chave, em geral na língua nativa do programador, que tem equivalentes nas linguagens de programação. Além disso, o pseudocódigo não requer todo a rigidez sintática necessária numa linguagem de programação, permitindo que o aprendiz se detenha na lógica do algoritmos e não no formalismo da sua representação. Na medida em que se obtém mais familiaridade com os algoritmos, então o pseudocódigo pode ser traduzido para uma linguagem de programação.

Exemplo de pseudocódigo:

```
leia (x, y) {Esta linha é um comentário}
se x > y então
        escreva (“x é maior”)
senão
        se y > x então
                escreva (“y é maior”)
        senão
                escreva (“x e y são iguais”)
        fim-se
fim-se
```

Exemplo 1 - Pseudocódigo que compara dois números

No exemplo acima é mostrado um pseudocódigo escrito em português para escrever o maior valor entre, x ou y. As palavras **leia**, **se**, **então**, **senão**, **senão-se**, **fim-se** e **escreva** são palavras-chave que representam estruturas presentes em todas as linguagens de programação. Entretanto, no pseudocódigo não é necessário se preocupar com detalhes de sintaxe (como ponto-e-vírgula no final de cada expressão) ou em formatos de entrada e saída dos dados. Deste modo, o enfoque no desenvolvimento do algoritmo fica restrito a sua lógica em si, e não na sua sintaxe para representação em determinada linguagem.

3.2.5 Linguagem Musical

Num sentido amplo podemos afirmar que a música é linguagem na medida em que todas as produções de cultura são formas de pensamento instauradas em contextos sociais. A música é comunicação, e por sua vez se faz de diversas maneiras, seja instrumental ou vocal, com um ou vários instrumentos.

A linguagem falada quando é cantada é mais fácil de ser compreendida, pois o ser humano a utiliza diariamente, torna-se mais fácil quando o vocabulário empregado tende a ser mais informal e tratando de temas imediatos.

A linguagem instrumental exige um pouco mais de reflexão, às vezes é mais fácil perceber a emoção do que a mensagem contida, e esta acaba sendo mais difícil de ser compreendida.

O recorte que será abordado nesta apostila é o "código", ou seja, a notação musical. Notação musical é o nome dado ao sistema de escrita que representa graficamente uma peça musical, ou um conjunto de sinais gráficos que representam uma organização de sons, permitindo que um intérprete que a execute semelhante a ideia do escritor, compositor ou arranjador. O sistema de notação mais utilizado é o sistema gráfico ortocrónico ocidental que utiliza símbolos grafados sobre uma pauta de 5 linhas paralelas, comumente chamada de pentagrama[4].

CONCEITOS DE PROGRAMAÇÃO

4 - LilyPond

O LilyPond é uma ferramenta baseada em TeX, disponível para várias plataformas e que permite a escrita de partituras musicais com base numa filosofia de linguagem de marcação. A sua utilização permite gerar documentos em diferentes formatos (como por exemplo o Postscript ou o PDF) viabilizando impressões de qualidade.

Uma vez que a Lilypond é baseado em TeX, a integração desta ferramenta com outros documentos de texto é possível, permitindo criar documentos que não contenham exclusivamente partituras musicais. Entre muitas das suas outras funcionalidades, esta

[4] Conjunto formado por 5 linhas e 4 espaços traçado na horizontal.

tecnologia permite também criar ficheiros em formato MIDI das músicas em que o utilizador se encontra a trabalhar.

LilyPond é principalmente escrito em C ++ e usa o Scheme (interpretado pelo GNU Guile) como sua linguagem de extensão, permitindo a personalização do usuário. Tem uma base de código relativamente grande; em 10 de março de 2017, a fonte inclui mais de 600.000 linhas de C ++, 140.000 linhas de Scheme e 120.000 linhas de código Python.

Ele usa uma notação de texto simples para entrada de música, que o LilyPond interpreta e processa em uma série de etapas. No estágio final, a notação musical é enviada para PDF (via PostScript) ou outros formatos gráficos, como SVG ou PNG . O LilyPond também pode gerar arquivos MIDI que correspondem à saída de notação musical.

O LilyPond é um aplicativo baseado em texto, por isso não contém sua própria interface gráfica de usuário para auxiliar na criação de pontuação. No entanto, ele possui uma linguagem de entrada flexível que se esforça para ser simples, facilitando a curva de aprendizado para novos usuários. LilyPond adere ao paradigma WYSIWYM; o fluxo de trabalho para composição da notação musical com o LilyPond é semelhante ao da preparação de documentos com o LaTeX .

LilyPond suporta notação musical experimental. Suas instalações de guitarra suportam afinações alternativas, como afinações de terços principais.

4.1 Linguagem LaTeX

Como descrito acima, o LilyPond é uma ferramenta que utiliza a linguagem semelhante ao LaTeX e de marcação. O LaTeX é um conjunto de macros para o programa de diagramação de textos TeX, utilizado amplamente na produção de textos matemáticos e científicos, devido a sua alta qualidade tipográfica. Entretanto, também é utilizado para produção de cartas pessoais, artigos e livros sobre assuntos diversos.

O sistema LaTeX fornece ao usuário um conjunto de comandos de alto nível, facilitando dessa forma sua utilização por iniciantes. Possui abstrações para lidar com bibliografias, citações, formatos de páginas, referência cruzada e tudo mais que não seja relacionado ao

conteúdo do documento em si. Foi desenvolvido na década de 80, por Leslie Lamport, estando atualmente na versão denominada LaTeX 2e.

4.2 Interfaces Derivadas do LilyPond

O LilyPond é um programa que possui código aberto e livre, o que permite que usuários, programadores e desenvolvedores do mundo todo façam mudanças que melhorem o desempenho do programa. Algumas derivações que trazem interfaces gráficas mais amigáveis surgiram e ganham novos adeptos. Abaixo serão apresentadas algumas.

4.2.1 Frescobaldi

O Frescobaldi é um editor de partituras baseado no LilyPond. Este editor é amplo, porém leve e fácil de usar além de ser um software livre, disponível sob a GNU General Public License. É projetado para rodar em todos os principais sistemas operacionais (GNU/Linux, Mac OS X e Microsoft Windows). Seu nome é derivado de Girolamo Frescobaldi, um compositor italiano de música de teclado do final da Renascença e início do período Barroco.

Em Abril de 2009, Frescobaldi ganhou um prêmio "HotPick" na LinuxFormat[1]. Em Maio daquele mesmo ano, o TuxRadar o listou como uma das 100 jóias de fonte aberta. Frescobaldi é escrito em Python e utiliza o PyQt4 para sua interface gráfica.

Entre os recursos disponíveis estão:

- Editor de texto com destaque de sintaxe e preenchimento automático;
- Visualização da música com recurso avançado de apontar e clicar;
- Tocador MIDI para amostra dos arquivos MIDI gerados pelo LilyPond;
- Assistente de notas para configurar uma partitura musical;
- Gerenciador de recortes para armazenar e aplicar fragmentos de texto, de modelos ou de scripts;
- Uso de múltiplas versões do LilyPond;
- Navegador da documentação e ajuda do LilyPond embutidos;
- Interface de usuário com cores, fontes e atalhos de teclado configuráveis;

- Traduzido para o Holandês, Inglês, Francês, Alemão, Italiano, Tcheco, Russo, Espanhol, Galício, Turco, Polonês, Português do Brasil e Ucraniano;

Além disso permite a:

- Mudança do tom da música
- Mudança da música do relativo para o absoluto e vice-versa
- Mudança do idioma utilizado para o nome das notas
- Mudança do ritmo (dobrar, reduzir à metade, adicionar ou remover pontos, colar) etc.
- Hifenizar a letra usando dicionários de hifenização de processadores de texto
- Adicionar facilmente chaves, dinâmica, articulação usando o painel Quick Insert
- Atualizar a sintaxe do LilyPond usando o convert-ly, com exibição das diferenças.

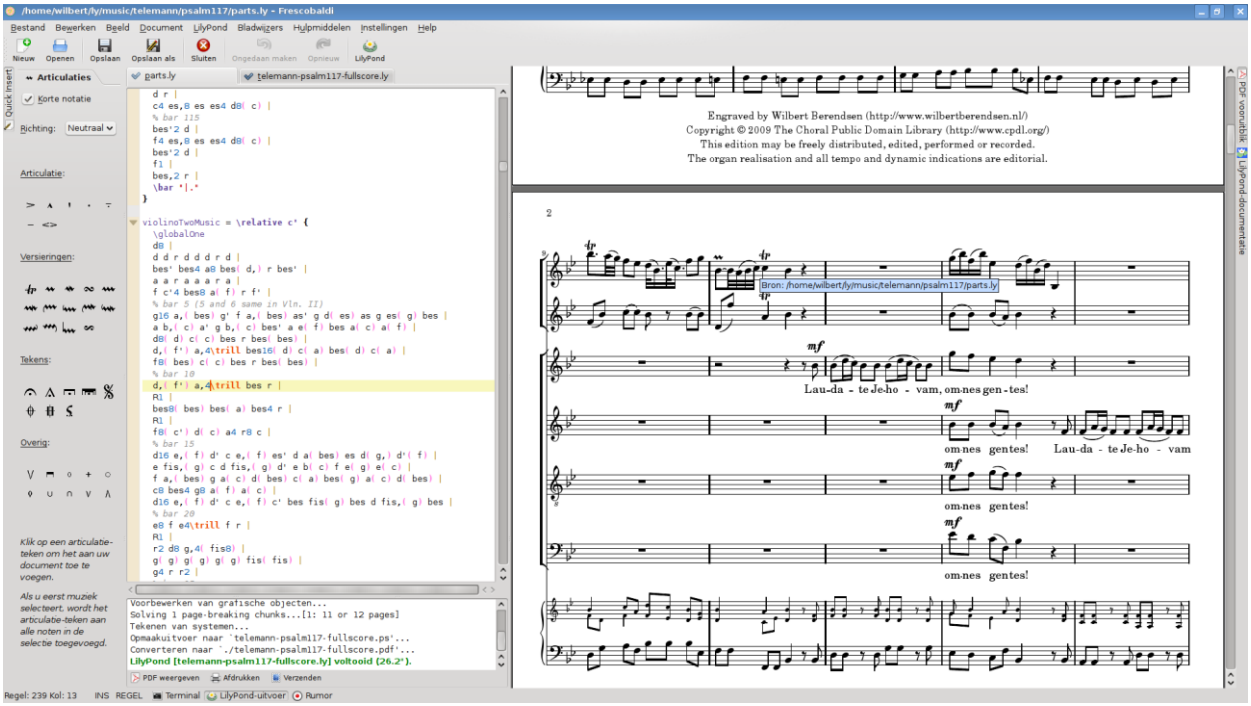


Figura 8 - Interface Gráfica do Frescobaldi

O Frescobaldi será o editor utilizado como IDE[5] neste curso, pois dispõe de inúmeras funcionalidades importantes para o ensino de programação.

4.2.2 Denemo

Denemo é uma interface gráfica livre para notação musical, baseada em LilyPond, um programa para gravação de partituras musicais. O programa começou a ser desenvolvimento em 1999 usando GTK+ 2 ou 3 e funciona no Linux, no Microsoft Windows e no Mac OS X.

O Denemo ajuda na escrita das notas para publicação e permite ao usuário entrar com as notas rapidamente, ao compor simultaneamente através do gravador musical do LilyPond. A Música pode ser digitada utilizando-se o teclado de um PC, extraída de uma entrada MIDI, ou tocada em um microfone conectado em uma placa de som. O programa reproduz através de um sampler interno. O Denemo inclui scripts para executar testes musicais e praticar exercícios para propósitos educacionais.

[5] IDE - Integrated Development Environment ou Ambiente de Desenvolvimento Integrado, é um programa de computador que reúne características e ferramentas de apoio ao desenvolvimento de códigos de programação e software com o objetivo de agilizar este processo

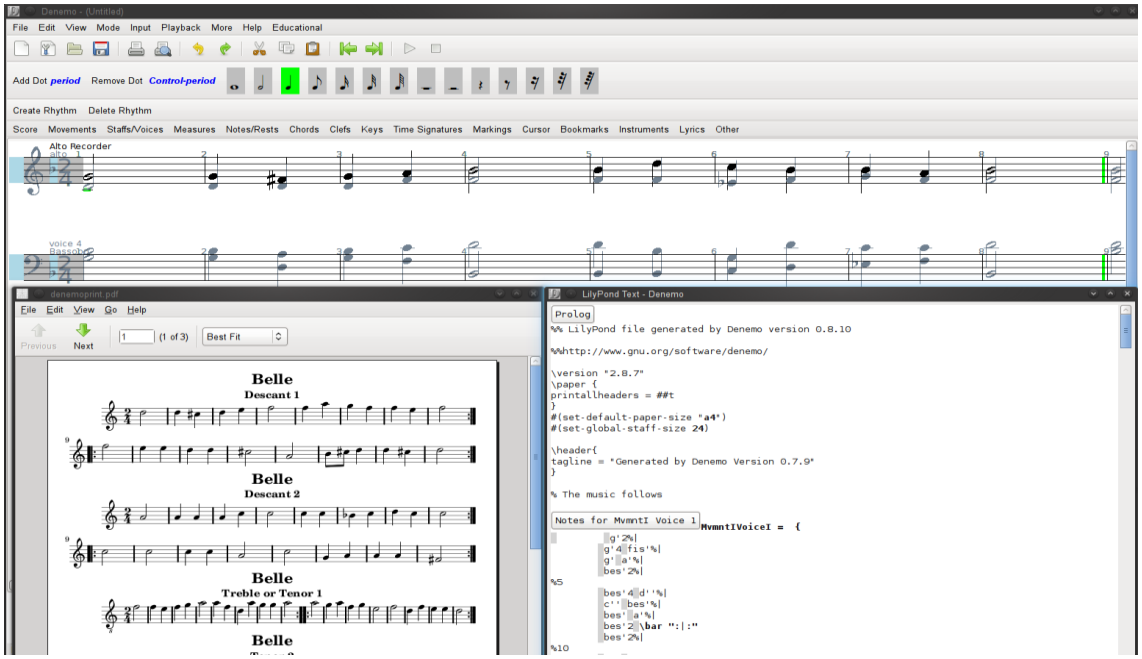


Figura 9 - Interface Gráfica do Denemo

4.2.3 LilyBin

Outra possibilidade de utilizar o LilyPond é através do LilyBin, que é um editor de partituras e que funciona online. O princípio de funcionamento segue as regras dos outros softwares baseados em LilyPond, onde as regras de sintaxe precisam ser respeitas.

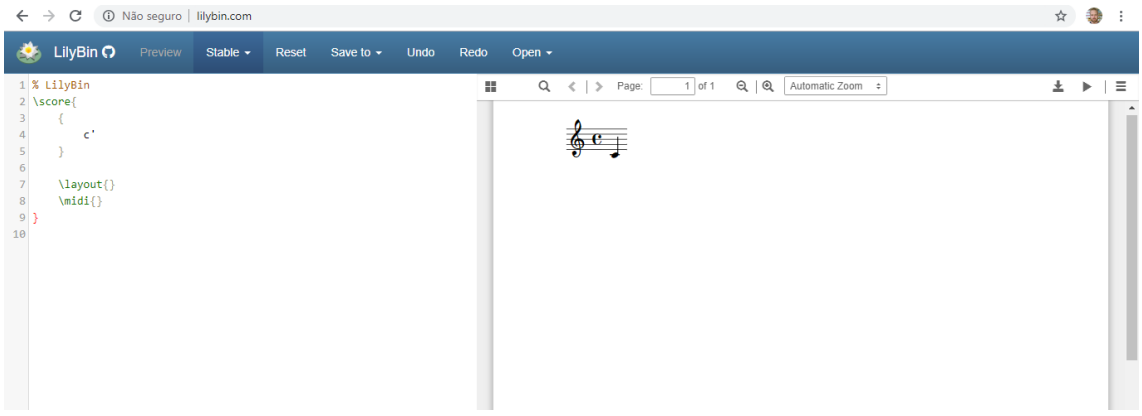


Figura 10 - Interface Gráfica do LilyBin

5 – Aspectos da Programação

Para que um programa de computador seja executado corretamente, no ato de sua construção, ele deve respeitar um série de regras que são definidas por cada linguagem. Nesta seção serão apresentados alguns conceitos presentes na programação de computadores que são inerentes a qualquer linguagem de programação.

A utilização do LilyPond se deu devido a sintaxe ser familiar à linguagem musical. Já a utilização da ferramenta Frescobaldi se dá pelo fato de funcionalidades que estão presentes no programa ajudarem no desenvolvimento de código.

5.1 Sintaxe

A sintaxe de programação é o conjunto de normas que regulam e coordenam as diferentes variáveis e sua associação. Na linguagem, a sintaxe é o conjunto de normas e leis combinatórias que estruturam a construção de orações e textos. A sintaxe se refere à inclusão do sujeito e predicado e ao papel das palavras em relação às outras. Na ciência da computação, a sintaxe compreende um conceito equivalente.

Para a informática, a sintaxe é um conjunto de normas que regem o trabalho e ligam as diferentes variáveis que compõem as instruções de operação. Em programação, existem três variáveis relevantes: a sintaxe, a semântica e a hierarquia.

A primeira tem a ver com uma linguagem de programação de um software ou sua aplicação pode ser entendida como uma série de caracteres em particular á combinação. A sintaxe está composta de regras que determinam se a combinação ou "string" é válida ou não, e, por conseguinte, operacional.

Dentro desta sintaxe também podem ser encontradas gramáticas e expressões regulares. Isso quer dizer que há caminhos comuns que são frequentemente utilizados por programadores na hora de trabalhar associando variáveis e caracteres. Termos comuns quando se fala sobre a sintaxe de programação são: os identificadores, as palavras reservadas, as literais ou as constantes e os símbolos especiais.

É comum em determinadas aplicações ou programas recebermos um erro classificado como "Erro de sintaxe", isto se refere a uma falha na programação deste software ou a combinação das variáveis em seu uso.

Como usuários, ao executar uma operação ou comando, provavelmente veríamos um botão, menu ou opção. No entanto, no código de programação ou "backend" do programa ou aplicação, veríamos uma sintaxe particular de associação de caracteres.

Na música a sintaxe também se faz presente, pois como em qualquer linguagem, ela traz uma série de regras que precisam ser seguidas. A melhor definição sobre a música é que ela é uma sequência de sons e silêncios organizados. Para chegarmos nesta definição propriamente dita temos que respeitar a sintaxe no que diz respeito a quatro características do som (altura, duração, timbre e intensidade).

Outro elemento musical a ser observado quanto a sintaxe é o ritmo. Ele trata da ordenação de sons no tempo segundo o pulso da música, que denominamos de tempo. Além disso temos a harmonia que é a união de notas de diferentes sons ouvidas ao mesmo tempo, formando a música.

Todas os pontos apontados acima fazem parte da sintaxe musical e podem ser traduzidos para o LilyPond.

5.2 Case-sensitive

Case-sensitive significa que caracteres em caixa alta e em caixa baixa são tratados de modo diferente. Por exemplo, as palavras sim e SIM são consideradas diferentes. A linguagem do LilyPond é um case sensível.

Nota	Entrada lilypond (caso sensível)
C	c
D	d
E	e

F	f
G	g
A	a
B	b

Quadro 1 – Notas e casos sensíveis

É possível observar que na música também temos casos sensíveis. Ao trabalharmos com cifras normalmente devemos utilizar a letra maiúscula para indicar a nota ou o acorde. Se usarmos letras minúsculas em uma partitura, provavelmente o músico não irá compreender o que está escrito. Quando queremos indicar que a nota ou o acorde é menor, devemos colocar um m minúsculo, se for maior um M maiúsculo. Desta forma também lidamos com casos sensíveis na música.

5.3 Entrada e Saída de Dados

Ao assistir a performance de um músico, estamos vendo também um processo complexo de entrada, processamento e saída de dados. Ao executar seu instrumento, o músico necessariamente precisa de informações que consideramos como entrada. Podemos considerar que a partitura são dados que servirão de base para o músico processar e executar seu instrumento. Mesmo que a uma peça seja executada sem que o músico olhe para a partitura, existem dados que serão utilizados pelo músico.

Nesta performance o músico é considerado como interface de processamento, que recebe os dados, faz o processamento dos mesmos e gera com isso uma saída que será apresentada por meio de um dispositivo de saída, que é seu instrumento musical. Essa saída tem como resultado uma sequência sonora.

Da mesma forma acontece na programação. Para que um programa apresente resultados, é necessário a inserção de dados que serão processados e tratados. Quando falamos da entrada de dados em programação, estamos nos referindo a informações inseridas ou captadas por

algum dispositivos de entrada, por exemplo, um teclado, um mouse, um sensor, etc. Estes dados serão lidos pelo programa e armazenados para posterior tratamento.

Ao utilizarmos o Frescobaldi, vamos considerar a entrada de dados como sendo a inserção de código e a saída como sendo uma partitura, um arquivo midi ou uma imagem. O Frescobaldi permite que a entrada de dados seja realizada por meio do teclado (onde o usuário insere as notas e códigos) ou através de entrada midi (onde o usuário insere os dados através de uma interface midi).

5.3.1 Entrada de dados no Frescobaldi

A entrada de dados no Frescobaldi deve respeitar algumas regras e organização, para que o programa possa interpretar corretamente o código e gerar a saída. Essas regras de são chamadas de regras de sintaxe e variam de acordo com a linguagem de programação utilizada.

A linguagem que será utilizada nesta apostila tem uma sintaxe similar à música visando facilitar a aprendizagem para músicos. Desta forma, a inserção de notas segue a mesma regra da cifra musical, onde a letra c representa a nota DÓ. Neste caso, o nome das notas deve ser escrito em minúsculo conforme é apresentado no Quadro 1.

Qualquer dado inserido no código de programação e chamado de instrução e são ações que um programa executa. Desta forma, as instruções necessariamente precisam ser delimitadas para que sejam compreendidas como tal.

A delimitação no Frescobaldi é feita através de chaves {} e significam o início e fim de uma instrução.

```
{
  c d e f
}
```

Exemplo 2 - Instrução contendo quatro notas.

Qualquer linguagem de programação requer que o desenvolvedor informe alguns dados no início do código para fornecer ao compilador as informações necessárias para a sua execução. Esses dados são chamados de arquivos de cabeçalho.

De forma semelhante acontece na música, quando temos indicações que são imprescindíveis no início da música. Em um partitura musical precisamos ter pelo menos a indicação de tonalidade e clave para que possamos iniciar a execução. Sem estas informações não é possível realizar a execução, pois não temos as referências mínimas necessárias. Podemos portanto considerar essas informações como arquivos de cabeçalho.

No Frescobaldi é necessário informar a versão do LilyPond que está sendo utilizada para que o compilador utilize o banco de dados correto. Essa informação deve ser inserida antes de qualquer outra informação no código.

```
\version        %Arquivo de cabeçalho
{               %Indicação de início de instrução
  c d e f       %Instrução
}               %Indicação do final da instrução
```

Exemplo 3 - Código com arquivo de cabeçalho

Uma prática recomendada e rotineiramente utilizada na escrita de códigos de programação é o uso de comentários. Tal pratica se faz importante em momentos em que a complexidade do código exige descrição. No Frescobaldi utilizamos o símbolo % seguido do comentário. Notamos na Figura 11 que na saída (partitura) não aparecem os comentários. Para comentar um conjunto de instruções utilizamos o comando %{ ... %}.

Já temos elementos suficientes para compilar esse código e gerar assim uma saída. O Frescobaldi tem como saída arquivos em formato *PDF, PNG, midi, PostScript, SVG.*

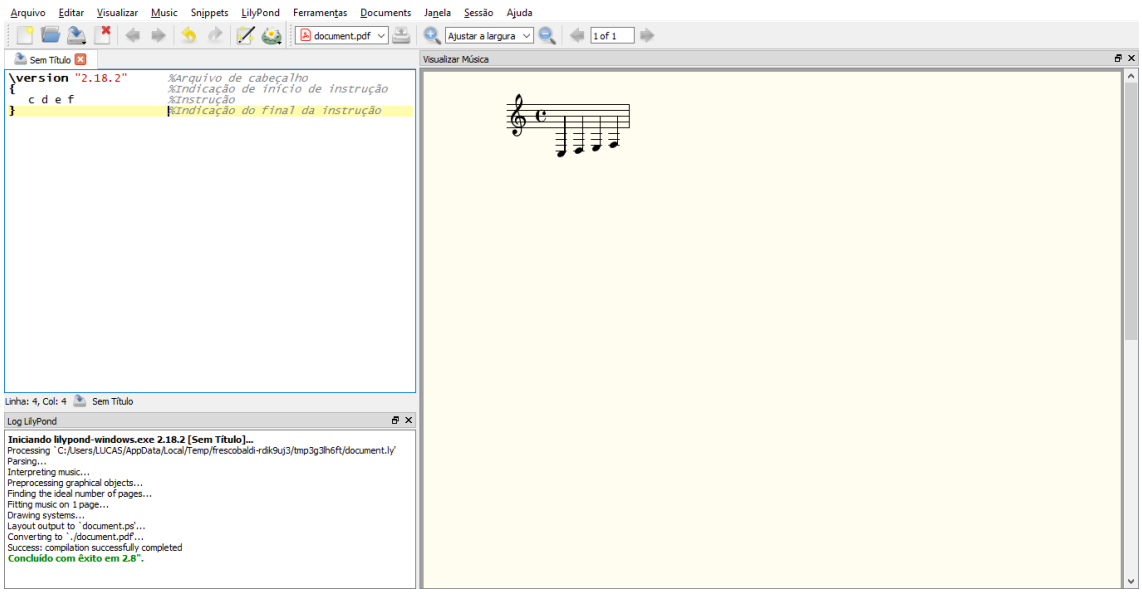


Figura 1115 - Código do Exemplo 3 compilado

Vejamos que no exemplo acima temos uma código em linguagem de programação que é compilado pelo programa e gera uma saída que é a partitura. O roteiro apresentado (elaboração do código com respeito as regras de sintaxe, compilação e saída de dados) é a base da programação.

Novamente podemos traçar um paralelo com a música, onde temos um código (partitura contendo dados) que é compilado e interpretado pelo músico que gera uma saída com algum dispositivo (instrumento musical).

Continuando com entrada de dados precisamos fornecer a duração das notas.

Seguimos a regra utilizada na música quanto a duração das notas. Segue o Quadro 2.

Duração	Entrada lilypond	
	Nota	Pausa
SEMIBREVE	1	r1
MINIMA	2	r2
SEMINIMA	4	r4
COLCHEIA	8	r8
SEMICOLCHEIA	16	r16
FUSA	32	r32
SEMIFUSA	64	r64

Quadro 2 - Quadro contendo as entradas de duração e pausa

Nomes da figuras das notas musicais	Figuras das notas musicais	Representação do valor das notas musicais.	Valor relativo das notas musicais
Semibreve			1
Mínima			2
Semínima			4
Colcheia			8
Semicolcheia			16
Fusa			32
Semifusa			64

Figura 12 - Nome e duração das figuras musicais

Para inserção de ornamentos, dinâmicas e barras de compasso, o Frescobaldi oferece o *Quick Insert (Win + alt + i)*, que é uma barra de ferramentas onde podemos inserir de forma rápida alguns códigos. Notamos também que outras funcionalidades fazem parte do software como coloração de sintaxe, numeração de linhas, recolha de bloco de instruções, criação e personalização de atalhos, janela de *debugging* onde são mostrados possíveis erros, entre outros.

A inserção de acidentes (Bemol e Sustenido) acontece com escrita da inicial do acidente após a nota conforme mostra o Exemplo 4. A definição de linguagem no início do código é que indica quais as letras que devem ser utilizadas.

```
\version "2.18.2"                    %Arquivo de cabeçalho
\language "english"
{                                    %Indicação de início de instrução
  cf\< ds e-> f\!\ff\bar "|."        %Instrução
}                                    %Indicação do final da instrução
```

Exemplo 4 - Código contendo indicações de dinâmica, acidentes, articulação e barras de compasso

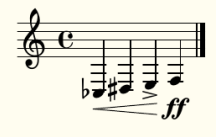


Figura 13 - Saída gerada pelo Exemplo 4

5.4 Palavra Chave

Em programação de computadores, uma palavra-chave*”* é uma palavra ou identificador que tem um significado especial para a linguagem de programação. O significado das palavras chave varia largamente de uma linguagem de programação para outra.

Na música existem símbolos, palavras ou comandos que se assemelham ao significado da palavras chave. A definição de andamento como “*andante, allegro, majestoso”* podem sem considerados palavras chave.

No LilyPond, a palavra-chave significa uma instrução ou comando que executa determinada ação. Ao utilizarmos uma palavra chave no LilyPond, ela deve necessariamente ser precedida de um caracter especial, que é a contra barra “\’’. A utilização deste caracter indica ao compilador que ele é seguido por alguma expressão importante e que tem alguma função definida em biblioteca.

No Exemplo 4, a palavra-chave é apresentada na segunda linha de instruções. *\language* significa que o compilador deve realizar a leitura de todas a palavras chave e funções em uma determinada linguagem que é definida pela usuário. A palavra *“english”* faz referência a um banco de dados que deve ser avaliado na hora da compilação.

```
\version "2.18.2"                  %Arquivo de cabeçalho
\language "english"                %Palavra-chave seguida de instrução
\relative c''                      %Palavra-chave seguida e instrução
{                                  %Indicação de início de instrução
  \time 2/4                        %Palavra-chave seguida e instrução de compasso
  \clef G                          %Palavra-chave seguida e instrução de clave
  \key g \major                    %Palavra-chave seguida e instrução de tonalidade
  cf\< ds e-> f\!\ff\bar "|."      %Instrução
```

} *%Indicação do final da instrução*

Exemplo 5 - Código contendo algumas palavras chave do LilyPond

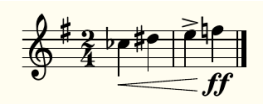


Figura 14 - Código compilado de exemplo 5

O Exemplo 5 traz algumas palavras chave que são utilizadas com frequência ao se escrever partituras utilizando o Frescobaldi.

Normalmente as palavras chave são seguidas de parâmetros para que a palavra-chave seja válida.

- *time* para definir a formula de compasso. Ela deve ser seguida do parâmetro definido pelo usuário (2/2, 2/4, 3/4, 4/4,...)
- *key* para definir a tonalidade (g *major*, f *major*,...)
- *clef* para definição de clave (G, F, C,...).

Outra palavra chave constantemente utilizada no Frescobaldi é *relative.* Ao utilizarmos esse comando, estamos habilitando o programa a escrever as notas umas próximas as outras. Para mudarmos de 8ª utilizamos apóstrofe (‘) para subir uma 8ª ou então a vírgula (,) para descer uma 8ª. A mudança de 8ª depende da quantidade de sinais utilizados, ou seja, se usarmos duas vírgulas mudamos duas 8ª.

5.5 Escopo

Em programação, o escopo é um contexto delimitante aos quais valores e expressões estão associados. Linguagens de programação têm diversos tipos de escopos. O tipo de escopo vai determinar quando as informações podem ser acessadas e quando são válidas.

Ao buscarmos respaldo na música vamos encontrar a mesma definição. Numa partitura, quando tivermos uma mudança de compasso, andamento ou tonalidade em um determinado trecho, dizemos que aquela mudança só é válida naquele momento, ou seja, seu escopo é

local. Quando é informada a clave no início do pentagrama, ele é valido na partitura toda, portanto tem um escopo global.

Da mesma forma ocorre na programação, quando definimos algum indicador dentro de alguma função, ela normalmente terá sua validade restringida aquela função, não sendo permitida a sua utilização em outro momento.

```
\version "2.18.2"
\language "english"
 \relative c'
{
  \time 4/4
  \clef G
  \key g\major
          cs4 d8 ef f g a4
          f16 g a e  a4 d f
         \time 3/2
          e2 b f
\numericTimeSignature \time 4/4
   c1
   \bar "|."
}
```

Exemplo 6 - Código contendo instrução de escopo local

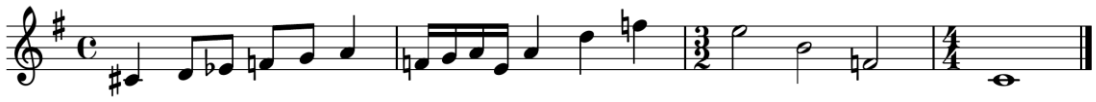


Figura 15 - Compilação do Exemplo 6

O escopo é válido para qualquer tipo de instrução, a exemplo de variáveis e funções que serão vistas posteriormente.

5.6 Estrutura de Controle

Na música existem algumas regras que são aplicadas ao fluxo de execução de uma partitura. Ao executarmos uma determinada música precisamos respeitar a ordem das notas e dos blocos dispostos na partitura. Normalmente temos na música uma introdução, parte A, parte

B, repetições, pulos de *CODA*, entre outros instruções que referem-se a ordem de execução. Cada parte é executada conforme a determinação do compositor ou arranjador através das indicações na partitura. As indicações na partitura podem ser feitas através de símbolos ou palavras que tratam justamente desse controle de fluxo.

Na programação temos algo semelhante acontecendo e chamamos isso de Estruturas de Controle e são comandos que definem qual o fluxo que o programa deve seguir ao avaliar determinada instrução.

Os tipos de estruturas de controle disponíveis diferem de linguagem para linguagem, mas podem ser cruamente caracterizados por seus efeitos. O primeiro tipo dentro de uma linguagem de programação é a continuação da execução em uma outra instrução. O segundo é a execução de um bloco de código somente se uma condição é verdadeira. O terceiro é a execução de um bloco de código enquanto uma condição seja verdadeira, ou de forma a iterar uma coleção de dados, uma estrutura de repetição. O quarto é a execução de instruções distantes entre si, em que o controle de fluxo possivelmente volte para a posição original posteriormente. O quinto é a parada do programa de computador.

Na música as estruturas de controle tem algumas funções semelhantes. Podemos observar que um ritornelo é uma estrutura de repetição, pois ele é executado enquanto a condição não é satisfeita. Comumente o ritornelo tem apenas duas casas, mas existem compositores que pedem que a primeira casa seja executada várias vezes até que a indicação na partitura seja satisfeita. Após esta condição ser satisfeita a execução passa para uma próxima instrução e com isso temos uma estrutura de continuação. Podemos também citar os pulos de *CODA, S, DA CAPO AO FINE*, e tantos outros.

Para representarmos algumas estruturas de controle no Frescobaldi vamos utilizar os conceitos de música para isso. O Exemplo 6 com a Figura 15 mostram uma estrutura de controle utilizando a palavra-chave *repeat* seguida da condição, que é apenas a execução da instrução entre chaves por duas vezes. Se quiséssemos que a instrução fosse executada mais vezes, devemos alterar o número.

\repeat volta 2 { c4 d e f }

```
c2 d
\repeat volta 2 { d4 e f g }
```

Exemplo 7 - Repetição simples

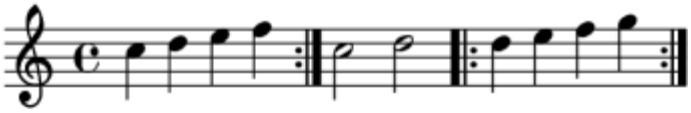


Figura 16 - Compilação do Exemplo 7

```
\repeat volta 4 { c4 d e f | }          %indicação do que deve ser repetido
\alternative {                          %comando que indica casas alternativas
  { c2 e | }                            %casa de 1ª, 2ª 3ª vez
  { f2 g | }                            %casa de 4ª vez
}
c1
```

Exemplo 8 – Repetição condicional

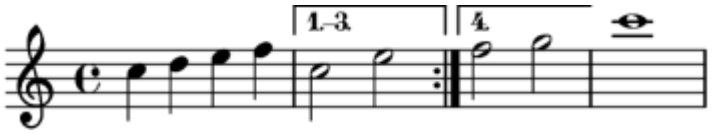


Figura 17 - Compilação do Exemplo 8

No exemplo 7 podemos visualizar uma estrutura de controle de execução de um bloco enquanto a condição for verdadeira. O código informa que demos executar o segundo compasso enquanto a condição (3 vezes) não for satisfeita. Ao executar o primeiro compasso pela 4ª vez a condição disposta no segundo compasso será analisada e se foi satisfeita o fluxo segue para o terceiro compasso sem executar o segundo.

Para indicação de desvios de seção utilizando o S ou *CODA* devemos inserir marcadores. Isso se faz necessário pois o Frescobaldi interpreta essas instruções como marcadores.

```
\version "2.18.2"
{
  \clef G
  \key g \major
  \time 4/4
```

```
 \relative c'' {
    c4 c c c
  \mark \markup { \musicglyph #"scripts.segno" }
    c4 c c c
  \mark \markup { \musicglyph #"scripts.coda" }
    c4 c c c
  \bar "||"
  \stopStaff
       s1 s1 s1
      \mark \markup {"D.S. al Coda" }
       s1 s1 s1
   \startStaff
 \mark \markup { \musicglyph #"scripts.coda" }
  \bar "||"
    c4 c c c
  \bar"|."
 }
}
```

Exemplo 9 - Código com estrutura de controle e desvios de seção.

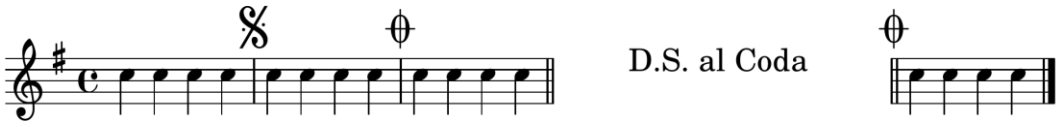


Figura 18 - Compilação do Exemplo 9

O exemplo acima apresentado é uma estrutura de controle que é rotineiramente utilizada na música para representar desvios para seções distintas.

No Frescobaldi existem duas maneiras de inserir esses símbolos, porém cada uma delas produz um resultado distinto. A utilização na janela *QUICK INSERT* insere as informações atreladas a nota e o compilador vai entender como um símbolo e não uma instrução. Na partitura não teremos nenhuma mudança visual, mas se formos exportar o *MIDI* não teremos o resultado apropriado. Desta forma, quando tratamos de fluxo de execução devemos utilizar *\mark \markup {}.*

5.7 Variáveis e Constantes

Programas de computador utilizam os recursos de hardware mais básicos para executar algoritmos. Enquanto o processador executa os cálculos, a memória é responsável por armazenar dados e servi-los ao processador. O recurso utilizado nos programas para escrever e ler dados da memória do computador é conhecido como variável, que é simplesmente um espaço na memória o qual reservamos e damos um nome. Por exemplo, podemos criar uma variável chamada "idade" para armazenar a idade de uma pessoa. Você pode imaginar uma variável como uma gaveta "etiquetada" em um armário.

Figura 19 - Variável como uma gaveta "etiquetada" em um armário.

Quando criamos uma variável em nosso programa especificamos que tipo de dados pode ser armazenado nela (dependendo da linguagem de programação). Por exemplo, a variável nome só poderia armazenar valores do tipo texto. Já a variável idade, só poderia armazenar valores do tipo número (inteiro).

Chamamos este espaço alocado na memória de variável, porque o valor armazenado neste espaço de memória pode ser alterado ao longo do tempo, ou seja, o valor ali alocado é "variável" ao longo do tempo.

Na música podemos encontrar estruturas semelhantes a variáveis. Ao lançarmos mão de uma grade de música vamos encontrar vários instrumentos dispostos nessa grade. Vamos considerar que a grade da música é o código principal de um programa e os instrumentos são as variáveis que compõem esse programa. Ao iniciar a leitura desse programa, as variáveis

são acionadas e executadas. A edição de grandes peças com vários instrumentos vai gerar grandes códigos, e consequentemente teremos dificuldade para compreender ou localizar determinadas instruções dentro de inúmeras linhas de código. Para evitar esse tipo de confusão, passaremos a utilizar variáveis, que poderão ser definidas no cabeçalho e posteriormente apenas chamadas. Variáveis nos musicais permitem quebrar expressões complexas e por isso há alguns pontos que precisam ser lembrados ao usar variáveis.

No Frescobaldi, o nome de uma variável deve conter apenas caracteres alfabéticos (sem números, sublinhados ou traços) e devem ser definidas (declaradas) antes da expressão musical principal, mas podem ser usadas muitas vezes, conforme necessário, em qualquer lugar depois de terem sido definidas.

```
\version "2.18.2"
exemplo = {c'4 e g c}
\relative c'
{
  c4 e g b | f d a e |
  \exemplo|
}
```

Exemplo 10 - Código contendo a declaração e uso de variável

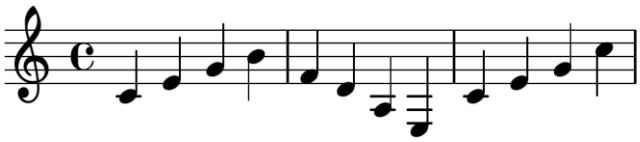


Figura 20 - Compilação do Exemplo 10

Na programação, o valor da variável pode ser alterado no decorrer do programa à depender da finalidade do software. Na seção onde será abordada a função e estrutura de grade voltaremos a falar sobre variável e como ela pode ser utilizada na definição de instrumento musical.

De forma semelhante a variável, porém sem a possibilidade de alteração de valor ou definição, a constante possui um valor fixo na execução do programa e não pode ser alterado. Um exemplo pode ser o valor da expressão matemática PI = 3,1416. Esse valor é sempre o

mesmo e quando utilizarmos ele no programa não é necessário a escrita do valor, e sim apenas o nome da constante.

Na música também temos estruturas que podemos considerar como constantes. Ao observarmos um ostinato rítmico temos uma constante e que se repete no decorrer de uma música.

A utilização de constantes no Frescobaldi segue a mesma regra das variáveis, onde o usuário define a constante utilizando as mesmas regras.

A validade desta constante ou variável depende do escopo onde ela foi definida. Na seção onde serão trabalhadas as funções vamos ver que se uma variável for definida dentro de uma determinada função, ela não pode ser chamada dentro de outra função que não possua nenhum vínculo com a função onde a variável está definida.

5.8 Funções

Em ciência da computação, mais especificamente no contexto da programação, uma sub-rotina (função, procedimento ou mesmo subprograma) consiste em uma porção de código que resolve um problema muito específico, parte de um problema maior (a aplicação final). Um exemplo seria uma função que calcula a área de um determinado espaço. Toda vez que for necessário calcular uma área no decorrer do código, basta o programador fazer referência aquela função que faz o cálculo, não sendo assim necessário reescrever todo o código.

Na música existem estruturas que tem por objetivo realizar ações semelhantes as funções de programação. No Jazz por exemplo, é normal que músicas executem improvisos dentro da música e em locais específicos parra isso. Nesse momento específico destinado ao improviso, todos os improvisadores devem respeitar a harmonia ali disposta, mesmo acontecendo improvisos e temas diferentes. Essa estrutura poderia ser chamada de função improviso, pois nela existem regras pré-definidas e o resultado desta função depende especificamente de cada músico improvisador.

O Frescobaldi possui algumas funções pré-definidas onde cada uma delas trabalha com parâmetros restritos à finalidade para que foi criada. As funções disponíveis dentro do Frescobaldi tratam de estrutura musical, como cabeçalho e layout.

5.8.1 header block

A função *\header {}* contém instruções alusivas cabeçalhos e roda pés. Nesta função ou bloco podemos inserir títulos, compositores, subtítulos, arranjadores, informações de direitos autorais, entre outros. Sempre utilizarmos essa função devemos delimitar as instruções utilizando chaves.

O Frescobaldi oferece um recurso de auto completar as palavras-chave. Para isso basta inserir a barra invertida que é utilizada ates de qualquer palavra-chave e pressionar o *ctrl + espaço* que será aberta uma lista de palavras-chave. Esse mesmo recurso pode ser utilizado dentro da função, podem sem a barra invertida.

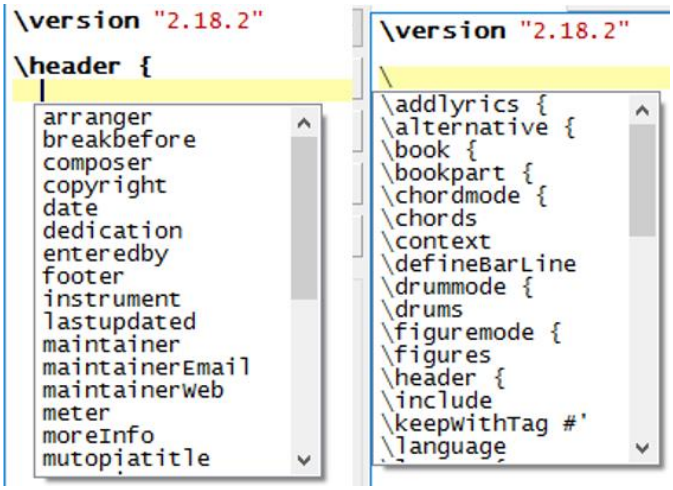


Figura 21 - Função Autocomplete para palavra-chave e instruções de bloco

O Exemplo 11 traz um código que tem a função *\header* e nela podemos observar que existem instruções relativas a parte de identificação da partitura, como título, compositor, instrumento, entre outros.

```
\version "2.18.2"
\language "english"
\header {
  title = "SUITE I."
```

```
  composer = "J. S. Bach."
  instrument = "violin"
}
```

Exemplo 11 - Código contendo a função \header

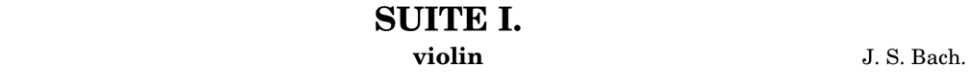


Figura 22 - Compilação do Exemplo 11

Notamos que no Exemplo 11 todas as informações que aparecem na partitura são escritas entre aspas (“ ”). Isso se faz necessário pois como exposto na seção 5.2, a linguagem que estamos utilizando é um caso sensível. Utilizando-se as aspas, o compilador interpreta essa instrução como sendo uma sequência de caracteres, ou seja, uma *string.* Na programação de computadores, uma cadeia de caracteres ou *string* é uma sequência de caracteres, geralmente utilizada para representar palavras, frases ou textos de um programa. Neste caso, o programa vai identificar esses caracteres como sendo letras, e não notas musicais.

Mesmo estando dentro de uma função que não permite a escrita de notas, precisamos utilizar as aspas, caso contrário, o compilador entendera o espaço entre uma palavra e outra como final de uma instrução e teremos um erro de sintaxe.

5.8.2 paper block

Outra função definida pelo Frescobaldi é *\paper{}* e trata de configurações relacionadas ao *layout* da partitura, como margens, quantidade de folhas, orientação da página, tipo e tamanho de papel, entre outros.

```
\paper {                              %função paper
  #(set-paper-size "a4" ‘landscape)   %tipo de papel e orientação
  top-margin = 10                     %margem superior
  bottom-margin = 10                  %margem inferior
  left-margin = 10                    %margem esquerda
  right-margin = 10                   %margem direita
  page-count = #2                     %numero de paginas
}
```

Exemplo 12 - Código contendo algumas intruções da função \paper {}

5.8.3 midi block

A criação de arquivos *midi* também será realizada através de uma função que é chamada de *\midi{}*. Nesta função podemos definir alguns parâmetros, como tempo de andamento, os instrumentos, entre outros.

```
\version "2.18.2"
\midi {
      \context {
    \Voice
    \consists "Staff_performer"
  }
  \tempo 2 = 100
 }
}
```

Exemplo 13 - Código contendo a função \midi{}

5.8.4 layout block

A personalização da partitura com a configuração espaço entre linhas, espaço entre sistemas, quantidade de sistemas por folha, quantidade de compassos por sistemas, entre outros, podem ser definidos na função *\layout {}*. A definição dos parâmetros desta função segue os mesmos critérios das demais funções.

5.8.5 score block

Esta função é uma das mais importantes utilizadas no Frescobaldi e é responsável pode receber as entradas e gerenciar as saídas. O *\score{}* pode receber várias entradas, como instrumentos diversos, variáveis, funções e gera comisso uma única saída, a partitura. No caso de uma grade, o *\score{}* é responsável por gerenciar os diversos instrumentos criados e unir todos em uma só partitura.

O Frescobaldi permite com um clique ao programador, inserir a estrutura desta função. Basta selecionar *snippets* na barra de ferramentas superior e clicar em *\scocre{...}* e a estrutura básica desta função será criada. Podemos perceber que ao criarmos a função *\score{}*, automaticamente são criadas as funções *\layout{}* e *\midi{}*. Essas funções, se forem utilizadas, devem acontecer dentro desta função, pois tem um escopo local. Se tentarmos acessar essas duas funções fora deste bloco, o compilador apontará um erro.

```
\version "2.18.2"
```

```
violin = \relative c'' {
  \clef G
  \key f \major
  {c d e f g1 \bar "|." }
}
viola = \relative c''{
  \clef G
  \key f \major
  {c4 d e f g1 \bar "|." }
}
\score {
    <<
  \new Staff { \violin }
  \new Staff { \viola }
  >>
  \layout {}
  \midi {}
}
```

Exemplo 14 - Código contendo a função \score{}

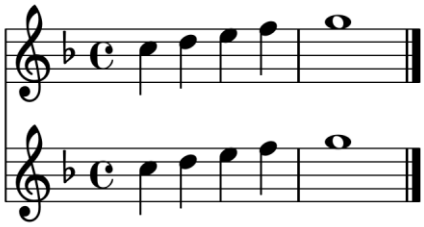


Figura 2316 - Compilação do Exemplo 14

O Exemplo 14 traz um pequeno código que contém a função *\score {}*. Nesta função, como pode ser observado, é composta de apenas uma instrução musical delimitada por chaves. Essa expressão musical, porém, pode ser apresentada de várias formas que vão da simples inserção de notas, passando pelo chamamento de variáveis, até complexas grades com inúmeros instrumentos musicais.

5.9 Include

Na programação de computadores uma prática comum é a inclusão de bibliotecas que contém funções específicas a depender do tipo e finalidade de um código. A quantidade e a finalidade de bibliotecas varia de acordo com cada linguagem.

Para compreender melhor esse tipo de comando, podemos fazer uma relação com a música e observar que esta função está presente nesse contexto também. Ao observarmos a grade de uma música, vamos ali encontrar a partitura de cada instrumento que faz parte da música. Cada instrumento individualmente pode ser considerado como sendo uma biblioteca com características específicas inerentes a ele. Portanto, podemos considerar a grade como sendo um código que incluiu várias bibliotecas que colaboram para o código.

No Frescobaldi podemos colocar isso em prática, elaborando um código central que é a grade, e incluir nela as partes individuais já editadas. A inclusão deve ser realizada utilizando-se a palavra-chave *include* mais o nome do arquivo que deseja ser incluído.

Ao realizarmos esse tipo de instrução, o programador deve ter muito cuidado, pois se o arquivo incluído conter algum erro, o compilador identificará o erro e não será gerada a saída desejada.

Recomenda-se para isso, criar no arquivo biblioteca uma variável que será chamada em tempo de execução no código principal dentro da função *score{}*.

Segue abaixo uma sequência de exemplos que vão demonstrar o uso do *include.*

```
\version "2.18.2"
\language "english"

trompeteum = \relative c'
{ cs4\mf d8[(ef\< f) g-.] a4|
  f16-. g( a e ) a4 d->-. f->-. |
  \time 3/2
  e2\ff^Intense b f |
```

```
  \clef alto
  \numericTimeSignature \time 4/4
  <c a>1\fermata \bar  "|."|
}

trompetedois = \relative c'
{\clef "bass" g2 <af c> 2|
 \tuplet 5/4 {a16 g f e f} g2 e4|
 fs2 d2 c2 |
 \numericTimeSignature <d ef>1 \fermata|
}

trompetetres = \relative c'
{ cs4\mf d8[(ef\< f) g-.] a4|
  f16-. g( a e ) a4 d->-. f->-. |
  \time 3/2
  e2\ff^Intense b f |
  \clef alto
  \numericTimeSignature \time 4/4
  <c a>1\fermata \bar  "|."|
}

trompetequatro = \relative c'
{\clef "bass" g2 <af c> 2|
 \tuplet 5/4 {a16 g f e f} g2 e4|
 fs2 d2 c2 |
 \numericTimeSignature <d ef>1 \fermata|
}
```

Exemplo 15 - Código contendo quatro variáveis

O Exemplo 15 apresenta o código de um arquivo que será utilizado para a criação de uma grade e foi chamado de *definicao.ly*. Note que neste código apenas são criadas quatro variáveis com as definições de clave. O único ponto que foi definido para que não tenho erro de sintaxe foi a linguagem, pois as notas e palavras chave utilizadas são apresentadas em inglês. Se esse código for compilado, o Frescobaldi não apresentará nenhuma saída, pois as variáveis apenas foram declaradas, e não evocadas.

O código também apresenta a sintaxe para escrita de acordes. Para isso, as notas que fazem parte do acorde deverão ser colocadas entre setas < >. O compilador entenderá que se trata de um acorde. Para fazer a ligação das hastes das notas basta escrever as notas entre [] colchetes.

A inserção de textos nas partituras pode ser realizado de duas formas utilizando-se a palavra-chave *\markup* seguida de parâmetros ({*a'8 ^ \ markup {\ itálico pizz. } g f e}*) ou então a inserção direta após a nota utilizando-se um acento circunflexo ^ seguido da *string* entre aspas ({*a'8 ^ "pizz". gfe*}).

Outro código que é apresentado faz referência a quiálteras e são usadas para dividir as notas em proporções diferentes daquela definida pela fórmula de compasso. Neste caso devemos utilizar a palavra-chave *\tuplet* seguida do parâmetro. Se for uma tercina, ou seja, três notas no lugar de uma, deverá primeiro ser informado a proporcionalidade, e depois, entre chaves, as notas que compõem esta tercina. O Exemplo 15 apresenta nas variáveis *trompetedois* e *trompetequatro* quiálteras de cinco notas no lugar de quatro. Portanto, a sintaxe utilizada é *\tuplet 5/4 {n1 n2 n3 n4 n5}* onde n significa a nota.

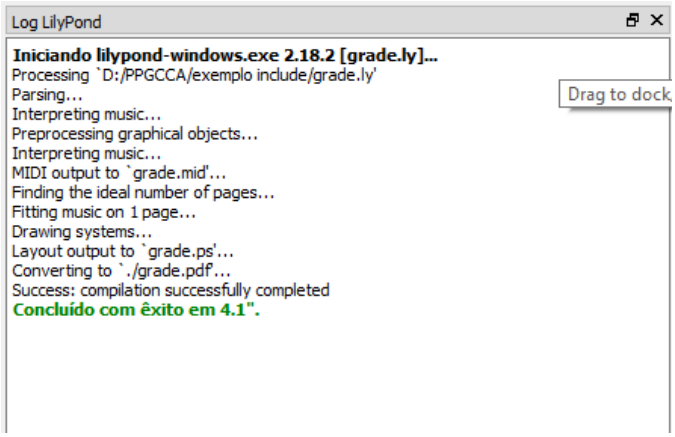


Figura 24 - Janela de Log apresentando o processo de compilação

A Figura 24 mostra que o processo de compilação foi realizado com sucesso e sem erros. Porém, o Frescobaldi não apresentou nenhuma saída e isso se deve pelo fato das variáveis que foram criadas não terem sido evocadas. Nesse exemplo foram criadas quatro variáveis que representam instrumentos diferentes. Podemos criar um arquivos com todos os instrumentos dispostos em variáveis ou então arquivos diferentes para cada instrumento. A criação de

arquivos individuais para cada instrumento facilita um possível manutenção ou localização de erros.

Para criar o arquivo principal, basta que a função principal *\score{}* seja inserida juntamente com os parâmetros. Os parâmetros desta função são os instrumentos (variáveis) que foram declaradas no arquivo *definicao.ly*. Recomenda-se que ambos os arquivos estejam salvos no mesmo diretório. Case esses arquivos estejam alocados em outros diretórios, é necessário inserir o caminho no parâmetro do *\include*.

```
\version "2.18.2"
\language "english"
\include "definicao.ly"
\header {
  title = "Vamos colocar o título aqui"
  composer = "O Alunos"
}
\score {
    <<
    \new Staff \with {midiInstrument = #"trumpet" instrumentName = "Tpt 1"} \trompeteum
    \new Staff \with {midiInstrument = #"trumpet" instrumentName = "Tpt 2"} \trompetedois
    \new Staff \with {midiInstrument = #"trumpet" instrumentName = "Tpt 3"} \trompetetres
    \new Staff \with {midiInstrument = #"trumpet" instrumentName = "Tpt 4"} \trompetequatro
    >>
    \layout {}
    \midi {}
}
```

Exemplo 16 - Código com a inclusão de arquivo externo

O Exemplo 16 montra o código para a criação de uma grade utilizando arquivos externos a partir do *\include*. Observe que o parâmetro do *\include* é o nome do arquivo entre aspas. Não precisamos fornecer o caminho do arquivo em questão pois ele está alocado no mesmo diretório que o principal.

Na função *\score{}* as variáveis criadas no arquivo *definicao.ly* são chamadas e os seu parâmetros são importados para o arquivo principal. A utilização deste recurso tem por

finalidade a legibilidade e manutebilidade do código, pois grandes peças musicais com inúmeros instrumentos gerariam códigos extensos.

Observando o Exemplo 16 vemos que temos dentro do função *\score{}* vários parâmetros específicos pertencentes a ela. Ela contém parâmetros relacionados ao tipo de partitura, como piano, percussão, simples, coral, entre outras. Escolhido o tipo de partitura devemos iniciar a inserção de dados referentes a ela, utilizando o comando *\with* seguido da instrução. No Exemplo acima temos "*\novo pentagrama simples \com {instrumento midi trompete nome do instrumento Tpt1} \trompeteum"* onde "*trompeteum"* é a evocação da variável declarada no arquivo *definicao.ly* incluída por meio do *\include.*

As informações de cabeçalho, como títulos, autores, compositores; informações sobre o *layout* de página e customização de partitura podem ser incluídas apenas no arquivo principal.

Exemplo 16

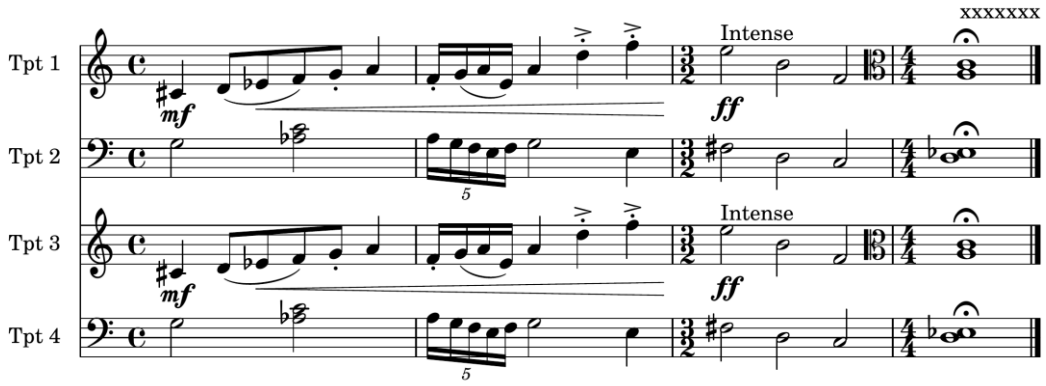


Figura 25 - Compilação do Exemplo 16

CONCEITOS COMPLEMENTARES

Nos capítulos anteriores foram apresentados alguns conceitos e termos utilizados na construção de algoritmos e programação que possuem elementos e características similares presentes na música. Esta seção continuará apresentando elementos do Frescobaldi que não foram contemplados nas seções anteriores.

6 – Funcionalidades do Frescobaldi

6.1 Setup Wizard e Funcionalidades

Para iniciar uma nova partitura de maneira mais rápida e sem inserir os comandos de cabeçalho, podemos utilizar o SETUP WIZARD do programa. Tal ferramenta é similar a todos os outros programas de editoração de partituras presentes no mercado. Nela podemos inserir nome da música, compositor, instrumentos, andamento, tonalidade, entre outros.

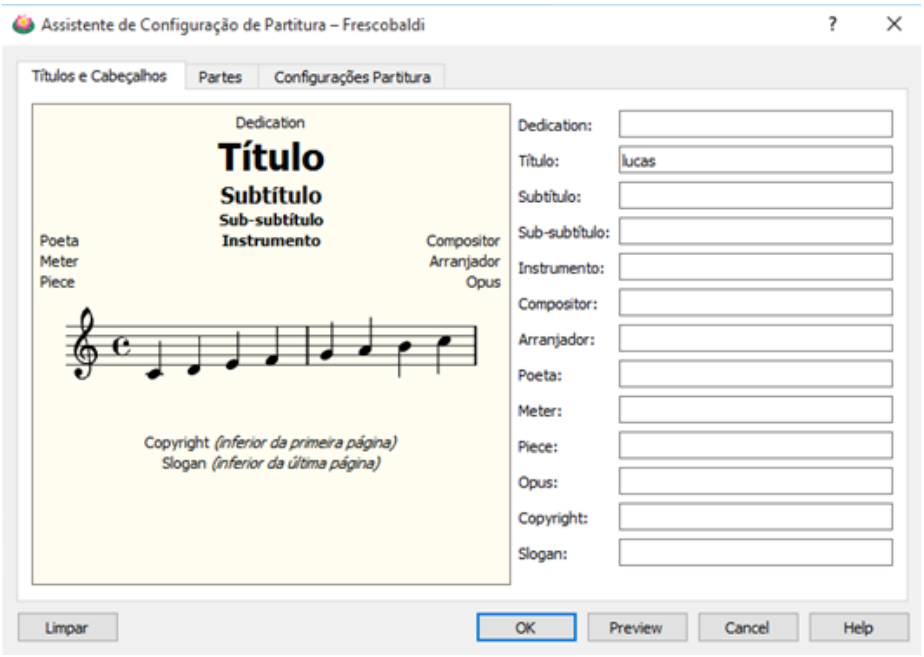


Figura 26 - Assistente de Configuração de Partitura

Também é possível criar *Templates* e modelos que poderão ser salvos para utilizar posteriormente. Esse procedimento normalmente é realizado quando o usuário cria lista de exercícios ou então métodos.

Outra funcionalidade que o Frescobaldi possui é exportar arquivos em formato svg que permite a edição da imagem. Basta abrir o arquivo exportado em algum editor de imagens ou gráficos e fazer a manipulação do arquivo. Partituras contemporâneas utilizam modelos gráficos por vezes diferentes do padrão, sendo necessário realizar ajustes em outras ferramentas.

- A editoração de grandes partituras acaba gerando grandes códigos e isso pode causar confusão quando queremos localizar algo. Para que isso não aconteça, o Frescobaldi permite o usuário habilitar uma função de recolha das instruções globais. Para que isso ocorra, precisamos habilitar a "dobradura" que está localizado no menu visualizar – *"Folding*" e vamos observar que podemos recolher e ocultar inforções de determinados blocos.
- A utilização de numeração de linhas permite localizar com agilidade códigos que contenham algum tipo de erro. Para isso basta ativar o "*Line Numbers"* no menu visualizar.
- Para executar um zoom de lupa basta aportarmos o Crtl + botão esquerdo do mouse. O nível de zoom pode ser definido nas preferências localizadas no menu editar. Na aba ferramentas podemos editar o nível do zoom bem como a resolução.
- Muitas vezes precisamos mostrar ou mandar alguma ideia sobre o que estamos fazendo para alguém. O Frescobaldi permite criar imagens rápidas de determinada área de nossa partituras. Para realizar tal operação, basta segurar o botão direito do mouse e selecionar a área desejada. Após a marcação, clicamos sobre a seleção com o botão direito e salvamos como imagem. Uma nova janela abrirá e podemos fazer alguns ajustes na imagem e salvar a mesma.
- Outra funcionalidade interessante é a possibilidade de editar notas por compasso em uma janela separada. Isso ocorre quando selecionamos a nota e clicamos com o botão esquerdo do mouse + *Edit in Place*. Neste caso podemos editar as notas por compasso.
- O manual da ferramenta também sempre está disponível para que possamos tirar qualquer dúvida sem a necessidade de abrirmos um navegador web. Para fazer isso basta abrir o navegador de documentação disponível no menu ferramentas.
- O Frescobaldi também permite reescrever um determinada nota. Se desejarmos transformar um C# em D# por exemplo, basta abrir a janela substituir localizada no menu Editar, escrever primeiramente a nota que queremos modificar e na segunda linha escrever a nota desejada. Quando compilarmos o programa veremos que substituímos a nota desejada.

- Para converter o modo relativo em modo absoluto, basta clicar em ferramentas + tom + Converter Relativo em Absoluto. Notamos no código, que o *\relative* foi excluído e as notas receberam apóstrofes ou vírgulas.
- A alteração de duração também pode ser feita com a seleção das notas e clicar sobre ferramentas + Ritmo e selecionar a duração que queremos. Ao editar durações dessa maneira, sempre verifique o deslocamento dos compassos. Procure no seu arquivo de log para se certificar de que você não tem erros.
- Transposição. Basta ir na em ferramentas + tom + *tranposition* e digitar o intervalo da transposição sempre digitando primeiro a tonalidade base e depois a tonalidade desejada.
- O Frescobaldi também permite remover articulações, dinâmicas, ornamentos, instrumentos, entre outros. Para isso basta escolher o que deseja remover no *Quick Remove* localizado no menu ferramentas.
- Outra funcionalidade é a habilitação de informações da partitura, com layout, distâncias e outro. Basta acionar *Layout Control Options* localizado no menu ferramentas.

6.2 Múltiplas Vozes

Na música é comum encontrar partituras onde várias vozes do mesmo instrumento estão escritas no mesmo pentagrama. Por exemplo, trompete 1 e 2 em uma e os trompetes 3 e 4 em outra. Para isso, basta utilizar a sintaxe <<{...}//{...}>> para escrever duas vozes no mesmo pentagrama. Deste modo as vozes são instanciadas como primeira e segunda voz e as hastes ficam separadas.

```
\new Staff <<
  \new Voice = "first"
    { \voiceOne r8 r16 g e8. f16 g8[ c,] f e16 d }
  \new Voice= "second"
    { \voiceTwo d16 c d8~ d16 b c8~ c16 b c8~ c16 b8. }
>>
```

Exemplo 17 - Código com separação de vozes nomeadas de forma personalizada

```
<<
 { r8 r16 g e8. f16 g8[ c,] f e16 d }
 \\
 { d16 c d8~ d16 b c8~ c16 b c8~ c16 b8. }
>>
```

Exemplo 18 - Código com vozes nomeadas automaticamente

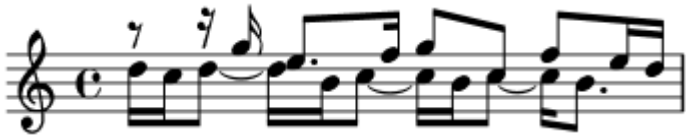


Figura 27 - Compilação do Exemplo 17 e 18 geram a mesma saída]

Para ritmos iguais e vozes diferentes utilizamos não utilizamos a barra dupla (//).

```
\new Voice <<
 { e4 f8 d e16 f g8 d4 }
 { c4 d8 b c16 d e8 b4 }
>>
```

Exemplo 19 - Código com ritmo alinhado e vozes diferentes

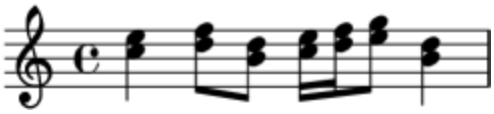


Figura 28 - Compilação do Exemplo 19

6.3 Letras e Textos

Para adicionar letra de música em um partitura devemos utilizar a palavra-chave *\addlyrics* onde a cada silaba ou palavra separada por um espaço significa uma nota conforme o Exemplo 20.

```
\relative {d"8 c16 a bs8 f es' d c4}

\addlyrics { Schad’ um das schö ne grü ne Band, }
```

Exemplo 20 - Código contendo notação musical e letras

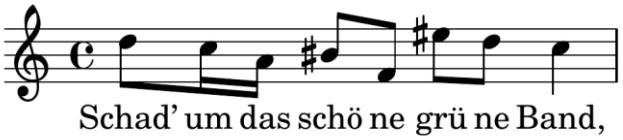
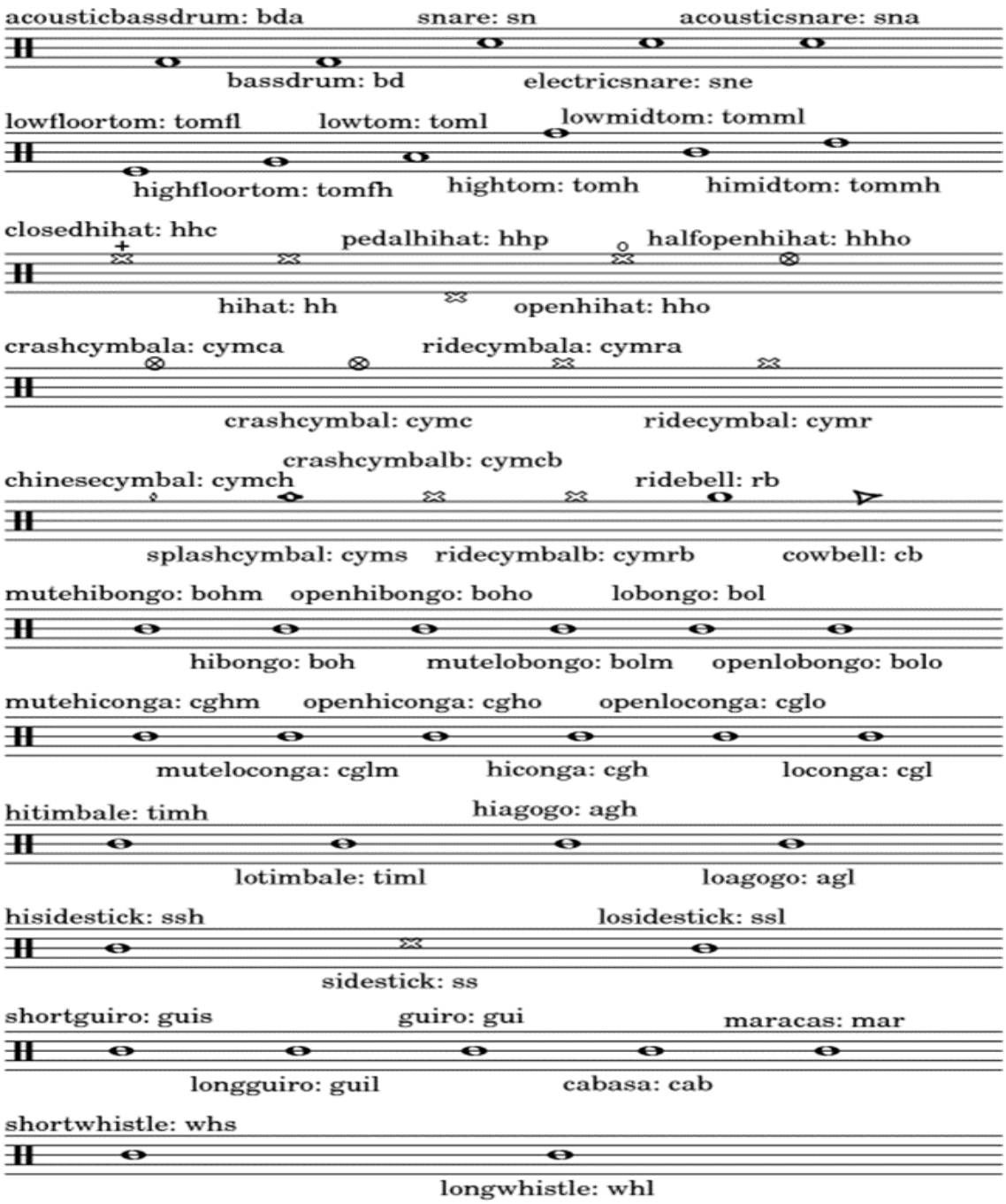


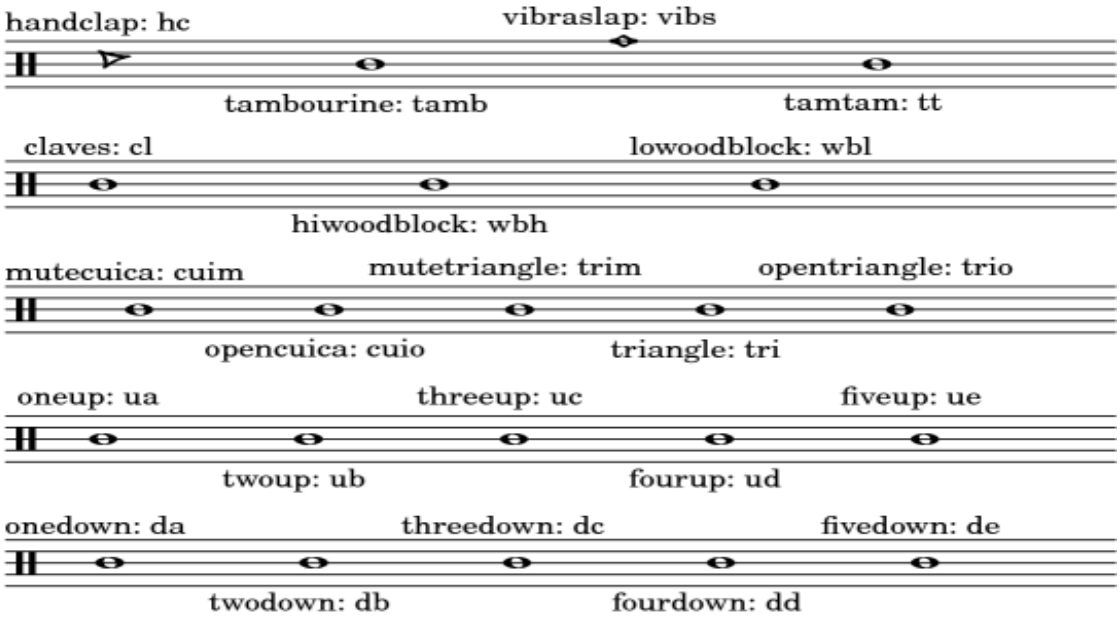
Figura 29 - Compilação do Exemplo 20

6.4 Notação de Percussão

Para inserir a notação básica de percussão vamos criar uma nova função com o nome *\drums {}*. Devemos inserir os argumentos de forma semelhante ao que fazemos com os instrumentos melódicos ou harmônicos. Entre as chaves vamos inserir as indicações de tempo, andamento além de escrever as notas.

O nome das notas deve seguir a abreviação do nome dos instrumentos conforme o Quadro 3 seguida da duração padrão que é utilizada em toda a notação musical. Para usarmos os ornamentos na caixa por exemplo, vamos escrever o nome da nota mais a duração e colocando em seguida o tipo do ornamento.


acousticbassdrum: bda
snare: sn
acousticsnare: sna
bassdrum: bd
electricsnare: sne
lowfloortom: tomfl
lowtom: toml
lowmidtom: tomml
highfloortom: tomfh
hightom: tomh
himidtom: tommh
closedhihat: hhc
pedalhihat: hhp
halfopenhihat: hhho
hihat: hh
openhihat: hho
crashcymbala: cymca
ridecymbala: cymra
crashcymbal: cymc
ridecymbal: cymr
crashcymbalb: cymcb
chinesecymbal: cymch
ridebell: rb
splashcymbal: cyms
ridecymbalb: cymrb
cowbell: cb
mutehibongo: bohm
openhibongo: boho
lobongo: bol
hibongo: boh
mutelobongo: bolm
openlobongo: bolo
mutehiconga: cghm
openhiconga: cgho
openloconga: cglo
muteloconga: cglm
hiconga: cgh
loconga: cgl
hitimbale: timh
hiagogo: agh
lotimbale: timl
loagogo: agl
hisidestick: ssh
losidestick: ssl
sidestick: ss
shortguiro: guis
guiro: gui
maracas: mar
longguiro: guil
cabasa: cab
shortwhistle: whs
longwhistle: whl



Quadro 3 - Quadro com notação de percussão e abreviação

```
\drums {
  hihat4 hh bassdrum bd
}
```

Exemplo 21 - Código contendo a função \drums{ } e parametros

Figura 30 - Compilação do Exemplo 21

Para a criação de rolos de caixa, a nota deve ser seguida de dois pontos (:) + a quantidade de notas + o til (~). Vejamos o Exemplo 22.

```
\drums {
  \time 2/4
  sn16 sn8 sn16 sn8 sn8:32 ~
  sn8 sn8 sn4:32 ~
  sn4 sn8 sn16 sn16
  sn4 r4
}
```

Exemplo 22 - Código apresentando o rolo de caixa

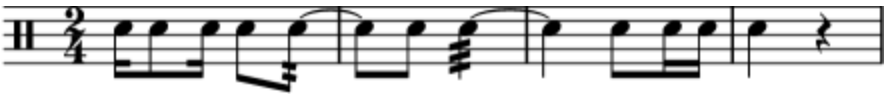


Figura 31 - Compilação do Exemplo 22

Para transcrever exercícios de caixa podemos inserir sobre a nota o lado que deve ser executado. Devemos adicionar após a nota o acento circunflexo (^) seguido do texto entre aspas. Observe o Exemplo 23

```
\drums {
  \repeat unfold 2 {
    sn16^"L" sn^"R" sn^"L" sn^"L" sn^"R" sn^"L" sn^"R" sn^"R"
    \stemUp
    sn16_"L" sn_"R" sn_"L" sn_"L" sn_"R" sn_"L" sn_"R" sn_"R"
  }
}
```

Exemplo 23 - Código com a sinalização para exercícios L e R

Figura 3217 - Compilação do Exemplo 23

Para escrever uma partitura contendo vários instrumentos de percussão podemos utilizar a escrita através de variáveis (up, down) e depois chamar elas simultaneamente dentro do *\new DrumStaff* <<>> conforme o Exemplo 24 ou escrever eles simultaneamente utilizando <<{...}//{...}>> conforme o Exemplo 25.

```
up = \drummode { crashcymbal4 hihat8 halfopenhihat hh hh hh openhihat }
down = \drummode { bassdrum4 snare8 bd r bd sn4 }
\new DrumStaff <<
  \new DrumVoice { \voiceOne \up }
  \new DrumVoice { \voiceTwo \down }
>>
```

Exemplo 24 - Código com instrumentos de percussão inseridos como variáveis

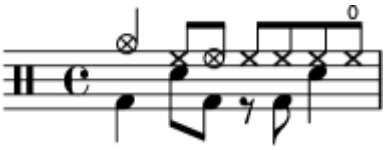


Figura 33 - Compilação do Exemplo 24

```
\new DrumStaff <<
  \drummode {
    bd4 sn4 bd4 sn4
    << {
      \repeat unfold 16 hh16
    } \\ {
      bd4 sn4 bd4 sn4
    } >>
  }
>>
```

Exemplo 25 - Código com instrumentos de percussão inseridos simultaneamente.

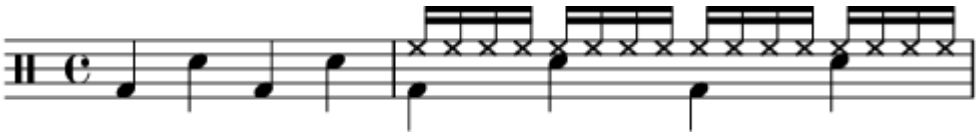


Figura 34 - Compilação do Exemplo 25