UNIVERSIDADE ESTADUAL DE GOIÁS
CÂMPUS ANÁPOLIS DE CIÊNCIAS EXATAS E TECNOLÓGICAS HENRIQUE SANTILLO
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

DANILO NOGUEIRA DA SILVA

LÓGICA DE PROGRAMAÇÃO: DIFICULDADES DE ENSINO-APRENDIZAGEM, MÉTODOS E FERRAMENTAS COMPUTACIONAIS

Anápolis
Dezembro, 2019

UNIVERSIDADE ESTADUAL DE GOIÁS
CÂMPUS ANÁPOLIS DE CIÊNCIAS EXATAS E TECNOLÓGICAS HENRIQUE SANTILLO
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

DANILO NOGUEIRA DA SILVA

LÓGICA DE PROGRAMAÇÃO: DIFICULDADES DE ENSINO-APRENDIZAGEM, MÉTODOS E FERRAMENTAS COMPUTACIONAIS

Trabalho de Curso apresentado ao Departamento de Sistemas de Informação do Campus Anápolis de Ciências Exatas e Tecnológicas Henrique Santillo da Universidade Estadual de Goiás, como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: Prof. Ms. Joilson dos Reis Brito

Anápolis
Dezembro, 2019

Câmpus Anápolis de Ciências Exatas e Tecnológicas Henrique Santillo

Universidade Estadual de Goiás

ESTADO DE GOIÁS

ATA DE DEFESA DE TRABALHO DE CURSO

Título do Trabalho: Lógica de programação: Dificuldades de ensino-aprendizagem, métodos e ferramentas computacionais
Autor: Danilo Nogueira da Silva
Orientador: M.e Joilson dos Reis Brito

O autor apresentou em sessão pública o Trabalho de Curso, às 19 horas, de 19 de novembro de 2019, terça-feira, no curso de Sistemas de Informação do Câmpus Anápolis de Ciências Exatas e Tecnológicas Henrique Santillo da Universidade Estadual de Goiás. Em seguida, cada membro da Banca Examinadora fez suas considerações e arguições. A Banca examinadora, em reunião fechada, realizou a avaliação do trabalho e da apresentação, proclamando o seguinte resultado:

(X) aprovado
( ) aprovado com correções
( ) reprovado.

A presidência encerrou a sessão pública de defesa do Trabalho de Curso.

Anápolis, 19 de novembro de 2019.

M.e Joilson dos Reis Brito
Orientador / Presidente da Banca Examinadora

M.ª Noeli Antônia Pimentel Vaz
Avaliadora

FICHA CATALOGRÁFICA

SILVA, Danilo Nogueira da.
Lógica de programação: dificuldades de ensino-aprendizagem, métodos e ferramentas computacionais / Orientador: Joilson dos Reis Brito – Anápolis, 2019, 101 p.

Trabalho de Curso (Graduação, Bacharelado em Sistemas de Informação) -- Universidade Estadual de Goiás, Campus Anápolis de Ciências Exatas e Tecnológicas Henrique Santillo, Departamento de Sistemas de Informação.

1 Abordagens para ensino de algoritmos. 2 Algoritmos. 3 Ferramentas computacionais. 4 Processo de ensino-aprendizagem. 5 Programação de computadores

CESSÃO DE DIREITOS

Danilo Nogueira da Silva
Anápolis, 24 de novembro de 2019

Dedico este trabalho aos meus pais, os quais nunca mediram esforços para que eu trilhasse todo o percurso até aqui.

AGRADECIMENTOS

Em primeira instância, agradeço a Deus pelo dom da vida e pelas dádivas de sabedoria, força, luz e saúde. Somente Ele conhece, em cada minucioso detalhe, toda a trajetória percorrida para a idealização, produção e conclusão deste trabalho.

Agradeço aos meus pais, Carlos e Telma, pelo amor e apoio incondicional em todas as etapas da minha vida. Os ensinamentos por eles repassados permitiram meu desenvolvimento e amadurecimento como pessoa, estudante e profissional. Expresso também minha gratidão aos meus irmãos, Dary e Thais, pelo carinho e orientações os quais foram ofertados em todos os momentos. Aos demais familiares, agradeço o empenho de cada um, à sua maneira, pelas contribuições dadas em prol da minha formação.

Ao longo da trajetória acadêmica, todas as batalhas foram vivenciadas na companhia dos meus estimados colegas de graduação, os quais se tornaram importantes companheiros. Agradeço a cada colega do curso de Sistemas de Informação pelos momentos vivenciados, nos quais, entre erros e acertos, construímos juntos o conhecimento. Dentre estes, dedico agradecimento pessoal à Eloisa Victoria Soares Lima, a qual foi minha colega de turma e hoje é, há pouco mais de 04 anos, minha primeira e eterna namorada. Sou extremamente grato pelo amor, companheirismo, paciência – extrema, diga-se de passagem – e auxílio a cada novo passo trilhado.

Não menos importante, manifesto gratidão por cada professor do curso de Sistemas de Informação da UEG CCET. Cada membro do corpo docente teve importância singular no meu aprendizado. Em especial, agradeço ao meu professor e orientador Joilson dos Reis Brito pela amizade, orientação, histórias e aprendizados que transcendem a esfera computacional; e à professora Elisabete Tomomi Kowata, por me apresentar, direcionar e fascinar pelo universo da pesquisa científica.

Finalmente, sou grato à sociedade a qual contribui diária e diretamente para a manutenção e fortalecimento do ensino público e de qualidade. Espero, a partir de mais esta etapa, somar forças para proporcionar a outros estudantes – atuais e futuros – novas e melhores oportunidades provenientes a partir da educação.

RESUMO

O processo de ensino-aprendizagem de algoritmos e programação de computadores é marcado pelo contato inicial com os conceitos introdutórios de algoritmos e programação. Este primeiro contato, em geral, é caracterizado pela manifestação de dificuldades, seja pelos alunos quanto à compreensão dos conceitos, seja pelos professores quanto à utilização de recursos para a transmissão dos conceitos. Mediante este contexto, é necessário conhecer o perfil dos alunos da graduação em Computação, quais são as dificuldades enfrentadas por estudantes e professores e, a partir daquelas, identificar alternativas de abordagens e ferramentas de apoio. O objetivo deste trabalho é expor a disponibilidade de métodos e/ou abordagens e ferramentas computacionais de apoio ao processo de ensino-aprendizagem, a fim de aprimorar a transmissão e compreensão dos elementos introdutórios. Para alcançar este objetivo, a pesquisa foi organizada em duas etapas: levantamento bibliográfico e coleta de dados. A primeira ocorreu mediante revisão bibliográfica e investigação, catalogação e análise de artigos, livros, trabalhos de conclusão de curso, notícias e afins. A segunda foi marcada pelo levantamento de informações com professores do ensino superior em Computação, os quais possuem experiência prévia e/ou atual com disciplinas introdutórias de algoritmos e programação. A partir desta investigação complementar, os dados obtidos foram analisados face ao referencial teórico trabalhado, de modo a evidenciar, em ambiente universitário, a ocorrência das dificuldades e as abordagens e recursos utilizados pelos professores. Os dados permitiram verificar, sobretudo, que os professores conhecem as dificuldades vivenciadas pelos estudantes, fazem uso de uma abordagem tradicional de ensino e, em geral, desconhecem recursos – métodos, abordagens e/ou ferramentas – para aprimorar o ensino. Desta forma, a conclusão deste estudo visa promover reflexão acerca do atual panorama do ensino de algoritmos e programação de computadores a respeito das dificuldades manifestadas e recursos disponíveis. A partir do conhecimento destes elementos, pode ser possível aprimorar o processo de ensino-aprendizagem, assim como estimular a investigação de outros recursos e verificar o real impacto proveniente da adoção, em ambiente universitário, de alguma abordagem não tradicional e/ou ferramenta computacional.

Palavras-chave: Abordagens para ensino de algoritmos. Algoritmos. Ferramentas computacionais. Processo de ensino-aprendizagem. Programação de computadores.

ABSTRACT

The teaching-learning process of algorithms and computer programming is marked by the initial contact with the introductory concepts of algorithms and programming. This first contact, in general, is characterized by the manifestation of difficulties, either by students regarding the understanding of concepts, or by teachers regarding the use of resources for the transmission of concepts. In this context, it is necessary to know the profile of undergraduate students in Computing, which are the difficulties faced by students and teachers and, based on these, identify alternative approaches and support tools. The objective of this work is to expose the availability of methods and/or computational approaches and tools to support the teaching-learning process, in order to improve the transmission and understanding of introductory elements. To achieve this objective, the research was organized in two stages: bibliographic survey and data collection. The first one occurred through bibliographic review and investigation, cataloguing and analysis of articles, books, final papers, news and similars. The second was marked by the gathering of information with teachers of higher education in Computing, who have previous and/or current experience with introductory subjects of algorithms and programming. From this complementary research, the data obtained were analysed in the light of the theoretical background worked, in order to highlight, in a university environment, the occurrence of difficulties and the approaches and resources used by teachers. The data allowed to verify, above all, that the teachers are aware of the difficulties experienced by the students, make use of a traditional teaching approach and, in general, do not know resources - methods, approaches and/or tools - to improve teaching. Thus, the conclusion of this study aims to promote reflection on the current panorama of teaching algorithms and computer programming regarding the difficulties manifested and available resources. From the knowledge of these elements, it may be possible to improve the teaching-learning process, as well as stimulate the investigation of other resources and verify the real impact arising from the adoption, in university environment, of some non-traditional approach and/or computational tool.

Keywords: Approaches to teaching algorithms. Algorithms. Computational tools. Teaching-learning process. Computer programming.

LISTA DE ILUSTRAÇÕES

Figura 01 – Etapas de um algoritmo....20
Figura 02 – Esquema de funcionamento do Coding Dojo....32
Figura 03 – Fases da Oficina de Lógica de Programação....33
Figura 04 – Modelo de processo da Cenarização de Conteúdos....35
Figura 05 – Sugestão de composição de níveis para o ensino de algoritmos e programação....36
Figura 06 – Aplicação da correlação linguagem de programação – português estruturado....38
Figura 07 – Realização de jogos na primeira etapa do Pensar para Programar....39
Figura 08 – Representação de "O mais leve e o mais pesado" e "Seguindo instruções"....42
Figura 09 – Interface do CodeBench....46
Figura 10 – Interface da descrição de problema no URI Online Judge....48
Figura 11 – Interface do ambiente de escrita e submissão de código no URI Online Judge....48
Figura 12 – Interface de escrita e submissão de código no The Huxley....50
Figura 13 – Interface do SuperLogo....52
Figura 14 – Interface do RoboMind....54
Figura 15 – Interface do Scratch 3.0....56
Figura 16 – Interface do Portugol Studio....59
Figura 17 – Interface do VisuAlg 3.0....61
Figura 18 – Dificuldades de aplicação relatadas pelos respondentes....81

LISTA DE GRÁFICOS

Gráfico 01 – IES dos respondentes 68
Gráfico 02 – Quantidade de turmas lecionadas pelos respondentes 69
Gráfico 03 – Abordagem introdutória adotada pelos respondentes 70
Gráfico 04 – Estratégia de ensino adotada em sala de aula 71
Gráfico 05 – Estratégia de ensino adotada em laboratório de informática 72
Gráfico 06 – Dificuldades dos alunos percebidas pelos respondentes 73
Gráfico 07 – Uso de abordagem tradicional de ensino pelos respondentes 76
Gráfico 08 – Abordagens não tradicionais aplicadas pelos respondentes 77
Gráfico 09 – Ferramentas computacionais de apoio utilizadas pelos respondentes 79
Gráfico 10 – Relevância de ferramentas voltadas ao ensino de conceitos individualizados 82

LISTA DE QUADROS

Quadro 01 – Conceitos introdutórios de algoritmos e programação de computadores.....16
Quadro 02 – Habilidades necessárias ao desenvolvimento do pensamento computacional e construção de algoritmos.....21
Quadro 03 – Hipóteses das causas de dificuldades no aprendizado de algoritmos e programação 24
Quadro 04 – Categorias de dificuldades no aprendizado de algoritmos e programação.....26
Quadro 05 – Atividades de computação desplugada.....40
Quadro 06 – Categorias de softwares educacionais 43
Quadro 07 – Protocolo de pesquisa.....65
Quadro 08 – Links das ferramentas e plataformas 97
Quadro 09 – Aplicativos móveis para o ensino-aprendizagem de algoritmos e programação.99

LISTA DE ABREVIATURAS E SIGLAS

Siglas	**Descrição**
AVA	Ambiente Virtual de Aprendizagem
ABNT	Associação Brasileira de Normas Técnicas
BNCC	Base Nacional Comum Curricular
CCET	Campus de Ciências Exatas e Tecnológicas
CEIE	Comissão Especial de Informática na Educação
CNE	Conselho Nacional de Educação
IBGE	Instituto Brasileiro de Geografia e Estatística
IDE	Ambiente de Desenvolvimento Integrado
IES	Instituição de Ensino Superior
MEC	Ministério da Educação
SBC	Sociedade Brasileira de Computação
SPE	Sistema Personalizado de Ensino
SQL	Linguagem de Consulta Estruturada
TC	Trabalho de Curso
TIC	Tecnologia da Informação e Comunicação
XP	Programação Extrema

SUMÁRIO

RESUMO 7
ABSTRACT 8
LISTA DE ILUSTRAÇÕES 9
LISTA DE GRÁFICOS 10
LISTA DE QUADROS 11
LISTA DE ABREVIATURAS E SIGLAS 12
INTRODUÇÃO 15
CAPÍTULO 1 - REFERENCIAL TEÓRICO 19
1.1 Algoritmo e pensamento computacional 19
1.2 Perfil dos alunos e o ensino da computação no Brasil 22
1.3 Dificuldades vivenciadas pelos alunos de Lógica de Programação 24
1.4 Dificuldades vivenciadas pelos professores 28
1.5 Métodos e/ou abordagens de ensino-aprendizagem de algoritmos e programação 30
1.5.1 Coding Dojo 31
1.5.2 Oficina de Lógica de Programação 33
1.5.3 Cenarização de Conteúdos 34
1.5.4 Organização do ensino sob o Sistema Personalizado de Ensino (SPE) 35
1.5.5 Correlação entre linguagem de programação e português estruturado 37
1.5.6 Pensar para Programar 38
1.5.7 Computação desplugada 40
1.6 Ferramentas computacionais de apoio ao processo de ensino-aprendizagem de algoritmos e programação 42
1.6.1 Juiz online 44
1.6.1.1 CodeBench 45
1.6.1.2 URI Online Judge 47
1.6.1.3 The Huxley 49
1.6.2 Ambiente de desenvolvimento visual 51
1.6.2.1 LOGO 51
1.6.2.2 RoboMind 53

1.6.2.3 Scratch 55
1.6.3 Compilador de Portugol 58
1.6.3.1 Portugol Studio 58
1.6.3.2 VisuAlg 60
CAPÍTULO 2 - METODOLOGIA 63
2.1 Caracterização da pesquisa 63
2.2 Instrumentos e procedimentos para a coleta dos dados 64
CAPÍTULO 3 - ANÁLISE DOS RESULTADOS 67
3.1 Informações gerais do questionário 67
3.2 Análise das respostas do questionário 68
CONCLUSÃO 84
REFERÊNCIAS 86
APÊNDICES 94
Apêndice A – Cronograma previsto e realizado 94
Apêndice B – Links para download das ferramentas e acesso às plataformas computacionais 97
Apêndice C – Aplicativos móveis para o ensino-aprendizagem de algoritmos e programação de computadores 98

INTRODUÇÃO

Em pleno século XXI, a educação permanece como um dos elementos de maior capacidade de transformação da sociedade. O progresso da civilização humana em direção ao estado atual não teria sido possível na ausência deste elemento transformador. Para Bersanette *et al* (2018), a educação deve focar na formação do ser humano em sua amplitude, e não apenas como profissional, dedicando-se ao preparo do indivíduo para lidar com incertezas, adversidades e mudanças sociais, políticas, econômicas e tecnológicas. O desenvolvimento da tecnologia avança progressivamente e, desta forma, faz-se necessário compreender o *status-quo* do processo de ensino-aprendizagem dos fundamentos de computação, como algoritmos e programação de computadores, fundamental para o desenvolvimento de novos recursos tecnológicos.

O ensino introdutório de algoritmos e lógica de programação é uma constância nos cursos superiores da área de Computação. De acordo com Santos e Costa (2006), a programação baseia-se na aplicação de princípios, formalismos e técnicas para o desenvolvimento de softwares adequadamente estruturados e confiáveis. Segundo o MEC (2016), os cursos de Ciência da Computação, Engenharia da Computação, Engenharia de Software, Sistemas de Informação – ambos bacharelados – e Licenciatura em Informática devem formar egressos os quais tenham capacidade em identificar problemas passíveis de solução algorítmica e, consequentemente, utilizar ambientes de programação para desenvolver soluções computacionais.

Em prol da formação de profissionais capacitados a aliar a teoria à prática, os cursos de graduação em Computação apresentam em sua estrutura curricular disciplinas voltadas ao ensino de algoritmos e fundamentos de programação. Para Santos e Costa (2006), a presença da programação em cursos de Computação é fundamental, pois é através desta que se supre o computador com meios para este ser útil ao usuário. Apesar das diferentes denominações – Algoritmos (I e II), Lógica de Programação (I e II), Linguagens e Técnicas de Programação (I e II), Fundamentos de Programação (I e II), dentre outras – a ementa destas disciplinas contempla tópicos fundamentais e, sobretudo, seminais ao aprendizado de programação de computadores. Estes tópicos são apresentados no Quadro 01 a seguir.

Quadro 01 – Conceitos introdutórios de algoritmos e programação de computadores

Entrada e saída de dados	Variáveis e constantes	Operadores aritméticos, lógicos e relacionais
Estruturas condicionais	Estruturas de repetição	Vetores e matrizes
Funções e procedimentos	Registros	Arquivos

Fonte: Do autor (2019).

Apesar da constante evolução tecnológica, a qual impactou no desenvolvimento de novos recursos e paradigmas computacionais, o ensino dos tópicos listados acima se faz presente no histórico educacional dos cursos bacharelado e licenciatura na área de Computação. Em outras palavras, observa-se regularidade na relação dos tópicos de algoritmos e programação de computadores ensinados nas universidades, sendo esta regularidade visível quanto ao tempo – decorrer dos anos – e espaço – distribuição geográfica das universidades. Esta recorrência quanto aos tópicos ensinados fundamenta-se na utilização dos mesmos para a construção de ferramentas computacionais desde as mais simples às mais complexas, configurando a base estrutural para a implementação de softwares.

O constante ensino de tais tópicos nas disciplinas iniciais de programação de computadores tem sido acompanhado da manifestação de dificuldades, por parte dos discentes, no que tange à compreensão e aplicação dos conceitos ensinados. As dificuldades enfrentadas pelos alunos nos períodos iniciais da graduação, sobretudo nas disciplinas introdutórias à programação, refletem-se na elevada incidência de reprovação e, em casos mais acentuados, evasão do ensino superior. De acordo com estudo desenvolvido por Silva Junior (2018), na Universidade Estadual de Goiás, estas disciplinas apresentam índice de reprovação de aproximadamente 50% e, após sucessivas reprovações –em média até três vezes -, são maiores as chances de ocorrer a evasão do curso.

Paralelamente às dificuldades vivenciadas pelos alunos, os professores de algoritmos e programação também experienciam obstáculos quanto à atividade docente. Imersos em um contexto de ensino moldado à luz da rotina “teoria-exercício”, os docentes correm o risco de não se ater adequadamente às dificuldades dos alunos, de modo que estes não conseguem traduzir a abstração de conceitos e técnicas em oportunidades reais de aplicação do conhecimento.

Frente a estas experiências nas duas esferas do processo de ensino-aprendizagem, estudos são desenvolvidos com o intuito de investigar meios de tornar este processo o mais proveitoso possível, para que alunos e professores partilhem experiências concretas e construam o conhecimento. Os meios estudados e aplicados em sala de aula estão associados tanto ao método de ensino quanto à utilização de ferramentas computacionais de apoio ao processo de ensino-aprendizagem. Marcolino e Barbosa (2015) observam uma intensificação de esforços em prol do aprimoramento de abordagens e técnicas para o ensino de disciplinas de algoritmos e programação, visto a relevância destas no contexto da graduação em Computação.

O curso de Sistemas de Informação do Campus Anápolis de Ciências Exatas e Tecnológicas (CCET) da UEG deu início, em 2019, ao desenvolvimento de softwares os quais serão aplicados nas disciplinas de Lógica de Programação I e II, com o objetivo de aprimorar a compreensão dos alunos acerca dos conceitos iniciais de programação de computadores. Estas ferramentas computacionais serão construídas a partir de Trabalhos de Conclusão de Curso (TCCs) e Projetos de Pesquisa na categoria Desenvolvimento de Software e, para que aquelas sejam relevantes e tenham êxito no processo, faz-se necessário conhecer o estado da arte de aplicações cujo foco seja o ensino dos tópicos listados acima.

Mediante este panorama, o presente trabalho buscou responder à seguinte questão de pesquisa: face às dificuldades vivenciadas pelos alunos nas disciplinas iniciais de algoritmos e programação, quais são as abordagens e ferramentas computacionais atualmente disponíveis para potencializar o ensino destas disciplinas?

A partir deste questionamento, delineou-se o objetivo geral do trabalho, a saber: expor o estado da arte quanto à disponibilidade de métodos e/ou abordagens e ferramentas computacionais para apoiar o ensino de tópicos básicos de algoritmos e programação de computadores.

O objetivo geral conduziu à definição dos objetivos específicos listados abaixo:

- identificar as principais dificuldades vivenciadas pelos alunos;

- identificar as principais dificuldades vivenciadas pelos professores;

- subsidiar a aplicação de métodos/abordagens de ensino-aprendizagem de algoritmos e programação face à abordagem tradicional;

- subsidiar futuros trabalhos de desenvolvimento de software voltados ao ensino de algoritmos e programação;

- e levantar, via pesquisa de campo, o atual panorama das dificuldades vivenciadas e das abordagens e ferramentas aplicadas pelos professores nas disciplinas introdutórias de algoritmos e programação de computadores.

Para o alcance dos objetivos mencionados acima, foi conduzida investigação segmentada em duas etapas: bibliográfica e de campo – via questionário online. Durante a condução da primeira etapa deste processo, a dificuldade de maior relevância foi a seleção e catalogação dos materiais bibliográficos encontrados. Para sanar esta dificuldade, recorreu-se à adoção de um protocolo de pesquisa e à utilização de software para organização do acervo que compõem as referências bibliográficas – ambos detalhados no Capítulo 2 deste trabalho.

Em relação ao universo do algoritmo e da programação de computadores, este trabalho contribui para instigar a reflexão entre os docentes quanto às estratégias aplicadas na condução destas disciplinas ao longo dos anos. Mediante a identificação de dificuldades dos alunos e professores e a exposição de abordagens não tradicionais e ferramentas computacionais, espera-se que este trabalho auxilie os professores na condução das aulas e na busca por demais recursos alternativos não abordados neste manuscrito.

No que concerne à divisão do trabalho em sessões, o mesmo está organizado da seguinte maneira: no Capítulo 1 é desenvolvido o referencial teórico, o qual contempla a discussão dos temas-chave desta pesquisa, a saber: algoritmo e pensamento computacional; perfil dos alunos e o ensino da Computação no Brasil; dificuldades vivenciadas pelos alunos; dificuldades vivenciadas pelos professores; métodos e/ou abordagens de ensino de algoritmos e programação; e ferramentas computacionais de apoio ao processo de ensino-aprendizagem de algoritmos e programação. No Capítulo 2 é descrita a metodologia utilizada para o desenvolvimento deste trabalho, assim como são fornecidos detalhes sobre o tipo de pesquisa – fins e meios -, os instrumentos e procedimentos utilizados, o tratamento dos dados, os resultados esperados mediante o tratamento daqueles e o protocolo de pesquisa utilizado. No Capítulo 3 são apresentados os resultados provenientes da análise dos dados, assim como considerações pertinentes a cada elemento da análise. Em seguida, é exposta a Conclusão do trabalho, a qual sinaliza os alcances obtidos ante a delimitação proposta e os objetivos do trabalho, assim como apresenta sugestões para o desenvolvimento de estudos futuros. Por fim, são disponibilizadas as Referências Bibliográficas e os apêndices produzidos durante o desenvolvimento do trabalho.

CAPÍTULO 1 - REFERENCIAL TEÓRICO

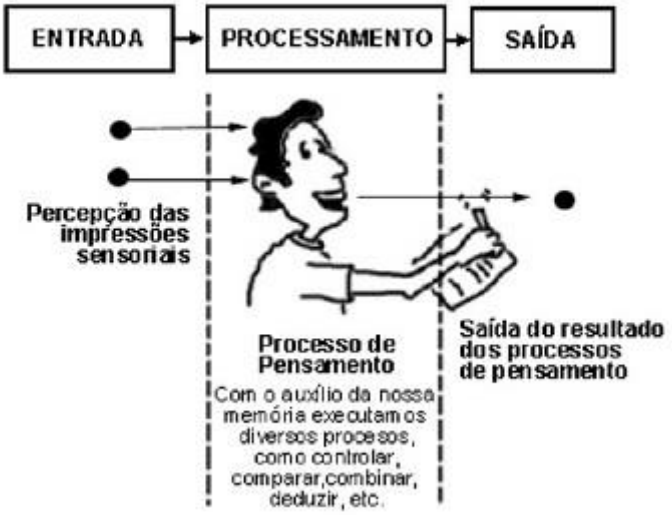
Neste capítulo, são apresentados os temas centrais desta investigação, a saber: algoritmo e pensamento computacional; perfil dos alunos e o ensino da Computação no Brasil; dificuldades vivenciadas pelos alunos; dificuldades vivenciadas pelos professores; métodos e/ou abordagens de ensino de algoritmos e programação; e ferramentas computacionais de apoio ao processo de ensino-aprendizagem de algoritmos e programação. A explanação sobre os temas é embasada em estudos e pesquisas previamente desenvolvidos, de modo que a discussão apresente validade e solidez quanto ao seu conteúdo e abordagem.

1.1 Algoritmo e pensamento computacional

O acervo bibliográfico atualmente disponível sobre algoritmo e suas aplicações não apresenta definição universalmente aceita sobre o conceito de algoritmo. Em função da orientação do tema, faz-se necessário definir o mesmo, visto ser determinante para a compreensão do estudo.

Substancialmente, o algoritmo deve ser analisado como elemento independente de qualquer viés ou paradigma tecnológico face sua existência e aplicação antecederem a gênese da programação de computadores. Lima Junior, Costa Vieira e Paula Vieira (2015) definem-no como uma sequência de ações executáveis voltadas à composição de solução para um determinado problema; Santos e Costa (2006) tratam-no como um conjunto definido de comandos responsáveis por uma sucessão finita de ações; Oliveira *et al* (2014) entendem-no como um conjunto de instruções que conduzem à uma saída desejada; e Bueno (2007, p. 47) como “Conjunto predeterminado e definido de regras e processos destinados à solução de um problema, com um número finito de etapas”. Na Figura 01 a seguir, estão evidenciadas as etapas primordiais de um algoritmo.

Figura 01 – Etapas de um algoritmo



Fonte: Magalhães (2007, editado).

Acerca do pensamento computacional, a conceituação deste assemelha-se à do algoritmo quanto à inexistência de padronização sobre seu conceito. Daltro (2011 *apud* Silva *et al*, 2014) define-o como o processo cognitivo desenvolvido pelo homem para encontrar algoritmo adequado à solução de um dado problema; Ribeiro *et al* (2013) conceituam-no como habilidade que torna possível a resolução sistemática de problemas a partir da análise das possíveis soluções e execução mediante o uso de recursos diversos; Barcelos e Silveira (2012) retratam-no como o processo cognitivo relacionado à abstração e decomposição de problemas em unidades menores e/ou menos complexas, com posterior adoção de recursos computacionais e estratégias algorítmicas para a solução.

O pensamento computacional insere-se no contexto deste trabalho como complementar à definição de algoritmo. Ribeiro *et al* (2013) distinguem a solução do problema – algoritmo – do processo de identificar a solução e desenvolvê-la – pensamento computacional.

O desenvolvimento do pensamento computacional em direção à construção do algoritmo requer do indivíduo o aprimoramento contínuo de determinadas habilidades. As principais estão listadas no Quadro 02, cada qual associada à sua definição:

Quadro 02 – Habilidades necessárias ao desenvolvimento do pensamento computacional e construção de algoritmos

Habilidade	Definição
Abstração	Habilidade de pensar o problema isolando detalhes técnicos e pensando-o como um conjunto de problemas menores e menos complexos.
Adaptação	Habilidade de adaptar soluções de problemas já conhecidos em novas soluções para problemas emergentes.
Humanização	Habilidade de propor soluções compreensíveis aos seres humanos e somente então transpô-las ao ambiente computacional.
Identificação de limites computacionais	Habilidade de identificar quais soluções são possíveis de desenvolvimento em ambiente computacional e quais não são.
Multidisciplinaridade	Habilidade de propor soluções a partir do contexto do problema e associá-la, sempre que possível, a contextos correlatos e complementares, potencializando a solução e os contextos em si.
Reconhecimento de padrões	Habilidade de reconhecer problemas recorrentes e aplicar-lhes a solução padrão, sempre que esta existir.
Seleção	Habilidade de escolher o modelo de solução mais adequado ao problema, visto que modelos distintos influenciam na complexidade, eficiência e eficácia da solução.

Fonte: Elaborado pelo autor a partir de Barcelos e Silva (2012) e Ribeiro *et al* (2013).

Realizada a conceituação de algoritmo e pensamento computacional, bem como a relação existente entre estes termos, é importante ater-se ao perfil geral dos alunos ingressantes nos cursos superiores na área de Computação, os quais são apresentados a estes conceitos nos períodos iniciais da graduação. O tópico a seguir também ilustra brevemente o panorama do ensino da Computação no Brasil, de modo a compreender o status da promoção desta vertente do conhecimento no país.

1.2 Perfil dos alunos e o ensino da computação no Brasil

Aos estudantes ingressantes em cursos superiores na área de Computação, espera-se destes o desenvolvimento, ao longo da jornada acadêmica, de conhecimentos, habilidades e atitudes direcionadas ao exercício ético, seguro e de excelência da profissão. Entretanto, este desenvolvimento é uma construção diária, a ser fomentada pelo ambiente acadêmico, face ao perfil dos ingressantes nos cursos da referida área.

Concernente às suas características estudantis, Lima Júnior, Costa Vieira e Paula Vieira (2015) traçam o perfil geral dos alunos iniciantes, cuja análise enquadra-os como provenientes do Ensino Médio regular – ausência de cursos técnicos e profissionalizantes em paralelo ao ensino –, e não naturalizados ao universo do algoritmo, do pensamento computacional e da programação. Moreira *et al* (2018) observam o perfil dos acadêmicos quando inseridos nos períodos iniciais da graduação. Para aqueles, observam-se: primeiro contato com disciplina(s) introdutória(s) de algoritmos e programação; dedicação de pouco tempo para o estudo dessas disciplinas; e baixa procura por atividades de monitoria e reforço, de modo a superarem as dificuldades e reforçarem o aprendizado.

A partir da vivência no processo de ensino-aprendizagem de algoritmos e programação, Perkins (1986 *apud* Cavalcante, 2013) categoriza os novatos em dois grupos: *stoopers* e *moovers*. O primeiro grupo diz respeito aos acadêmicos os quais se deparam com dificuldades e soluções incorretas durante a resolução de problemas e, face a esta realidade, acreditam ser incapazes de prosseguir por conta própria. O segundo grupo engloba os estudantes que, mesmo após analisarem o problema em seus múltiplos aspectos – pensamento computacional – e obterem a solução – algoritmo – prosseguem com a refatoração e busca de melhores soluções. Feita a categorização, constata-se maior quantidade de *stoopers* em relação aos *moovers* no ambiente universitário.

Ciente da necessidade de formação de cidadãos adaptados ao contexto da era digital e do atual perfil dos ingressantes nos cursos superiores de Computação, o Conselho Nacional de Educação (CNE) realizou em 2018 o Seminário Internacional sobre Computação na Educação Básica. Neste, foram discutidos como, por quem e quais seriam os conteúdos de computação a serem ensinados aos estudantes ainda na Educação Básica. A visão mais aceita pela comunidade educacional é a introdução dos conceitos básicos em disciplinas já abordadas regularmente em sala de aula, como Português, Matemática e Física, por exemplo.

Em dezembro de 2018, o CNE aprovou a nova versão da Base Nacional Comum Curricular (BNCC), contemplando as Tecnologias da Informação e Comunicação (TIC) na promoção da educação em níveis básico, fundamental e médio. Por meio desta, almeja-se aprimorar o ensino frente ao progresso tecnológico da sociedade, de modo a capacitar cidadãos como usuários e produtores de tecnologia digital, sobretudo através de recursos computacionais. Progressivamente à evolução do mercado de trabalho e das profissões, espera-se que os profissionais detenham conhecimentos suficientes para aplicarem recursos computacionais no desenvolvimento de suas atividades.

Para o Ensino Médio, o documento reitera a necessidade de transmissão da teoria e prática de conceitos de computação – neste caso, englobam-se o algoritmo e o pensamento computacional – no intuito de

> utilizar, propor e/ou implementar soluções (processos e produtos) envolvendo diferentes tecnologias, para identificar, analisar, modelar e solucionar problemas complexos em diversas áreas da vida cotidiana, explorando de forma efetiva o raciocínio lógico, o pensamento computacional, o espírito de investigação e a criatividade. (MEC, 2018, p. 475).

Ainda que a promoção do ensino de Computação nos níveis básico, fundamental e médio tenha sido oficialmente documentada, práticas concretas desta natureza são poucas e tímidas, seja pela recente formalização, seja pela precariedade de recursos viabilizadores do ensino, sobretudo na esfera pública. Assim, tornam-se evidentes dificuldades às quais os estudantes encaram no início da graduação, sobremaneira em disciplinas introdutórias ao ensino de algoritmos e programação.

1.3 Dificuldades vivenciadas pelos alunos de Lógica de Programação

A construção do conhecimento requer a parcela daquele que aprenda e possa converter o conteúdo/teoria em aplicação prática, ao mesmo tempo em que demonstre domínio e ética neste processo. No que concerne ao ensino-aprendizagem de algoritmos e programação, esta parcela é composta pelos alunos dos cursos de graduação em Computação. Segundo Carvalho, Oliveira e Gadelha (2016), cada aluno aprende a programar em um ritmo próprio, influenciado por conhecimentos prévios e motivação pessoal.

Para que as dificuldades vivenciadas pelos discentes possam ser compreendidas, é necessário investigar as possíveis causas daquelas. Costa (2013) tece hipóteses sobre possíveis causas das dificuldades dos alunos, evidenciadas no Quadro 03 a seguir.

Quadro 03 – Hipóteses das causas de dificuldades no aprendizado de algoritmos e programação

Hipótese	Ideia
Utilização de métodos inadequados de estudo	É comum ao aluno decorar o conteúdo ao invés de aplicar abstração e diálogo de ideias com outros alunos e professores.
Desmotivação	É comum ao aluno demonstrar desinteresse em função da ausência de conhecimentos prévios e pouca adaptação aos métodos tradicionais de avaliação.
Recursos insuficientes para o aprendizado extraclasse	É relativamente comum, sob condições econômicas e sociais desfavoráveis, a impossibilidade de acesso a recursos como computador próprio e Internet.
Atraso ou evasão às aulas	É relativamente comum ao aluno atrasar-se eventualmente - e em casos mais graves, rotineiramente - em função do emprego, distância residência-universidade e transporte.

Fonte: Costa (2013).

Para Raabe e Silva (2005 *apud* Medeiros, Silva e Aranha, 2013), a raiz dos problemas de aprendizado são o raciocínio lógico-matemático requerido, a apreensão dos conceitos e o ritmo de aprendizagem de cada aluno. A respeito destas causas, verifica-se amadurecimento das mesmas ao longo do percurso educacional do estudante. O quão estas dificuldades são minimizadas ao longo da vida dependem, sobretudo, do esforço individual de cada discente e da manifestação de condições - sociais, econômicas e afins – adequadas.

Enquanto alguns autores consideram o raciocínio lógico como uma das causas das dificuldades no processo de ensino-aprendizagem de programação, outros já o enxergam como uma dificuldade em si. A utilização do raciocínio lógico e a visualização completa de um problema em paralelo à assimilação dos conceitos de programação ensinados caracterizam-se como as dificuldades-chave para Marcolino e Barbosa (2015). O desenvolvimento do raciocínio lógico também é apontado por Barbosa *et al* (2011) como um dos obstáculos vivenciados pelos alunos na medida em que estes encontram dificuldades para, a partir de um problema, elaborar e estruturar a solução algorítmica.

Embora o desenvolvimento do raciocínio lógico seja comumente apontado como a dificuldade de maior predominância no processo de ensino-aprendizagem de algoritmos e programação, é importante ressaltar que existem outras barreiras as quais os alunos podem vir a encarar. Lima Junior, Costa Vieira e Paula Vieira (2015) desenvolveram pesquisa quali-quantitativa com 131 (cento e trinta e um) estudantes acerca das principais dificuldades vivenciadas no aprendizado de algoritmos e programação. Embora o raciocínio lógico tenha sido o grande destaque, com 31 % das respostas, a capacidade de abstração (28%), a leitura e a interpretação de textos (24%) e os conhecimentos matemáticos insuficientes (17%) também foram citadas.

O trabalho desenvolvido por Oliveira, Rodrigues e Queiroga (2015) constata realidade comum a boa parte dos alunos ingressantes em cursos superiores e que, no caso de graduação em Computação, possui efeitos ainda mais impactantes: o medo de errar. Os discentes não conseguem perceber o quão importante é errar e, a partir do erro, traçar novas e melhores estratégias para a resolução dos problemas. Durante a trajetória acadêmica e profissional, estes indivíduos serão constantemente confrontados com problemas e situações que exigirão inferências e experimentações para o alcance do resultado desejado. Entretanto, é importante ressaltar que o erro deve ser fruto de planejamento prévio, o qual, ocasionalmente, exigirá do aluno refinamento da estratégia aplicada à resolução de problemas. A prática de estruturar resolução errada e somente alcançar a solução por meio de sucessivas correções – prática

comum aos iniciantes na escrita de códigos em linguagem de programação – não deve ser estimulada.

A revisão sistemática conduzida por Souza, Batista e Barbosa (2016) delineou categorias de dificuldades, a partir das quais é possível identificar com maior precisão os principais desafios a serem superados durante o desenrolar das disciplinas. Algumas destas categorias, apesar de serem elaboradas a partir do ensino de algoritmos e programação, podem ser aplicadas a outras disciplinas da graduação, devidamente adaptadas. Abaixo, o Quadro 04 apresenta cinco das seis categorias aplicáveis aos alunos.

Quadro 04 – Categorias de dificuldades no aprendizado de algoritmos e programação

Categoria	Definição
Aprendizagem de conceitos	Restrições quanto à assimilação de conceitos básicos de programação de computadores, como variáveis, estruturas de repetição e funções.
Aplicação de conceitos de programação	Limitações quanto à aplicação de conceitos na construção de soluções computacionais.
Compreensão de programas	Dificuldades na leitura e compreensão de programas.
Fatoração e refatoração de programas	Limitações quanto à estruturação do programa em procedimentos, funções, módulos, camadas e afins.
Motivação	Comportamento de desânimo e/ou desinteresse na realização das atividades e trabalhos propostos.

Fonte: Souza, Batista e Barbosa (2016).

Conforme os autores responsáveis pela categorização, esta última – motivação – ocorre como consequência das anteriores. Para Santiago e Kronbauer (2016), esta motivação seria resultado da incapacidade, por parte do aluno, em construir modelos mentais adequados à compreensão do conteúdo ensinado durante as aulas. Apesar da pequena quantidade, há estudos os quais relacionam o desenvolvimento do raciocínio lógico a ramos biológicos, como a

Neurociência[1]. Neste sentido, Bastos, Adamati e Carvalho (2015 *apud* Queiroz e Rebouças, 2018) pesquisaram o comportamento do cérebro e descobriram que, quando o raciocínio lógico é pouco desenvolvido, o cérebro necessita acessar múltiplas regiões de sua estrutura, percorrendo vários caminhos para encontrar a solução do problema. Já quando o raciocínio é melhor trabalhado, apenas a região necessária à proposição da solução é acessada, de modo que o tempo de resposta à problemática é reduzido.

A parca evolução da habilidade na construção de modelos mentais pode estar associada à introdução precoce às linguagens de programação. Cabe ao professor da disciplina e corpo docente avaliarem o momento adequado para a transição da resolução de problemas ao aprendizado de uma linguagem de programação, a qual servirá de base para a resolução de futuros problemas. Segundo pesquisa conduzida por Gomes *et al* (2015), verificou-se intensificação da ocorrência de erros na solução de problemas a partir do momento em que o aluno utiliza alguma linguagem de programação. Esta constatação está associada ao aumento da complexidade necessária à proposta da resolução, visto que o discente deve ater-se então a duas realidades: a lógica do problema e a sintaxe e semântica da linguagem adotada. Para Valaski e Paraiso (2012 *apud* Raiol *et al*, 2015), o contato inicial dos estudantes com alguma linguagem de programação ocorre durante as disciplinas introdutórias, concentradas nos períodos iniciais da graduação. A desmotivação dos alunos nesta etapa, cujas consequências podem ser a reprovação e a evasão, são influenciadas pela imposição de uso de linguagens de programação com sintaxe específica e complexa.

Ambrósio *et al* (2011) conduziram pesquisa em parceria Brasil-Portugal e relacionaram o comportamento dos alunos com e sem dificuldades de aprendizado em algoritmos e programação. Por meio desta pesquisa constataram, mediante entrevista com alunos e professores, que estudantes com poucas ou nenhuma dificuldade focam na compreensão do problema em si, investigando o funcionamento da solução em paralelo ao desenvolvimento e maturação de modelos mentais, para só então migrarem para o ambiente computacional. Por outro lado, os discentes com dificuldades apresentam a tendência de ignorar o pensamento intermediário, partindo da leitura primária e superficial do problema para a codificação. Em suma, os bons alunos concentram-se em aspectos semânticos, e os alunos com dificuldades em aspectos sintáticos. Para este segundo grupo, o ato de ignorar a etapa de

[1] Ramo da ciência voltado ao estudo e desenvolvimento de conhecimentos referentes ao sistema nervoso, sobretudo da espécie humana.

processamento das informações isolada de pormenores técnicos – por exemplo, um *Integrated Development Environment* (IDE, em tradução livre, Ambiente de Desenvolvimento Integrado) - prejudica a análise das saídas esperadas.

A pesquisa desenvolvida por Gomes *et al* (2015) tratou do contato inicial dos alunos com a linguagem de programação C, a qual é utilizada com recorrência nas disciplinas introdutórias de algoritmos e programação, em função da sua tipificação e paradigma estrutural. Os dados levantados por este estudo evidenciaram que os principais erros cometidos pelos acadêmicos estão relacionados à abertura e fechamento de *tokens* – a saber, "{ }", "( )" e ";" -; uso correto de tipos de dados – variáveis e constantes –; e leitura e compreensão das mensagens de aviso e erro resultantes da compilação do programa, as quais são apresentadas em inglês e cuja ordem das informações dificulta a compreensão da mensagem, e consequentemente, a correção do programa.

Outros fatores também foram elencados pelos graduandos como responsáveis pela manifestação de dificuldades no processo de ensino-aprendizagem de algoritmos e programação. Este elencar foi resultado da pesquisa desenvolvida por Moreira *et al* (2018), a qual revelou como fatores: o pouco tempo para dedicar ao estudo, a dificuldade em interpretar as questões, o pouco conhecimento de matemática básica, o cansaço e a metodologia utilizada pelo professor.

Cientes das dificuldades dos alunos, é importante atentar-se às dificuldades vivenciadas pelos professores no ensino de algoritmos e programação.

1.4 Dificuldades vivenciadas pelos professores

Com base no tópico anterior, percebe-se que os alunos são frequentemente assolados por dificuldades durante o aprendizado de algoritmos e programação. Entretanto, este grupo não é o único a encarar obstáculos no decorrer do processo de ensino-aprendizagem. Os professores das disciplinas introdutórias de algoritmos e programação também se deparam com limitações, as quais refletem diretamente no comportamento dos alunos durante a condução das aulas. A pesquisa conduzida por Salazar, Odakura e Barvinski (2015) apontam a didática e a metodologia do professor como os aspectos mais relevantes, segundo os alunos, para promover motivação durante a aprendizagem. Para os estudantes, a motivação advém, dentre outros

fatores, de um professor com boa didática, empenhado em ensinar e capaz de apresentar problemas bem estruturados e propor exercícios desafiadores e dinâmicos.

De forma semelhante a outras disciplinas vinculadas à graduação em Computação, a condução das aulas baseia-se na lógica da teoria-exercício. Na concepção de Silva *et al* (2009), o ensino é fundamentado na apresentação do conteúdo, exibição de exemplos básicos, explicação de novo conteúdo e aumento gradual na dificuldade dos exercícios, sucessivamente. Os alunos são então submetidos às mesmas condições e desafios, como se partilhassem o mesmo nível de entendimento dos conceitos. Tem-se aqui uma relevante dificuldade vivenciada rotineiramente pelos docentes. Esta mesma dificuldade é citada por Falckembach (2006), ao discutir as causas da baixa motivação dos alunos, sendo uma daquelas o método aplicado pelos professores para promover o ensino. Submetidos a condições externas – nem sempre passíveis de controle pelo corpo docente -, os educadores encontram-se, por vezes, incapazes de ater-se ao perfil, às metas, às necessidades, às expectativas e às preferências de cada educando, em vistas de proporcionar ensino personalizado. Esta relação professor-aluno – a qual carece de tempo para fortalecimento - também é apontada por Barbosa, Fernandes e Campos (2011) como fator de influência para um baixo aproveitamento nas disciplinas introdutórias de algoritmos e programação.

O estudo desenvolvido por Franz *et al* (2014) pontua a aplicação constante de técnicas tradicionais de ensino de algoritmos e programação, tais como aulas expositivas e dialogadas, aulas em laboratório, exercícios extraclasse, trabalhos em grupo e afins. Entretanto, nem sempre este conjunto de técnicas mostra-se como o meio mais eficiente para o ensino. Oliveira, Rodrigues e Queiroga (2015) ressaltam a tímida utilização, por parte dos professores, de materiais e abordagens diferenciadas, e esta realidade incorre em monotonia e desinteresse nas aulas por parte dos estudantes.

Parafraseando Amaral *et al* (2017), o ato de lecionar algoritmos e programação baseia-se no método tradicional de exposição do conteúdo, seguido da apresentação de exemplos básicos e proposição de exercícios. Nesta organização, o professor é entendido como o detentor do conhecimento, e o aluno o receptor do mesmo. Ainda de acordo com estes autores, existem fatores complicadores atuantes sobre esta lógica, os quais potencializam as dificuldades dos discentes, a saber:

- incapacidade em prover atendimento individualizado a cada aluno – em grande parte, devido ao elevado número de acadêmicos matriculados nas disciplinas iniciais;

- indisponibilidade de tempo;

- e necessidade de cumprimento de um plano de trabalho.

A revisão sistemática conduzida por Souza, Batista e Barbosa (2016) apontou, além das cinco categorias listadas e conceituadas no tópico anterior, uma sexta, associada às dificuldades vivenciadas pelos professores de algoritmos e programação no decorrer do exercício da profissão. Esta última categoria contempla: a transmissão dos conceitos que compõem o conteúdo abordado; o desenvolvimento de materiais de apoio; e a avaliação das atividades e trabalhos desenvolvidos pelos alunos. No que tange ao repasse do conhecimento de educador para educando, verifica-se, como já identificado pelos estudos previamente citados, a prática do ensino tradicional. Em relação ao desenvolvimento de materiais de apoio e avaliação da produção do aluno, a limitação dos professores também atravessa barreiras nem sempre transponíveis por aqueles, de modo que é necessário discutir diferentes estratégias para a obtenção de êxito nestas atividades – assunto discutido em tópicos posteriores deste trabalho.

Face aos desafios vivenciados por alunos e professores no processo de ensino-aprendizagem de algoritmos e programação, faz-se necessário refinar e/ou desenvolver métodos e/ou abordagens e ferramentas computacionais de apoio a este processo. Esta ação é justificada pela ideia de Santos e Costa (2006), para os quais o estudante deve ser exposto à adequada estrutura e composição de grade curricular, com o objetivo de construir sólida base de conhecimento e munir-se de condições para o prosseguimento da vida acadêmica e profissional. Salazar, Odakura e Barvinski (2015) complementam esta ideia à medida em que a revisão e aperfeiçoamento da didática e metodologia do professor promovem maior interesse dos alunos pelas disciplinas de algoritmos e programação e, dessa forma, impactam em menores índices de reprovação e evasão.

1.5 Métodos e/ou abordagens de ensino-aprendizagem de algoritmos e programação

Observadas as dificuldades vivenciadas por alunos e professores no processo de ensino-aprendizagem de algoritmos e programação, é válida e necessária a investigação de métodos e/ou abordagens as quais possam ser aplicadas a este processo. Para Bueno (2007), método é o processo ou técnica aplicada ao ensino de algo; Ximenes (2001) conceitua abordagem como a forma ou técnica destinada ao tratamento de um assunto ou tema.

Os métodos tradicionais aplicados ao ensino revelam-se insuficientes para o aprendizado do coletivo, em face às dificuldades manifestadas, reprovação e mesmo evasão do ensino superior. O estudo e desenvolvimento de técnicas de apoio visam atenuar os problemas e criar novos espaços e possibilidades para o desenvolvimento das habilidades requeridas dos alunos (Franz *et al*, 2014).

No meio acadêmico, verifica-se recorrente utilização de métodos tradicionais de ensino de algoritmos e programação. Stadelhofer *et al* (2018) aplicaram questionário para 151 (cento e cinquenta e um) professores universitários de todas as regiões do Brasil, cuja atividade docente contemplasse, no passado ou presente, o ensino de algoritmos e programação para algum curso de nível superior. Após a obtenção das respostas e tratamento dos dados, constatou-se que 80,8% dos docentes utilizam estratégias de ensino similares, compostas por aulas expositivas, apresentação de slides, resolução de exercícios, aulas práticas e trabalhos – individuais e coletivos – em laboratório.

A pesquisa conduzida por Holanda, Coutinho e Fontes (2018) justifica a necessidade de investigar abordagens viáveis ao ensino-aprendizado de algoritmos e programação. Para estes autores, a introdução a algoritmos e à programação é complexa e constituída de muitas variáveis, e por meio de técnicas e ferramentas, busca-se minimizar os problemas enfrentados pelos estudantes.

Para Henrique e Tedesco (2007), após a compreensão das dificuldades manifestadas no ambiente de ensino-aprendizagem, é possível traçar estratégias de ensino focadas nas etapas de aquisição e de avaliação do conhecimento, sendo estes os momentos mais críticos para discentes e docentes, respectivamente.

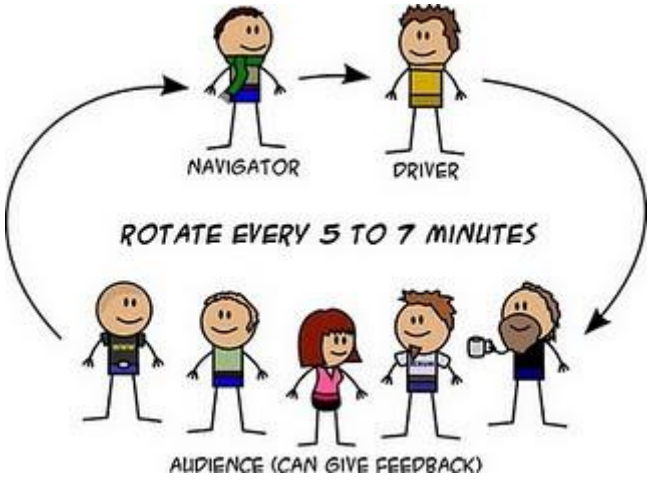
A seguir, são detalhadas abordagens não tradicionais para o ensino de algoritmos e programação de computadores em ambiente educacional, sobretudo no universitário.

1.5.1 Coding Dojo

A abordagem Coding Dojo é semelhante à programação em par proposta pela Programação Extrema (XP, do inglês *Extreme Programming*). Em essência, o *dojo master* – o líder do evento - propõe um problema a ser resolvido por uma dupla de desenvolvedores. Um dos desenvolvedores é responsável por manusear o computador, enquanto o segundo transmite ideias e indica correções e melhorias. Os demais participantes compõem plateia e, em situações

de dificuldades ou após tempo prévio determinado, substituem a dupla inicial, dando prosseguimento à resolução (Franz *et al*, 2014). Para a realização desta dinâmica, são necessários 01 (um) computador, 01 (um) projetor e espaço para abrigar os participantes - em geral, de 05 (cinco) a 20 (vinte) pessoas. Na Figura 02 abaixo, está esquematizado o funcionamento do Coding Dojo.

Figura 02 – Esquema de funcionamento do Coding Dojo



Fonte: Ferreira (2018).

O objetivo desta abordagem é aprimorar as habilidades de compreensão da situação-problema, a aplicação de técnicas de programação e a resolução de problemas. Através da programação assistida, busca-se promover a diversão e o entendimento da solução construída. Considera-se que o objetivo foi plenamente alcançado quando a resolução ocorre no tempo previsto do evento e todos os participantes entendem a solução proposta, sendo capazes de reproduzi-la por si mesmos (Organization Coding Dojo, 2016).

Dentre os pontos positivos desta iniciativa, destacam-se o aspecto colaborativo, a abertura a novas ideias e a receptividade para com estudantes e desenvolvedores com diferentes níveis de habilidades. É necessário frisar que o Coding Dojo pode causar certo desconforto em alunos e programadores com perfil tímido e/ou retraído, em função da exposição aos demais participantes (Franz *et al*, 2014).

1.5.2 Oficina de Lógica de Programação

Esta dinâmica foi desenvolvida por Pereira *et al* (2004) no Colégio Estadual Baltazar Carneiro, localizado no município de Cardoso Moreira, interior do Rio de Janeiro. Embora o foco desta oficina tenha sido alunos do Ensino Médio, é totalmente possível adaptá-la ao ensino superior. A organização dos conteúdos a serem repassados e o tempo de cada fase ficam a critério dos organizadores, e devem ser ponderados a partir do perfil dos estudantes.

Para a organização, o evento foi divido em 03 (três) fases, cada qual com conhecimentos a serem transmitidos e objetivos a serem cumpridos. O alcance destes era de fundamental importância para o avanço às próximas etapas. A primeira etapa, denominada "Iniciação Lúdica", consistiu na resolução de problemas de múltiplos domínios, a fim de estimular o raciocínio lógico isento de aspectos técnicos de programação. A segunda fase, denominada "Falando Sério", foi composta da formalização da solução através de linguagem natural e apresentação e aplicação de conceitos básicos, a saber: variáveis, conectivos lógicos e estruturas sequenciais e condicionais. Por fim, a terceira e última etapa, denominada "E Agora?", consistiu na utilização de linguagem de programação – Pascal – para a codificação das soluções, considerados os conceitos aprendidos até o momento, e a introdução de estruturas de repetição. A Figura 03 abaixo esquematiza as 03 (três) fases desta abordagem.

Figura 03 – Fases da Oficina de Lógica de Programação

Fonte: Do autor (2019).

Em todas as etapas, mesmo naquelas em que o aluno ainda não recorria à linguagem de programação para a proposição da solução, o mesmo era estimulado a utilizar o ambiente computacional, de modo a familiarizar-se com as ferramentas e recursos disponíveis.

Como resultados, obteve-se maior interesse dos estudantes pela disciplina como consequência do aspecto colaborativo e sequenciado de apresentação dos conteúdos e resolução dos exercícios.

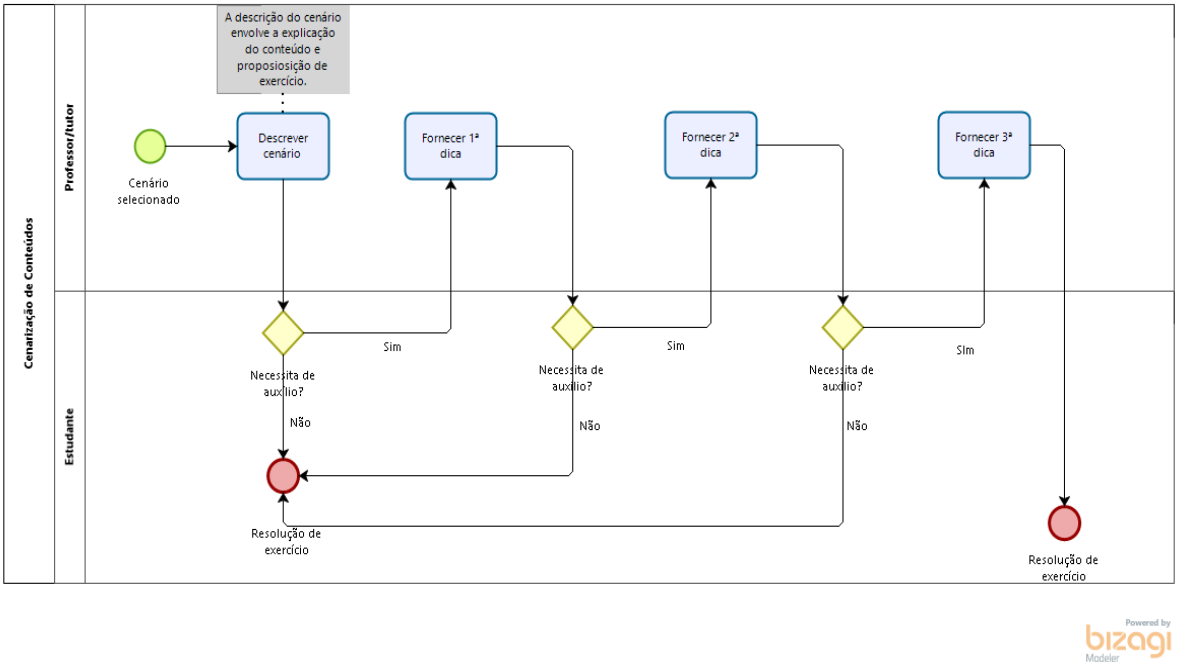
1.5.3 Cenarização de Conteúdos

Esta proposta é oriunda do estudo conduzido por Holanda, Coutinho e Fontes (2018), o qual centrou-se na investigação e desenvolvimento de metodologia de ensino em vista às dificuldades e aos métodos de aprendizado mais utilizados pelos alunos para a assimilação dos conteúdos. Constatou-se que o estudo individual e as monitorias são os métodos mais comuns para o aprendizado.

Ao traçar paralelo entre as dificuldades e os métodos de aprendizado mais recorrentes, os autores delinearam abordagem caracterizada pela composição de cenários. Para cada um destes, são definidos os conteúdos básicos a serem ensinados e a formulação de dicas a serem repassadas aos acadêmicos.

Quanto aos conteúdos, cabe ao docente definir quantos e quais serão para cada cenário. A respeito das dicas, estas são categorizadas em 03 (três) grupos: explicação de entradas disponíveis, saídas esperadas e processamento necessário; sintaxe do(s) conteúdo(s) básico(s) do cenário; e descrição narrativa da lógica do problema. Na Figura 04 abaixo, é ilustrado o modelo de processo desta abordagem.

Figura 04 – Modelo de processo da Cenarização de Conteúdos



Fonte: Adaptado de Holanda, Coutinho e Fontes (2018).

A Cenarização de Conteúdos obteve resultados satisfatórios quando de sua aplicação, visto elevar a quantidade de códigos corretos produzidos pelos estudantes. Entende-se por "correto" o código compilado e executado com sucesso, resultando na obtenção da saída esperada. Alunos novatos e repetentes na disciplina sentiram-se estimulados e foram beneficiados pela aplicação desta abordagem.

Ressalta-se, por fim, que esta abordagem pode ser adaptada às etapas iniciais de ensino de algoritmos e programação, nas quais os alunos ainda não foram introduzidos ao ambiente computacional – linguagens de programação, IDEs e afins.

1.5.4 Organização do ensino sob o Sistema Personalizado de Ensino (SPE)

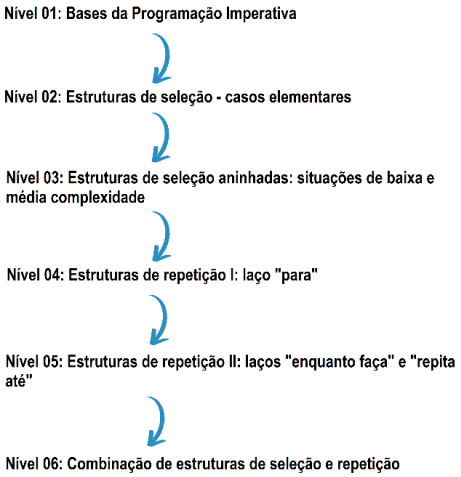
O Sistema Personalizado de Ensino (SPE) é um método de ensino desenvolvido nos Estados Unidos na década de 1960. Sob sua perspectiva, cada aluno possui ritmo de aprendizado diferenciado e, para tal, os educadores devem fazer uso dos mais variados recursos de ensino – tradicionais ou não – de modo a garantir ao estudante a adequada assimilação dos conteúdos, respeitado o ritmo de aprendizado daquele.

Apoiados por esta metodologia, Rocha *et al* (2010) propuseram a alteração na dinâmica de ensino da disciplina de Algoritmos I ofertada pelo curso Bacharelado em Sistemas de Informação da IES 14. Cientes das dificuldades vivenciadas pelos alunos quanto ao desenvolvimento do raciocínio lógico e assimilação dos conteúdos básicos e fundamentais da disciplina, assim como da organização tradicional do ensino, os pesquisadores julgaram ser mais eficiente remodelar a maneira pela qual os docentes transmitem o conhecimento. A responsabilidade dos docentes deixou de ser a mera transmissão de conteúdo para ser a de acompanhar, aprimorar, treinar e gerenciar pessoas e conteúdo. A respeito dos resultados obtidos com a aplicação desta, reduziu-se a 0% o percentual de evasão dos alunos, e a taxa de aprovação na disciplina foi elevada a 84%, ante a média de 51% quando da não estruturação da disciplina sob o SPE.

O método é válido mediante a observação dos diferentes ritmos de aprendizado e nível de conhecimento dos alunos da disciplina. A estruturação é realizada em duas frentes: organização do conteúdo e dinâmica das aulas.

A respeito do conteúdo, este é organizado em níveis – cada qual com conceitos básicos a serem assimilados, tanto na teoria quanto na prática. Para avançar de nível, o aluno deve realizar atividades individuais e coletivas, para aperfeiçoamento, e prova individual do nível, com taxa necessária à aprovação igual ou superior a 90%. Para ser aprovado na disciplina, exige-se do aluno a conclusão de ao menos 60% do total de níveis previamente estabelecidos. A Figura 05 abaixo apresenta a sugestão de composição de níveis proposta por Rocha *et al* (2010).

Figura 05 – Sugestão de composição de níveis para o ensino de algoritmos e programação



Fonte: Adaptado de Rocha *et al* (2010).

A respeito da dinâmica das aulas, o professor abandona o ensino homogêneo do conteúdo – apresentação do mesmo tópico para todos os alunos – e foca em ensinar para cada grupo de alunos (composto por alunos de mesmo nível) o assunto pertinente àquele nível.

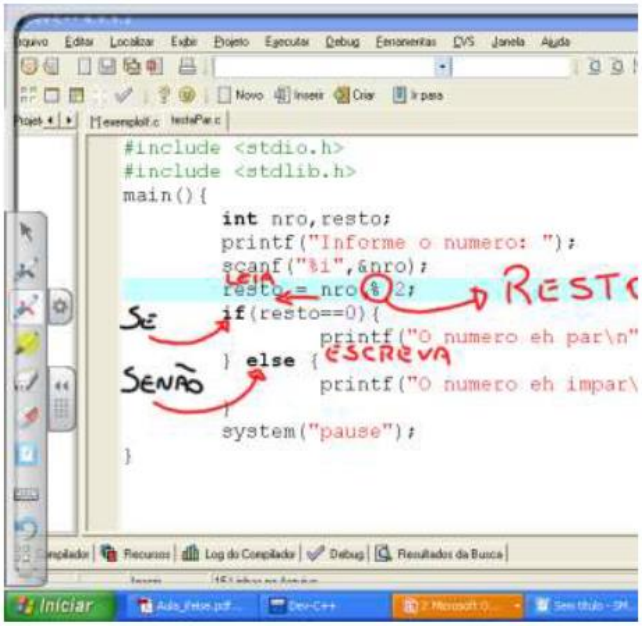
Para a viabilização desta estratégia, faz-se necessária a disponibilização dos materiais de estudo em ambiente de fácil acesso, assim como a abertura de canais de comunicação e interação dos alunos, seja para a discussão de exercícios, disponibilização de atividades e recebimento de *feedback*. Na aplicação desta metodologia na UFPA, foi utilizado o Ambiente Virtual de Aprendizagem (AVA) Moodle. Caso seja possível, o professor da disciplina deve também indicar monitor para realizar acompanhamento extraclasse dos alunos.

1.5.5 Correlação entre linguagem de programação e português estruturado

As etapas iniciais do processo de aprendizado de algoritmos e programação envolvem a apresentação dos conceitos básicos em português estruturado, isento de detalhes técnicos das linguagens de programação. Mediante esta realidade, Priesnitz Filho, Abegg e Simonetto (2012) buscaram relacionar a representação dos conceitos introdutórios, codificados em linguagem de programação, com sua representação em português estruturado. Esta abordagem foi aplicada a acadêmicos de curso da área de Computação – curso este não especificado na pesquisa – da IES 16. Após a aplicação, foi conduzido questionário com os alunos e obteve-se 100% de aceitação dos discentes quanto à facilidade de ensino proporcionada pela associação entre a representação em português estruturado e a linguagem de programação.

Para o desenvolvimento desta metodologia de ensino, foram necessários 01 (um) computador, 01 (um) projetor e 01 (uma) lousa digital interativa. Por meio destes recursos, o professor foi capaz de desenvolver e exibir os códigos implementados para os problemas propostos e, durante a codificação da solução e sua explicação, os recursos da lousa digital (lápis, marcador e borracha) possibilitaram a associação aos conceitos em português estruturado – linguagem imperativa. Realizada a associação, buscou-se facilitar a compreensão dos estudantes, os quais conseguiram entender como os conceitos previamente ensinados em linguagem imperativa são representados em uma IDE – neste caso, foi utilizada a linguagem de programação C. Abaixo, a Figura 06 exemplifica a adoção deste método.

Figura 06 – Aplicação da correlação linguagem de programação – português estruturado



Fonte: Priesnitz Filho, Abegg e Simonetto (2012).

Quando da não disponibilidade de lousa digital para recorrer a esta abordagem, o docente pode vir a utilizar outros mecanismos, como o próprio quadro-negro/quadro-branco e giz/pincel para a correlação dos conceitos. A abordagem não impõe o nível de detalhamento da correlação, de modo que esta deve ser adequada ao perfil dos alunos.

1.5.6 Pensar para Programar

O nome desta abordagem traz consigo uma importante reflexão acerca de um dos requisitos fundamentais para o desenvolvimento de habilidades de compreensão de algoritmos e de técnicas de programação de computadores: pensar. E o desenvolvimento do ato de pensar pode ser realizado através de diversas dinâmicas, inclusive aquelas isentas de recursos computacionais.

Fixados a esta ideia, Marques, Souza e Mombach (2017) estruturaram a abordagem “Pensar para Programar”, a qual foi aplicada a 29 (vinte e nove) alunos do curso Técnico em Informática da IES 03. A aplicação ocorreu entre os meses de agosto e dezembro de 2016. O objetivo geral foi o desenvolvimento do pensamento computacional nos discentes, o qual é elemento essencial à compreensão de algoritmos e técnicas de programação. Como resultados,

verificou-se significativo aprimoramento nas habilidades de resolução de problemas, trabalho colaborativo e comunicação, assim como amadurecimento do pensamento computacional.

Embora a dinâmica tenha sido desenvolvida com alunos do Ensino Técnico integrado ao Ensino Médio, aquela é totalmente passível de adoção no Ensino Superior, sobretudo nos períodos iniciais da graduação, nos quais encontram-se as disciplinas introdutórias de algoritmos e programação.

A condução desta abordagem consiste na realização de encontros semanais divididos em duas etapas: na primeira, são praticadas técnicas como Lobogames (jogos lógicos de tabuleiro executados diretamente no tabuleiro ou com os participantes simbolizando as peças), Jogos *Boole* (revistas com desafios de raciocínio lógico estruturados sob a forma de enigmas e problemas) e Problemas de Lógica (histórias e quebra-cabeças cuja resolução da problemática proposta exigem aplicação de pensamento computacional). Na segunda etapa, são resolvidos exercícios de construção de algoritmos e codificação, os quais envolvam ao menos um conceito básico de programação trabalhado na primeira etapa. A Figura 07 abaixo retrata a realização de jogos com os alunos durante a primeira etapa da abordagem.

Figura 07 – Realização de jogos na primeira etapa do Pensar para Programar

Fonte: Marques, Souza e Mombach (2017).

As atividades desenvolvidas ao longo da primeira fase foram escolhidas pelos pesquisadores em função da facilidade de aplicação das mesmas. Cabe a cada grupo de professores e/ou tutores selecionarem quais práticas serão aplicadas aos alunos a partir do perfil do corpo discente, respeitadas suas potencialidades e dificuldades.

1.5.7 Computação desplugada

A computação desplugada é uma abordagem composta por um conjunto de atividades lúdicas voltadas ao ensino de fundamentos e conceitos de computação, isentando tais atividades da utilização de recursos de hardware e software e favorecendo ambiente sem distrações e excesso de detalhes técnicos (Marques *et al*, 2017). Adicionalmente, Lima *et al* (2018) sugere que tais práticas tracem analogias ao cotidiano dos estudantes, de modo a potencializar a compreensão dos conteúdos abordados através das atividades desplugadas.

Bell, Witten e Fellows (2011) foram os responsáveis pelo desenvolvimento desta abordagem, a qual foi sendo enriquecida pelo acréscimo de novas atividades próprias ao ensino de conceitos de computação. Todo o conteúdo da obra encontra-se vinculado à licença *Creative Commons*, a qual permite, sob adequada referenciação, a reprodução e adição de novas práticas.

Dentre as dinâmicas atualmente desenvolvidas para esta abordagem, duas delas são destacadas no Quadro 05 abaixo, e a Figura 08 ilustra a aplicação das mesmas.

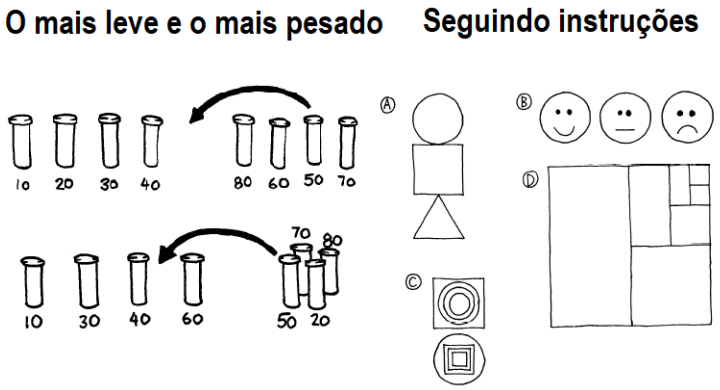
Quadro 05 – Atividades de computação desplugada

Atividade	Execução	Objetivo
O mais leve e o mais pesado	Através da utilização de recipientes preenchidos com material sólido ou líquido de fácil acesso – como areia, por exemplo -, os acadêmicos são orientados a realizar a ordenação destes recipientes à medida que os mesmos são pesados em balança. Em geral, os acadêmicos recorrem inicialmente à ordenação por seleção (*selection sort*). Este método de ordenação consiste na identificação do elemento de menor valor pertencente a	Ensinar conceitos e técnicas de ordenação de elementos, tais como *selection sort*, *quick sort*, *merge sort* e *bubble sort.*

	um conjunto de ‘n’ elementos; este é então posicionando no início da estrutura ordenada, à medida que é identificado o próximo elemento de menor valor do conjunto restante e adicionado à estrutura ordenada, até que todos os elementos sejam devidamente ordenados.	
Seguindo instruções	Um estudante reproduz o papel de programador e fornece instruções para seus colegas, os quais representam computadores. Estas instruções podem ser destinadas à realização de um desenho, modelagem de algum objeto ou trilha de percurso. Recomenda-se, progressivamente, a limitação de instruções e formalização das mesmas, de modo a evitar ambiguidades e aproximar-se das instruções recebidas por um computador.	Ensinar sobre a transmissão de comandos para um computador, e consequente ação realizada a partir do comando repassado.

Fonte: Elaborado pelo autor a partir de Bell, Witten e Fellows (2011).

Figura 08 – Representação de "O mais leve e o mais pesado" e "Seguindo instruções"



Fonte: Adaptado de Bell, Witten e Fellows (2011).

A riqueza desta abordagem encontra-se no aspecto lúdico e na variedade de conceitos passíveis de ensinamentos. Além das abordagens exemplificadas acima, há outros temas contemplados: números binários, representação de imagens, compressão de textos, detecção e correção de erros, teoria da informação, algoritmos de busca, redes de ordenação, árvores geradoras mínimas, roteamento e bloqueios em redes e autômatos de estados finitos.

Em relação às vantagens e benefícios advindos com a adoção deste método de ensino, destacam-se a possibilidade de ensinar a computação para grupos desprovidos de recursos computacionais – sobretudo hardware, software e redes -, o incentivo ao trabalho colaborativo e comunicação entre os participantes. Para os alunos dos níveis básico, fundamental e médio, verifica-se também o estímulo ao ingresso em cursos superiores da área de Computação.

1.6 Ferramentas computacionais de apoio ao processo de ensino-aprendizagem de algoritmos e programação

A adoção de ferramentas computacionais no processo de ensino-aprendizagem de algoritmos e programação é apresentada como medida complementar às abordagens tradicional e não tradicional apresentadas anteriormente. Para Lima *et al* (2018), essa complementariedade potencializa o método de ensino e torna-o mais abrangente.

Frente ao desenvolvimento tecnológico, a área de ensino-aprendizagem de algoritmos e programação deve adaptar-se àquele por meio do desenvolvimento de recursos digitais que

integrem os conceitos básicos desta área aos novos padrões de sistemas e programas computacionais (AMARAL *et al*, 2017).

Igualmente importante ao desenvolvimento de ferramentas computacionais de apoio é a reestruturação metodológica do ensino e a clara visualização dos objetivos a serem alcançados por meio do uso desses recursos digitais. Gomes e Melo (2016) salientam que, na inexistência de reestruturação metodológica e definição de objetivos, a adoção de softwares no meio educacional culmina em desperdício de potencial tecnológico e insucesso no aprendizado.

Bernasette *et al* (2018) discorrem sobre a utilização das ferramentas em ambientes educacionais, a qual deve ocorrer somente após adequada reflexão docente acerca das contribuições passíveis para o alcance do objetivo almejado. Para os autores, o objetivo primordial é criar subsídios por meio dos quais os alunos possam superar as dificuldades vivenciadas no aprendizado do conteúdo, reduzindo os índices de reprovação nas disciplinas e de evasão dos cursos.

De modo a categorizar os diferentes tipos de softwares educacionais existentes, Fiocco (2007 *apud* Nascimento, 2016) classificou as ferramentas computacionais conforme descrito no Quadro 06 a seguir. Estas categorias não são mutuamente exclusivas, sendo possível o desenvolvimento de ferramentas pertencentes a múltiplas classificações.

Quadro 06 – Categorias de softwares educacionais

Categoria	Descrição
Exercícios e práticas	Empregado para revisar conteúdo já ministrado através de dinâmica de perguntas e respostas.
Jogos	Empregado para aliar o ensino ao entretenimento.
Multimídia e Internet	Utilizado para o ensino ao empregar sons, imagens, textos e internet.
Simulação	Utilizado para simular situações cuja reprodução não é possível em ambiente real.
Tutorial	Utilizado para apoiar o ensino em função do ritmo de aprendizagem do aluno.

Fonte: Adaptado de Fiocco (2007 *apud* Nascimento, 2016).

Independente da forma de apresentação do recurso digital – software de computador, plataforma *web* ou aplicativo para *smartphones* e *tablets* -, Sales e Dantas (2010) ressaltam a

interatividade e o *feedback* imediato como elementos-chave do recurso, capazes de proporcionar aprendizado dinâmico e provocar motivação nos estudantes.

O acesso a recursos tecnológicos, como Internet e *smartphones*, em território brasileiro passa por contínuo crescimento. Este crescimento foi averiguado por Taylor e Silver (2019) em pesquisa conduzida pelo *Pew Research Center* em 2018. Nesta, atestou-se o uso de *smartphones* por 85% da população brasileira com idade entre 18 e 34 anos, e de 32% para os indivíduos com 50 anos ou mais. O Instituto Brasileiro de Geografia e Estatística (IBGE, 2018) publicou, através da Agência de Notícias do IBGE, os resultados da Pesquisa Nacional por Amostra de Domicílios Contínua (PNAD Contínua) conduzida em 2017 sobre o uso de TIC e constatou a presença da Internet em 74,9% dos domicílios do Brasil, e, nestes, os *smartphones* são os equipamentos utilizados para o acesso à rede mundial de computadores em 98,7% das residências.

Mediante este panorama tecnológico vivenciado no país, estimula-se a investigação e desenvolvimento de ferramentas de apoio ao processo de ensino-aprendizagem que façam uso de recursos tecnológicos. Cientes desta realidade, Antônio da Silva, Dias da Silva e Martins (2018) afirmam ser possível aprimorar o rendimento dos alunos durante as aulas a partir do emprego de ferramentas computacionais no processo de ensino-aprendizagem.

Nos tópicos seguintes, são apresentados exemplos de ferramentas computacionais categorizadas em 03 (três) grandes grupos: juiz online, ambiente de desenvolvimento visual e compilador de Portugol. Para facilitar o contato com os softwares detalhados nos tópicos a seguir, o Apêndice B agrega os links de download das ferramentas e acesso às plataformas.

O apêndice C apresenta, de forma sucinta, ferramentas *mobile* (para dispositivos móveis) voltadas ao ensino-aprendizagem de algoritmos e programação. Desta forma, o aluno também pode ser estimulado a aprender os conteúdos introdutórios através de recursos de fácil acesso e sem estar limitado à utilização de um computador convencional.

1.6.1 Juiz online

Compreende-se por juiz online a plataforma para correção automática de códigos submetidos pelos usuários, os quais selecionam a linguagem de programação para codificação e obtém *feedback* imediato sobre a solução construída e submetida. O *feedback* objetiva

informar o usuário sobre o quanto sua solução é ou não adequada ao problema proposto (Kurnia *et al*, 2012 *apud* Dagostini *et al*, 2018).

Abaixo, são apresentados exemplos de ferramentas classificadas como “juiz online”.

1.6.1.1 CodeBench

O CodeBench é uma plataforma *web* de juiz online desenvolvida na IES 11. Na página *web* de apresentação da plataforma, são descritos os seus principais objetivos, listados abaixo:

- proporcionar aos discentes um conjunto de ferramentas pedagógicas de estímulo ao aprendizado;

- prover o docente de informações úteis sobre o progresso de aprendizado de seus alunos;

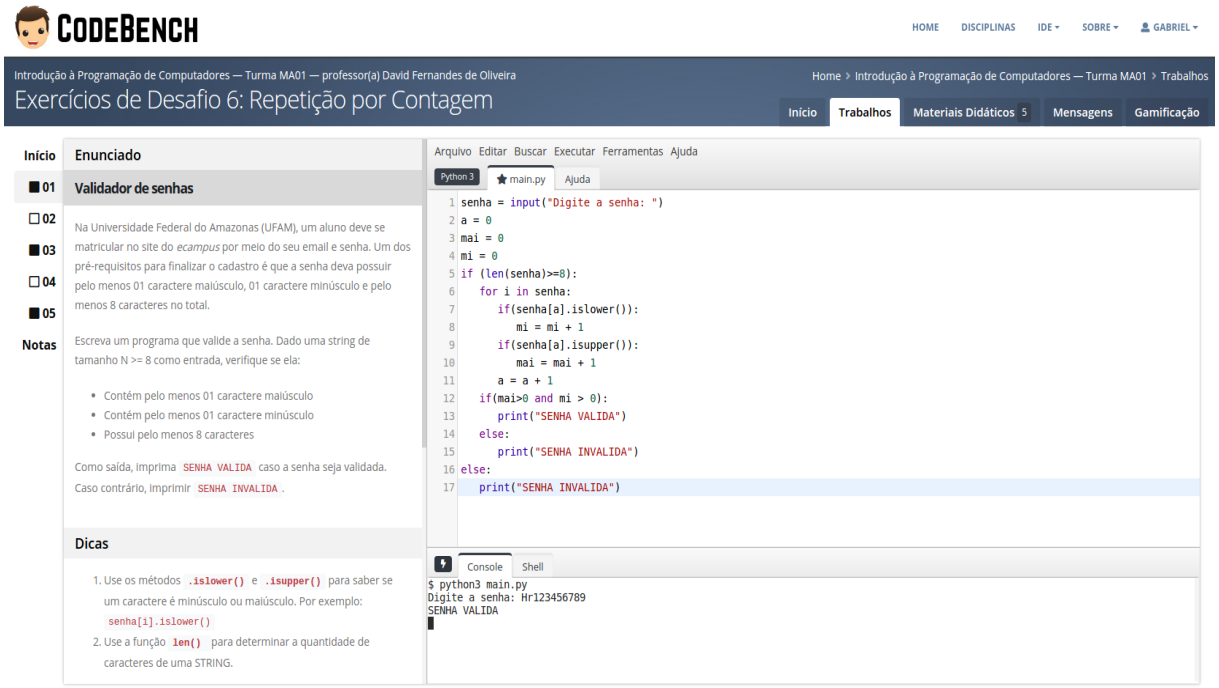
- proporcionar ao docente um conjunto de ferramentas facilitadoras ao seu trabalho;

- e fomentar a utilização de práticas modernas e criativas ao processo de ensino.

O funcionamento desta ferramenta é explicado por Carvalho, Oliveira e Gadelha (2016). O acesso do professor permite cadastrar novos exercícios – compondo base unificada de exercícios -, propor planos de teste (entradas e saídas esperadas), montar turmas e acompanhar o desempenho dos alunos de suas turmas face à resolução dos problemas disponibilizados. O acesso dos alunos possibilita resolver problemas com base nos conhecimentos adquiridos, submeter a solução em IDE integrada à plataforma e obter *feedback* em tempo real da solução proposta, com indicação de erros e sua localização no código.

A avaliação da solução submetida pelos estudantes é realizada em duas etapas: sintática e semântica. Na primeira, são averiguados possíveis erros de escrita do código com base no dicionário da linguagem de programação (C, C++, Java, Python, Haskell e Lua). Na segunda, é validada a saída obtida a partir do valor de entrada; para esta segunda etapa, os casos de teste submetidos pelo professor proponente do exercício atuam como balizadores para a validação. Na Figura 09 abaixo, é apresentada a interface da aplicação.

Figura 09 – Interface do CodeBench



Fonte: Instituto de Computação da UFAM ([201-?]).

Os autores conduziram aplicação do CodeBench na disciplina de Introdução à Programação de Computadores em 03 (três) turmas de cursos de graduação distintos que possuíam, no mesmo período, esta disciplina; este grupo foi denominado “experimental”. A mesma disciplina foi lecionada, paralelamente, em outras 05 (cinco) turmas, sem a utilização da plataforma; este grupo foi denominado “controle”. Ao final do semestre, foi comparado o índice de aprovação entre os dois grupos. Para o primeiro, obteve-se 70% de aprovação e, para o segundo, este percentual foi de apenas 30%.

Quando da conclusão do período letivo, os alunos e professores foram questionados sobre os principais atrativos observados e possíveis aspectos de aprimoramento. Mediante as respostas obtidas, foram adicionados à ferramenta novos recursos, dentre os quais destacam-se dois: gamificação e detecção de plágio. A primeira funcionalidade visa estimular os alunos na resolução dos exercícios, colocando-os como personagens de uma história na qual, para trilharem o caminho em direção a um vilão e derrotá-lo, necessitam acumular pontos de caminhada e força; estes são obtidos após o aluno lograr êxito na submissão da solução de um problema. A segunda funcionalidade é utilizada pelos docentes para avaliar possíveis casos de plágio em submissões realizadas pelos alunos das turmas, de modo a auxiliar o professor quando da atribuição de notas aos estudantes.

1.6.1.2 URI Online Judge

O URI Online Judge é uma plataforma também desenvolvida por equipe brasileira e possibilita exercitar o aprendizado de múltiplos conceitos de algoritmos e programação de computadores. Ao contrário do CodeBench, esta não permite ao professor e/ou tutor cadastrar novos problemas, sendo que os atualmente existentes são organizados de acordo com a problemática apresentada (Dagostini *et al*, 2018). Esta organização contempla problemas assim classificados:

- iniciantes;
- ad-hoc[2];
- strings;
- estruturas de dados e bibliotecas;
- matemáticos;
- paradigmas;
- grafos;
- geometria computacional;
- e *Structured Query Language* (SQL, em tradução, Linguagem de Consulta Estruturada).

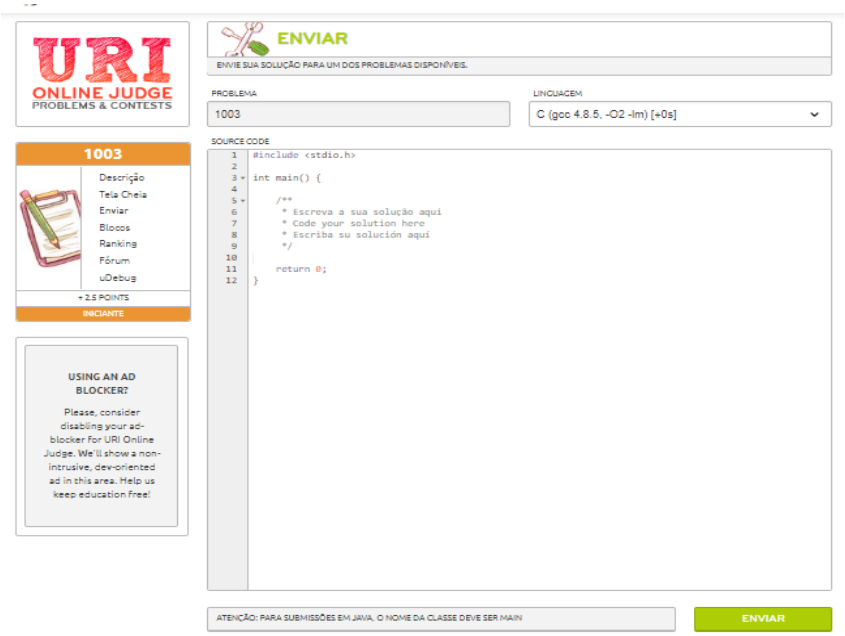
Atualmente, a ferramenta é ativamente utilizada em diferentes IES's, tais como a IES 02, a IES 19 e a *Baylor University*, por exemplo. A aceitação é fruto de sua eficiência ante à proposta de juiz online (URI Online Judge, [201-?]). Os exercícios disponibilizados seguem estrutura padrão: contextualização do problema (personagens e história fictícia), especificações da entrada e descrição da saída esperada (Dagostini *et al*¸2018). As Figuras 10 e 11, apresentadas abaixo, ilustram as interfaces de um problema de soma de dois valores e do ambiente para escrita e submissão do código.

[2] Ad-hoc é uma expressão latina cuja tradução refere-se a algo com uma finalidade específica. O elemento tido como ad-hoc é destinado "para um determinado fim".

Figura 10 – Interface da descrição de problema no URI Online Judge

Fonte: URI Online Judge ([201-?]).

Figura 11 – Interface do ambiente de escrita e submissão de código no URI Online Judge



Fonte: URI Online Judge ([201-?]).

O URI Online Judge pode ser aplicado a estudantes iniciantes e veteranos no que diz respeito ao domínio dos conceitos básicos de algoritmos e programação de computadores, visto que a base de problemas atualmente disponível contempla exercícios de baixa, média e elevada

complexidade e dificuldade. Desta forma, estimula-se o aprendizado e aprimoramento em todas as etapas do estudo (URI Online Judge, [201-?]).

Os principais aspectos de destaque da plataforma são: a possibilidade de o professor e/ou tutor criar sala dedicada ao acompanhamentos dos seus alunos; a variedade de linguagens de programação disponíveis – C, C++, Java, Pyton, Kotlin e outras, em um total de 17 (dezessete) –; a gratuidade, não sendo necessária a aquisição de licença ou pagamento de taxas regulares; e a disponibilidade online, a qual possibilita o acesso por meio da utilização de múltiplos sistemas operacionais e navegadores de internet (URI Online Judge, [201-?]; Dagostini *et al*¸2018).

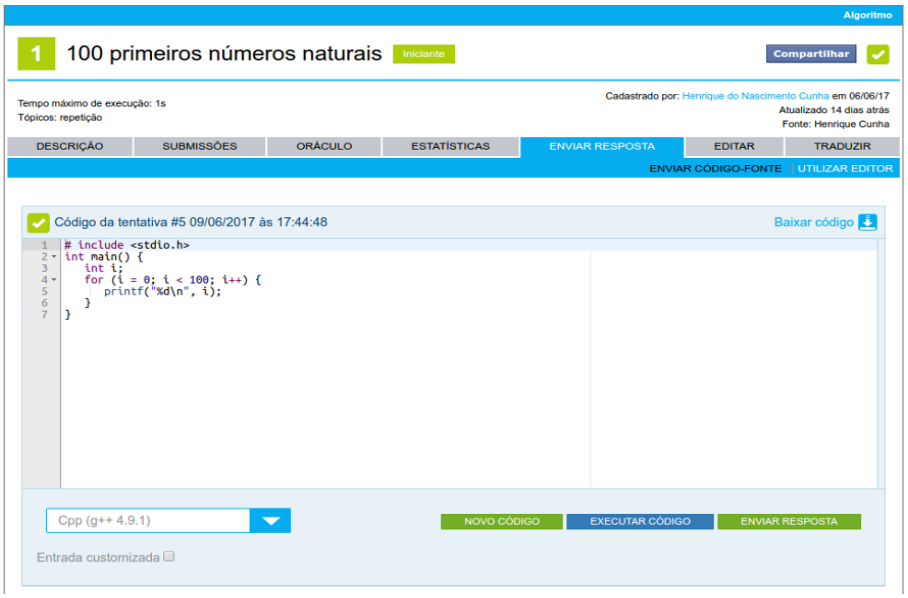
1.6.1.3 The Huxley

A plataforma The Huxley foi desenvolvida na IES 10 e surgiu com a proposta de atuar como apoio ao aprendizado de algoritmos e programação, oferecendo exercícios de lógica para serem utilizados para amadurecimento da compreensão e aplicação de conceitos introdutórios (Antônio da Silva, Dias da Silva e Martins, 2018).

A justificativa de desenvolvimento deste juiz online incide sobre o estímulo ao aprendizado de programação e à praticidade na correção de exercícios, obtenção de *feedback* e acompanhamento individualizado dos estudantes (Paes *et al*, 2013).

Ao acessar o site da plataforma, é disponibilizado ambiente próprio para o desenvolvimento do código, o qual é submetido e validado com base em casos de teste. O aluno pode selecionar a linguagem de programação na qual o código é escrito, estando disponíveis 07 (sete) opções, dentre as quais C, C++, Java e Python. Em situações de dificuldades e/ou dúvidas, os monitores e professores cadastrados podem ser consultados para dicas e orientações. Os professores podem então organizar os alunos em turmas, para as quais podem ser desenvolvidas listas de exercícios e avaliações com base em exercícios existentes na plataforma e/ou novos problemas desenvolvidos e submetidos pelos professores (Antonio da Silva, Dias da Silva e Martins, 2018). Na Figura 12 abaixo, está ilustrada a interface de escrita e submissão de código da plataforma.

Figura 12 – Interface de escrita e submissão de código no The Huxley



Fonte: The Huxley ([201-?]).

De modo a validar o uso do The Huxley quanto aos seus objetivos didáticos, Antônio da Silva, Dias da Silva e Martins (2018) aplicaram o mesmo a alunos do curso técnico Informática para Internet ao cursarem a disciplina Lógica de Programação. Após a utilização da ferramenta por 01 (um) semestre para o desenvolvimento de códigos pertinentes aos conteúdos ministrados durante as aulas, os estudantes foram questionados sobre a importância da ferramenta para o aprendizado. A maior motivação para a resolução dos exercícios e melhor compreensão dos conteúdos foi amplamente citada, e a aprovação dos estudantes com conceito A – pleno entendimento teórico e prático dos componentes curriculares – aumentou em 13,22%.

Os benefícios da ferramenta podem ser percebidos por alunos e professores. Para os primeiros, o ranqueamento mediante obtenção de pontos e medalhas estimula o acesso constante à plataforma, ultrapassando os limites do espaço educacional (escola, universidade e afins). Para os professores, a correção automatizada de exercícios pela ferramenta garante-lhes tempo para análise do desempenho individualizado dos alunos – via *dashboard* e planilhas -, possibilitando atribuir notas a partir do resultado de cada estudante, assim com analisar as principais dificuldades vivenciadas pelos acadêmicos e, desta forma, atuar para a mitigação das mesmas (Paes *et al*, 2013).

1.6.2 Ambiente de desenvolvimento visual

Esta categoria de software contempla aplicações cujos comandos, ao serem codificados pelos usuários, são traduzidos em ações executadas por elemento visual animado, como um personagem. Este comportamento é relevante para compreender a lógica de funcionamento e o impacto da utilização das estruturas básicas envolvidas na construção de algoritmos e programas de computador (Souza, Batista e Barbosa, 2016).

Abaixo, são apresentados exemplos de ferramentas classificadas como “ambiente de desenvolvimento visual”.

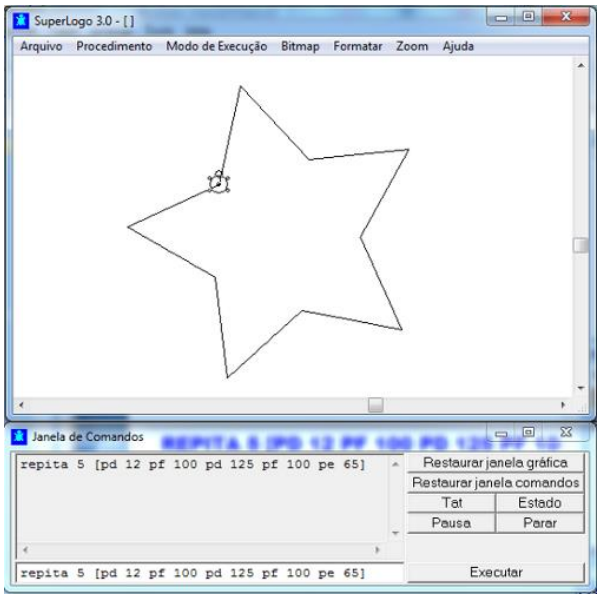
1.6.2.1 LOGO

A linguagem de programação LOGO surgiu na década de 1960 e é atribuída a Seymour Papert, cientista do *Massachusetts Institute of Technology* (MIT, em tradução, Instituto de Tecnologia de Massachusetts) influenciado pela Teoria do Construtivismo[3] de Jean Piaget. O nome LOGO remete à palavra grega “logos”, a qual é conceituada como “razão”, “pensamento”, “ciência”, dentre outros. Atrelada a uma filosofia própria, prega o aprendizado pelo erro, sob o qual o aluno, ao cometer erro durante a programação, é estimulado a refletir sobre o mesmo e corrigi-lo, potencializando sua compreensão sobre a linguagem e, acima de tudo, sobre o problema a ser resolvido. Em resumo, pode ser definida como a aprendizagem pela descoberta (Universidade Federal de Lavras, [200-?]).

A variedade de conteúdos passível de ensinamento a partir da utilização do LOGO é considerável, assim como as diferentes IDEs atualmente disponíveis para a codificação. O SuperLogo, MegaLogo, xLogo e KTurtle são exemplos de ambientes de desenvolvimento os quais implementam a linguagem LOGO e se caracterizam pela apresentação de duas interfaces distintas: área de código e área de visualização. Na primeira, o usuário escreve o código em linguagem semelhante ao português estruturado; e na segunda, uma tartaruga – anteriormente robô – realiza ações com base nos comandos informados. Na Figura 13 abaixo, é ilustrada a interface do ambiente SuperLogo.

[3] Esta teoria compreende o ensino como uma construção constante, mediante a qual o conhecimento é continuamente desenvolvido a partir da interação entre o sujeito e o meio. Na educação, reforça a necessidade de “aprender a aprender”.

Figura 13 – Interface do SuperLogo



Fonte: Motta e Miranda (2018).

De modo a verificar o impacto da aplicação do LOGO em ambiente educacional, Raiol *et al* (2015) organizaram oficina de introdução a algoritmos e programação com estudantes iniciais do curso Bacharelado em Sistemas de Informação da IES 15 durante o primeiro semestre de 2013. Para esta dinâmica, foi utilizada a IDE KTurtle e focou-se no ensino dos seguintes conceitos: algoritmo, raciocínio lógico, variáveis, entrada e saída de dados, operadores (aritméticos, lógicos e relacionais), estruturas de seleção e repetição e funções. Ao término do semestre, verificou-se que os participantes da oficina apresentaram média superior nas disciplinas de Técnicas de Programação I e Lógica Matemática quando comparados aos seus colegas que não haviam comparecido à oficina.

Os principais atrativos desta tecnologia são: a amigabilidade – viabiliza o aprendizado para estudantes de qualquer faixa etária, de crianças a adultos -; a interatividade – exibição de resposta imediata e lúdica aos comandos do usuário –; e a localização do idioma – os comandos são traduzidos para o português brasileiro, incorrendo em facilidade na compreensão e aplicação dos mesmos (Universidade Federal de Lavras, [200-?]; Raiol *et al*, 2015).

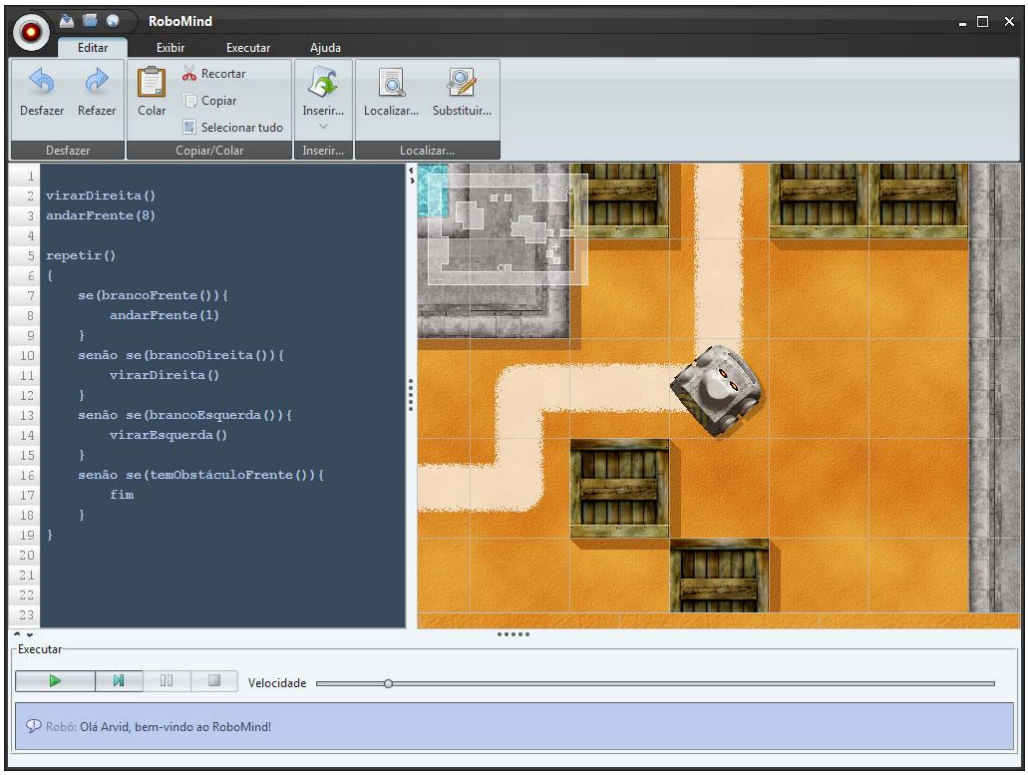
1.6.2.2 RoboMind

O RoboMind é uma IDE cujos princípios são semelhantes aos dos ambientes que implementam a linguagem LOGO, explanada no tópico anterior. Entretanto, ao contrário desta, os programas são codificados em linguagem própria – Robo/Roo – e o elemento central da parte visual é um robô, ao invés de uma tartaruga. O pesquisador Arvid Halma, da Universidade de Amsterdã, é o responsável pela criação do ambiente e de sua linguagem, cujo foco é o contexto educacional de introdução à programação de computadores, Inteligência Artificial e Robótica (Silva *et al*, 2014).

Concernente ao funcionamento do RoboMind, a linguagem Robo/Roo é utilizada para a escrita do código, o qual é refletido em ações executadas por um robô cuja movimentação é exercida sobre um plano bidimensional em formato de quadrado (Nofitasari, Yuana e Maryono, 2017, tradução nossa). Em outras palavras, o aspecto visual - constituído de um robô, um terreno plano para movimentação e obstáculos – é reflexo da programação codificada na linguagem da IDE (Yuana e Maryono, 2016, tradução nossa).

As ações passíveis de execução pelo robô consistem em mover-se e curvar-se para frente, trás, esquerda e direita; pintar o solo durante o seu deslocamento, agarrar e posicionar objetos em frente ao robô; e lançar moeda para executar ação aleatória. De modo a combinar estas ações e propor soluções para os exercícios, é possível ensinar conceitos introdutórios de algoritmos e programação de computadores, a saber: variáveis, expressões lógicas e aritméticas, estruturas sequencial, condicional e de repetição, funções e comentários (Universidade de Amsterdã, [201-?], tradução nossa). Além disso, ao obter-se as possíveis soluções de um exercício, é possível testá-las e avaliá-las quanto à sua eficiência observando o tempo gasto pelo robô para executar as ações codificadas (Yuana e Maryono, 2016, tradução nossa). Na Figura 14 abaixo, é exibida a interface do RoboMind.

Figura 14 – Interface do RoboMind



Fonte: Universidade de Amsterdã ([201-?]).

Para fins de validação da ferramenta em contexto educacional, Nofitasari, Yuana e Maryono (2017) conduziram estudo mediante aplicação da ferramenta com estudantes da disciplina de Programação Estruturada da IES 01. Para a maioria destes, o primeiro contato com a programação de computadores ocorreu na universidade, sem a existência de experiência prévia com a prática. O objetivo deste estudo centrou-se no ensino de conceitos básicos de algoritmos e programação, a saber: estruturas sequencial, condicional e de repetição e modularização por funções. Foram dedicadas 04 (quatro) aulas ao ensino dos conceitos e resolução de 05 (cinco) exercícios de cada conteúdo por aula ministrada, com a resolução destes implementada no RoboMind. Os resultados obtidos neste estudo validaram a importância da ferramenta face aos objetivos propostos: 87% dos estudantes sentiram-se motivados em continuar o estudo de algoritmos e programação; quanto às estruturas, 97% compreenderam a sequencial, 88% compreenderam a condicional, 77% a de repetição e 85% a modularização por funções.

Os principais atrativos deste software são: a facilidade de compreensão da linguagem – passível de entendimento por iniciantes na programação de computadores -; a gratuidade, não sendo necessária a aquisição de licença ou pagamento de taxas regulares; a disponibilidade

multiplataforma, com versões para os sistemas operacionais Windows, Linux e Mac OS X; e o *feedback* imediato, com indicação da linha na qual encontra-se o erro de codificação (Nofitasari, Yuana e Maryono, 2017, tradução nossa). Embora não exista tradução para o português brasileiro, esta situação é comum em outras IDEs e vista como estímulo para o aprendizado do idioma inglês, tão necessário ao aprendizado de programação de computadores.

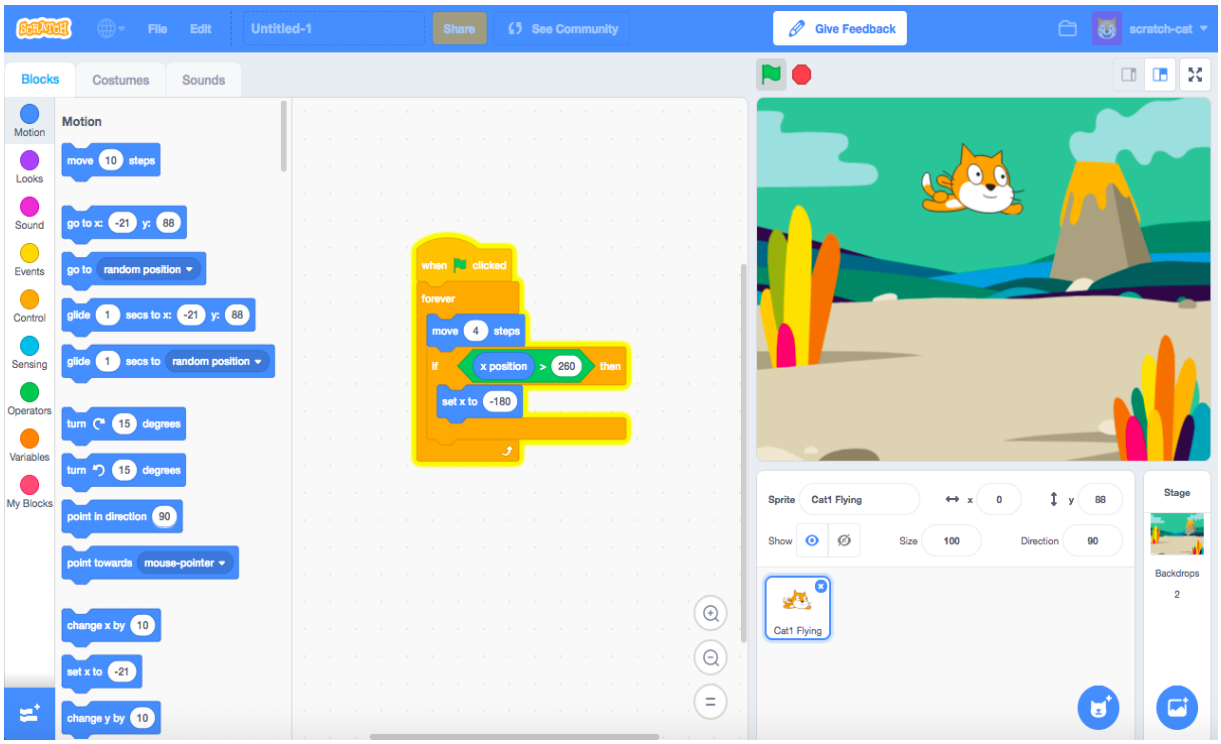
1.6.2.3 Scratch

O Scratch é uma linguagem de programação desenvolvida no MIT em 2007 pelo grupo *Lifelong Kindergarten*, voltada ao ensino de conceitos básicos de programação de computadores e *design* (Salazar, Odakura e Barvinski, 2015). O seu nome é uma referência ao *scratching*, movimento utilizado pelos músicos do *hip-hop* para girar os discos de vinil para frente e para trás e, dessa forma, produzir efeitos sonoros e músicas originais. Neste sentido, a ferramenta permite, em ambiente digital, misturar diferentes tipos de mídias – imagens, sons e afins – para a produção de conteúdos lúdicos e criativos (Trentin *et al*, 2013).

A premissa do Scratch é possibilitar o aprendizado de programação de computadores por qualquer pessoa, sem restrição de idade e conhecimentos prévios. Os desenvolvedores podem criar histórias, animações, jogos, simulações, desafios e outros e, por conseguinte, aprender e potencializar noções de programação, de matemática e de *design* (Salazar, Odakura e Barvinski, 2015).

O aspecto lúdico da ferramenta é viabilizado pela apresentação dos comandos disponíveis e área de execução de ações. Os comandos são organizados em blocos, categorizados pela lógica de funcionamento e distinguidos por cor. À medida que os blocos são arrastados e encaixados entre si, é possível estruturar programas de diferentes níveus de complexidade. O elemento principal da área de execução de ações é um gato de cor alaranjada – ícone do software -, o qual coloca em prática a lógica estruturada no programa. Diversos outros elementos visuais podem ser acrescidos nesta região a depender da criatividade e objetivos do programador (Trentin *et al*, 2013). A Figura 15 ilustra a interface do Scratch 3.0 – versão mais recente da aplicação.

Figura 15 – Interface do Scratch 3.0



Fonte: The Scratch Team (2018).

No que tange aos conceitos de algoritmos e programação que podem ser ensinados a partir da utilização do Scratch, Oliveira, Rodrigues e Queiroga (2016) listam: variáveis, operadores (aritméticos, lógicos e relacionais), estruturas condicionais e de repetição, vetores e funções. Para verificar a eficiência no contexto educacional do ensino superior, destacam-se 02 (dois) estudos, cada qual com percepções distintas.

Oliveira, Rodrigues e Queiroga (2016) aplicaram a ferramenta para alunos da disciplina de Programação Estruturada do curso de Tecnologia em Análise e Desenvolvimento de Sistemas da IES 05, totalizando 40 alunos. Nesta, materiais didáticos e lúdicos com o conteúdo de lógica de programação foram apresentados aos alunos antes da abordagem dos mesmos em linguagem de programação formal[4], neste caso, o C++. Nesta mesma lógica, sempre que possível, foi proposto aos alunos a resolução de exercícios no Scratch para, posteriormente, resolvê-los no C++, utilizando a IDE DevC++. Os resultados obtidos ao final do semestre foram satisfatórios e podem ser traduzidos a partir da percepção dos alunos e

[4] Toda e qualquer linguagem de programação utilizada para o desenvolvimento de sistemas computacionais a nível profissional, voltados à utilização em ambientes diversos, como o corporativo. A caráter de exemplos, tem-se o Java, o PHP, o C++, o Python, dentre outras.

índices de aprovação. Os estudantes pontuaram: a maior facilidade na compreensão dos conceitos básicos para então traduzi-los em linguagem de programação formal; o aspecto visual contextualiza e possibilita acompanhar as ações, sendo possível identificar erros e corrigi-los com maior facilidade; o fortalecimento de noções matemáticas – ângulos e operadores – e de *design*. Quanto às aprovações, o índice foi de 92%, fruto da maior liberdade, criatividade, confiança e satisfação dos alunos nas aulas e atividades extraclasse. Assim, concluiu-se que este software pode ser utilizado em aulas práticas antecedentes à apresentação do conteúdo em uma linguagem formal de programação, como C, C++, Pascal e Java.

Salazar, Odakura e Barvinski (2015) desenvolveram estudo com foco no elemento "motivação". Para tal, o Scratch foi apresentado a alunos de graduação em Computação – curso não especificado – da IES 13 cuja trajetória acadêmica já tivesse contemplado disciplina introdutória de algoritmos e programação. A apresentação consistiu em aula introdutória de 50 (cinquenta) minutos sobre o Scratch e sua aplicabilidade para o ensino de conceitos básicos de algoritmos e programação; em seguida, os alunos desenvolveram exercícios contemplando os conceitos repassados na etapa introdutória. Após esta fase prática, os alunos foram submetidos a um questionário para verificação de percepções. Quanto à utilidade da ferramenta para o aprendizado de algoritmos e programação, 71% relataram que seria muito útil, 10% útil e 19% pouco útil. Ao serem questionados a traçarem comparativo entre as aulas de laboratório vivenciadas durante as disciplinas e a aula de Scratch, 72% julgaram que seria interessante o seu uso antes da apresentação de linguagem formal de programação, 24% responderam que a mesma é empolgante porém insuficiente ao ensino dos conceitos, e 4% avaliaram-na como viável à substituição das linguagens já utilizadas na universidade. Mesmo com os índices positivos apresentados, os autores concluíram que a aplicação deste software não é suficiente ao aprendizado dos conceitos básicos, embora possa atuar como elemento motivador para o aprendizado.

Por fim, os principais pontos positivos deste ambiente são: possui aspecto lúdico – forte apelo à visualização da solução e abstração das situações-problema - ; é gratuita; é um software livre; permite aos usuários compartilhar projetos em plataforma *online* e, assim, tornar sua produção acessível aos demais usuários; a disponibilidade multiplataforma, com aplicação offline (executável em Windows e Mac OS X) e plataforma online com IDE integrada; e a promoção do interesse por outras áreas do conhecimento, como engenharia, matemática e *design* (Oliveira, Rodrigues e Queiroga, 2016; Trentin *et al*, 2013; Salazar, Odakura e Barvinski, 2015).

1.6.3 Compilador de Portugol

O compilador de Portugol apresenta-se como componente de um ambiente de desenvolvimento no qual é possível escrever e verificar a validade de códigos escritos em pseudo-linguagem português estruturado, formalmente denominado no ensino de algoritmos e programação como “Portugol” (MACP, [201-]).

Esta pseudo-linguagem é uma mescla de termos em português e símbolos comuns em linguagens de programação formais, como operadores aritméticos, lógicos e relacionais. Por meio desta, é possível conceituar e aplicar conceitos básicos de algoritmos e programação de computadores, como: tipos de dados, variáveis, constantes, estruturas condicional e de repetição, vetores, matrizes e funções (Noschang *et al*, 2014).

Abaixo, são apresentados exemplos de ferramentas cujo compilador é caracterizado como “compilador de Portugol”.

1.6.3.1 Portugol Studio

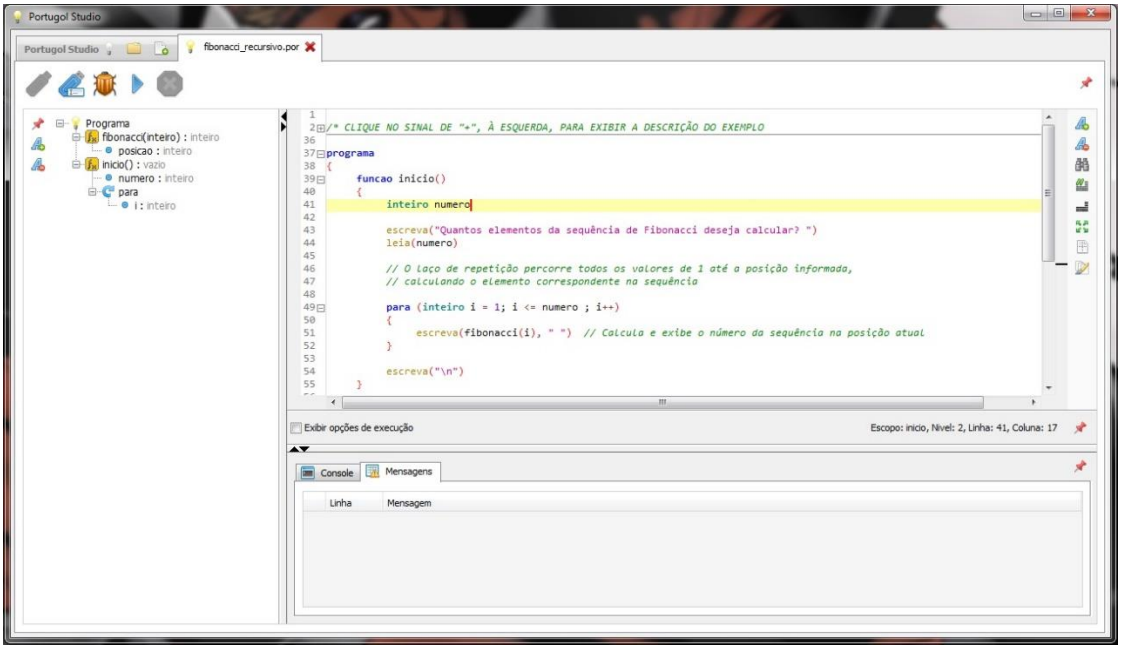
Esta IDE é fruto do trabalho desenvolvido por estudantes, professores e pesquisadores da IES 18. A primeira versão do software foi lançada em 2013 e passa por constantes atualizações (LITE, [201-?]).

Ao contrário de outras IDEs populares no ambiente universitário e profissional, tais como DevC++, Eclipse e NetBeans, o Portugol Studio é notadamente didático, voltado ao ensino de conceitos de algoritmos e programação de computadores e amadurecimento do raciocínio lógico. A premissa básica para a sua concepção é a importância primária de aprender a resolver problemas, para posterior foco no aprendizado de uma linguagem formal (Noschang *et al*, 2014).

De modo a obter êxito como sistema computacional de apoio ao aprendizado, o Portugol Studio oferece recursos para tornar mais agradável as experiências iniciais com a programação. Dentre os principais recursos disponibilizados pelo ambiente, destacam-se: sintaxe em português – comandos, mensagens e menus -; destaque visual (realce em cores) para as palavras reservadas da linguagem; mensagens com dicas e orientações para correção de erros de sintaxe e semântica; depuração de código de fácil configuração, para acompanhamento em

tempo real dos valores das variáveis utilizadas no código; e seção Ajuda contendo exemplos de utilização dos conceitos introdutórios, todos estes com possibilidade de execução, alteração e depuração. A interface da plataforma é constituída de 03 (três) regiões principais: árvore de símbolos – apresenta as variáveis com seus respectivos valores e escopo -; código fonte para a escrita da solução em Portugol; e console para entrada e saída de dados (Noschang *et al*, 2014). Na Figura 16 abaixo é mostrada a interface do Portugol Studio.

Figura 16 – Interface do Portugol Studio



Fonte: Moreira (2015).

Atualmente, a ferramenta é utilizada como recurso primário para o ensino introdutório de conceitos de algoritmos e programação por universidade federal no estado de São Paulo e pelos alunos dos cursos da área de Computação da UNIVALI. Paralelamente, a ferramenta contabiliza aproximadamente 275.000 (duzentos e setenta e cinco mil) *downloads* (LITE, [201-?]).

Face à utilização da ferramenta para fins didáticos, Anido (2014) reitera a importância da atual distribuição do Portugol Studio como um software livre, visto permitir contribuições e consequente evolução dos recursos oferecidos. Ainda assim, ressalta que a ferramenta poderia agregar novos usuários e expandir sua popularidade mediante a implementação de uma versão *web*, a qual possibilitasse a utilização do ambiente sem a necessidade de instalação – disponível

apenas para Windows -, visto a popularização de dispositivos móveis, tais como *smartphones* e *tablets*.

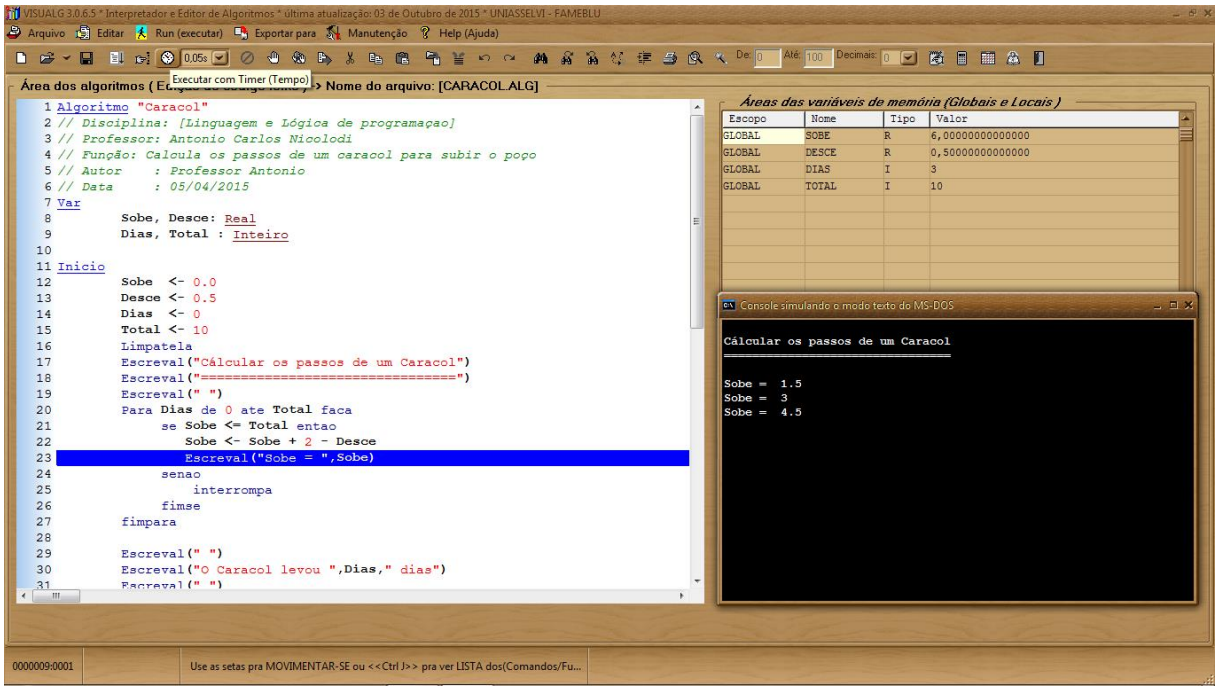
1.6.3.2 VisuAlg

O VisuAlg é uma IDE voltada ao ensino de conceitos introdutórios de algoritmos e programação de computadores. Desenvolvida em 1986 pelos professores Antonio Carlos Nicolodi e Cláudio Morgado de Souza, a ferramenta atualmente encontra-se na versão 3.0, e a versão 4.0 está em fase de desenvolvimento (VisuAlg, [201-?]). O objetivo deste software é agilizar e facilitar o aprendizado de conceitos introdutórios de algoritmos e programação (Leite *et al*, 2013). Parafraseando Dantas *et al* (2012), esta IDE permite exercitar a prática de programação em um ambiente próximo da realidade do aluno ao utilizar linguagem de fácil compreensão e adaptação, incorrendo em maior produtividade e motivação para a aquisição do conhecimento introdutório.

A partir de comandos escritos em Portugol, o VisuAlg permite abordar: tipos de dados, variáveis, constantes, estruturas de fluxo – sequencial, condicional e de repetição -, vetores, matrizes, funções, procedimentos e recursão. Realizada a codificação, é possível testar a implementação e a lógica empregada na construção da solução. Por vias automatizadas, é possível aplicar testes de mesa, relacionando as entradas fornecidas pelo usuário às saídas esperadas mediante o processamento daquelas. Os menus, comandos e mensagens são apresentados no idioma português e o aluno foca na construção, teste e análise da solução, abstraindo de recursos marcadamente técnicos característicos de linguagem de programação formal (Leite *et al*, 2013).

A funcionalidade da aplicação, resultante da disponibilização dos recursos acima mencionados, é um dos destaques deste software. Segundo Dantas *et al* (2012), a facilidade de utilização dos recursos e de entendimento da linguagem proporcionam aos alunos experiência focada na análise dos problemas a serem solucionados e das soluções propostas, a partir de *feedback* instantâneo. A Figura 17 abaixo exibe uma das possibilidades de personalização da interface do VisuAlg 3.0.

Figura 17 – Interface do VisuAlg 3.0



Fonte: VisuAlg ([201-?]).

Para expor a validade do VisuAlg no contexto didático, a ferramenta foi aplicada por Leite *et al* (2013) em turma de Técnicas de Programação do curso de Ciência da Computação da IES 09. Neste contexto, 68% dos estudantes não possuíam experiência prévia com programação e desta forma figuraram turma pertinente para a validação dos aspectos didáticos do software. Após a utilização do VisuAlg para o ensino inicial dos conceitos elicitados no parágrafo anterior, os discentes foram questionados sobre o quão a ferramenta havia influenciado o aprendizado: 81% dos discentes julgaram satisfatório o aprendizado dos conceitos pelo uso da ferramenta; 79% classificaram a interface da aplicação como agradável e simples; e os mesmos 79% consideraram fácil a utilização do compilador.

Com o amadurecimento da compreensão e aplicação dos conceitos, faz-se necessário introduzir linguagem de programação formal, de modo a capacitar o estudante para atuação no meio profissional. Desta forma, para os estudantes que já possuem experiência com programação, assim como para profissionais da área, o VisuAlg pode não ser satisfatório e adicionalmente tornar a experiência de programar uma atividade maçante e desmotivadora (Leite *et al*, 2013). Aos iniciantes e inexperientes no ato de programar, tem-se nesta uma ferramenta que age como facilitador ao instigar o aprendizado por meio da aquisição de novos

conhecimentos, aplicação dos mesmos na resolução de problemas e identificação e correção de erros sintáticos e semânticos (Dantas *et al*, 2012).

CAPÍTULO 2 - METODOLOGIA

Abaixo, encontram-se detalhados os aspectos que nortearam o desenvolvimento deste trabalho. Este capítulo encontra-se dividido em duas seções, a saber: caracterização da pesquisa e instrumentos e procedimentos para a coleta dos dados.

2.1 Caracterização da pesquisa

Os estudos desenvolvidos por Vergara (2016) ressaltam a importância de classificar uma pesquisa quantos aos fins e aos meios. A partir da taxonomia desenvolvida por essa autora, esta pesquisa é classificada como exploratória e descritiva.

Exploratória porque é verificada pequena – porém crescente - quantidade de estudos os quais propõem conhecer e sistematizar métodos e/ou abordagens e ferramentas computacionais de apoio ao processo de ensino-aprendizagem de programação. Além deste fato, a pesquisa atuará como subsídio para o desenvolvimento futuro de aplicações computacionais.

Descritiva pois expõe as principais dificuldades vivenciadas pelos alunos e professores e busca associar aquelas à disponibilidade de métodos e/ou abordagens e ferramentas computacionais de apoio ao processo de ensino-aprendizagem de programação.

Em relação aos meios discorridos por Vergara (2016), a presente pesquisa é bibliográfica, documental e de campo.

Bibliográfica pois recorre à consulta e análise de materiais bibliográficos previamente publicados sob a forma de livros, artigos, revistas, anais e correlatos. Tais materiais encontram-se disponíveis em formato analógico e/ou digital e configuram-se de primeira mão – originais, idênticos quanto ao conteúdo, formato e veículo de publicação – e segunda mão – editados no que tange ao conteúdo, formato e/ou veículo de publicação.

Documental pois utiliza de material desenvolvido na Universidade Estadual de Goiás e cedido a esta instituição, configurando acervo patrimonial do órgão público supracitado.

Pesquisa de campo pois envolve a aplicação de questionário para a coleta de dados sobre as dificuldades vivenciadas, os métodos e/ou abordagens e as ferramentas computacionais utilizadas pelos professores no ensino de algoritmos e programação de computadores.

2.2 Instrumentos e procedimentos para a coleta dos dados

Para a exploração dos assuntos norteadores deste trabalho, recorreu-se à revisão bibliográfica. Esta foi adotada mediante a pesquisa de materiais disponíveis em meios analógicos e sobretudo digitais. Para cada tópico e subtópico pertencente ao Referencial Teórico, focou-se na pesquisa de bibliografia cujo conteúdo principal do material fosse o assunto do tópico e/ou subtópico. Encontrado o material, o mesmo foi salvo para posterior leitura e análise.

A leitura prévia contemplou o título e resumo e serviu como filtro para separar os materiais que seriam lidos na íntegra dos que seriam descartados em função de inadequações à pesquisa. Após a leitura íntegra dos materiais selecionados, cada qual foi catalogado no software livre Zotero – ferramenta de gerenciamento de dados bibliográficos. Neste, foi possível salvar anotações gerais, citações diretas e indiretas a serem utilizadas no trabalho, além de possibilitar a obtenção automatizada de referência bibliográfica no padrão da Associação Brasileira de Normas Técnicas (ABNT).

A condução da pesquisa foi pautada na formulação de um protocolo de pesquisa, o qual expõe elementos envolvidos na produção deste trabalho. O Quadro 07 a seguir apresenta maiores detalhes sobre este protocolo.

Quadro 07 – Protocolo de pesquisa

Elemento	Detalhes
Questão de pesquisa	Face às dificuldades vivenciadas pelos alunos nas disciplinas iniciais de algoritmos e programação, quais são as abordagens e ferramentas computacionais atualmente disponíveis para potencializar o ensino destas disciplinas?
Fonte de dados	- Base Nacional Comum Curricular do MEC. - Comissão Especial de Informática na Educação (CEIE). - Google Acadêmico. - IEEE Xplore Digital Library. - Patrimônio da Universidade Estadual de Goiás. - Portal de Conteúdo da Sociedade Brasileira de Computação (SBC). - Research Gate.
Tipo de literatura	- Artigos. - Dicionário. - Documentação de softwares. - Livros. - Normas Técnicas. - Notícias. - Resoluções. - Trabalho de Conclusão de Curso.
Idioma	- Inglês. - Português.
Período de busca	2004 a 2019.
Strings de busca	- (“conceito” or “definição” or “explicação”) and (“algoritmo”).

	- (“dificuldades” or “problemas”) and (“alunos” or “estudantes”) and (“ensino” or “aprendizado”) and (“programação” or “algoritmos”). - (“ferramentas” or “aplicações” or “softwares”) and (“ensino” or “aprendizado”) and (“algoritmos” or “programação”). - (“método” or “abordagens” or “meios” or “estratégias”) and (“ensino” or “aprendizado”) and (“algoritmos” or “programação”). - (“perfil” or “características” or “padrão”) and (“estudantes” or “alunos” or “acadêmicos”) and (‘computação” or “software” or “algoritmo”). - “ensino de algoritmos”. - “estratégia” and “algoritmos”. - "ferramentas" and "ensino" and "algoritmos".

Fonte: Do autor (2019).

Durante a pesquisa de material para composição do Referencial Teórico, foi averiguada necessidade de conhecer a realidade do processo de ensino-aprendizagem de algoritmos e programação vivenciada por docentes do ensino superior na área de Computação.

Neste contexto, foi desenvolvido questionário - a partir do recurso “Formulários” oferecido pela plataforma Google – com questões pertinentes às dificuldades e à utilização de métodos e/ou abordagens não-tradicionais e ferramentas computacionais de apoio ao processo de ensino-aprendizagem de algoritmos e programação.

O público-alvo do questionário foram professores do ensino superior da área de Computação, os quais já tenham lecionado ou lecionem disciplina(s) introdutória(s) de algoritmos e programação. A análise e a discussão dos resultados obtidos encontram-se disponíveis no Capítulo 3.

CAPÍTULO 3 - ANÁLISE DOS RESULTADOS

Nesta seção, são apresentadas informações sobre o questionário submetido aos professores e realizada a análise dos dados obtidos mediante as respostas dos docentes. A análise contempla a exposição dos resultados obtidos e a discussão realizada para cada um dos questionamentos.

3.1 Informações gerais do questionário

O objetivo desta pesquisa – via questionário - foi identificar a forma de atuação dos professores, assim como métodos e/ou abordagens e ferramentas computacionais de apoio utilizadas no ensino. É importante ressaltar que esta coleta de dados foi complementar ao foco principal do estudo, o qual é a exposição de abordagens não tradicionais e ferramentas computacionais de apoio ao processo de ensino-aprendizagem.

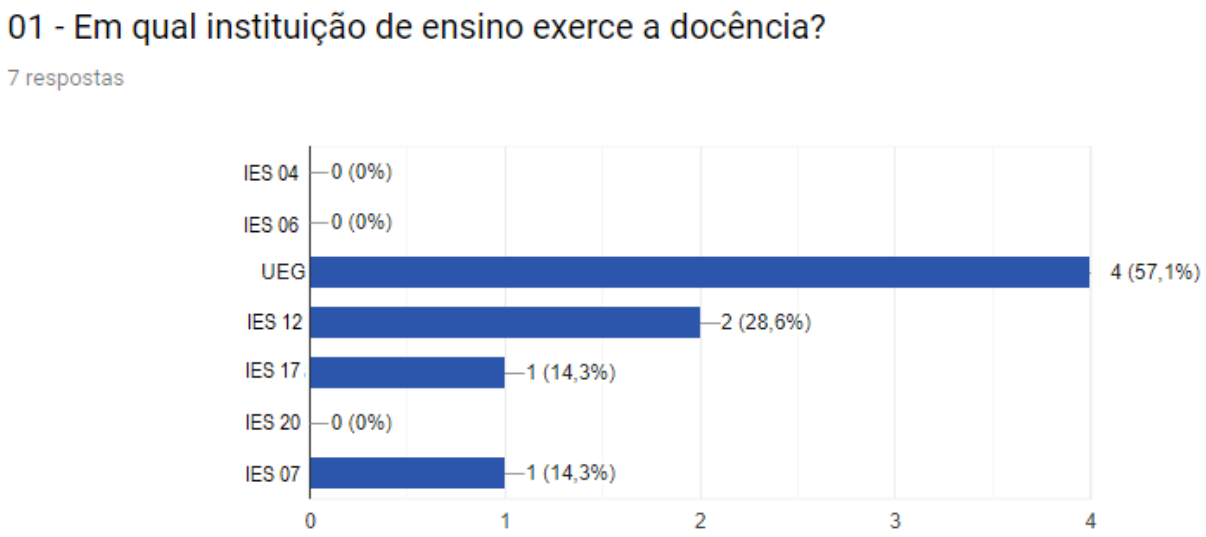
O questionário foi composto de 10 (dez) perguntas objetivas e subjetivas elaboradas pelo orientando e orientador deste trabalho. O link para acesso foi distribuído através da plataforma WhatsApp. O repasse foi realizado a docentes do ensino superior em Computação os quais ministram e/ou já ministraram alguma disciplina de algoritmos e programação de computadores. Nesta submissão, foram considerados docentes das seguintes IES: IES 04, IES 06, UEG, IES 12, IES 10, IES 17 e IES 07.

O prazo para o aceite de respostas foi de 15 (quinze) dias, tendo sido iniciado em 10 de agosto de 2019 e encerrado em 24 de agosto de 2019. Quando do encerramento, foi verificado o total de 07 (sete) professores respondentes e, posteriormente ao encerramento, foi realizada a análise dos resultados, a qual é apresentada no tópico a seguir.

3.2 Análise das respostas do questionário

A primeira pergunta buscou evidenciar as IES nas quais os docentes ministram ou já ministraram disciplina de algoritmos e programação de computadores. No Gráfico 01, são exibidos os resultados.

Gráfico 01 – IES dos respondentes



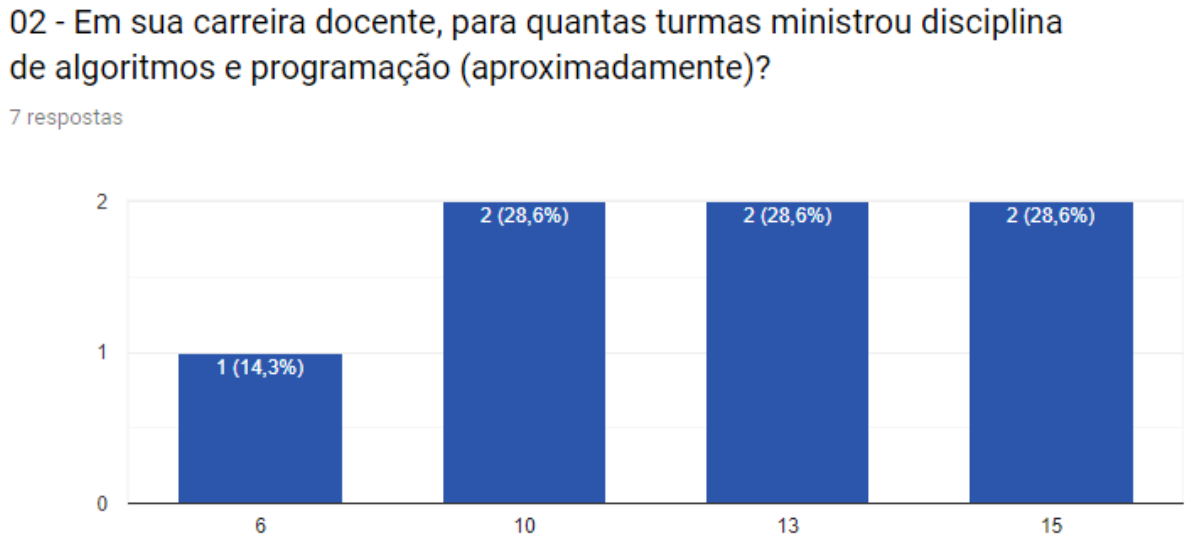
Fonte: Do autor (2019).

Foi observada maior participação de professores da UEG (57%), seguida por docentes da IES 12 (28,6%), IES 17 (14,3%) e IES 07 (14,3%). Mediante contabilização do total de respondentes, verificou-se em planilha auxiliar ao questionário (gerado pela própria ferramenta Google Formulários) que o professor da IES 17 também é docente na IES 07, totalizando os 07 (sete) respondentes. Quanto às IES 04, IES 06 e IES 20, nenhum professor respondeu ao questionário.

Esta maior participação proveniente da UEG e IES 12 pode ter sido reflexo da proximidade do orientando e orientador para com os docentes das IES, os quais demonstraram imediato interesse em contribuir com o estudo face à realidade do processo de ensino-aprendizagem.

O foco da segunda pergunta foi expor a experiência dos professores no ensino de algoritmos e programação a partir do questionamento do quantitativo de turmas para as quais lecionou. No Gráfico 02, são expostos os resultados.

Gráfico 02 – Quantidade de turmas lecionadas pelos respondentes



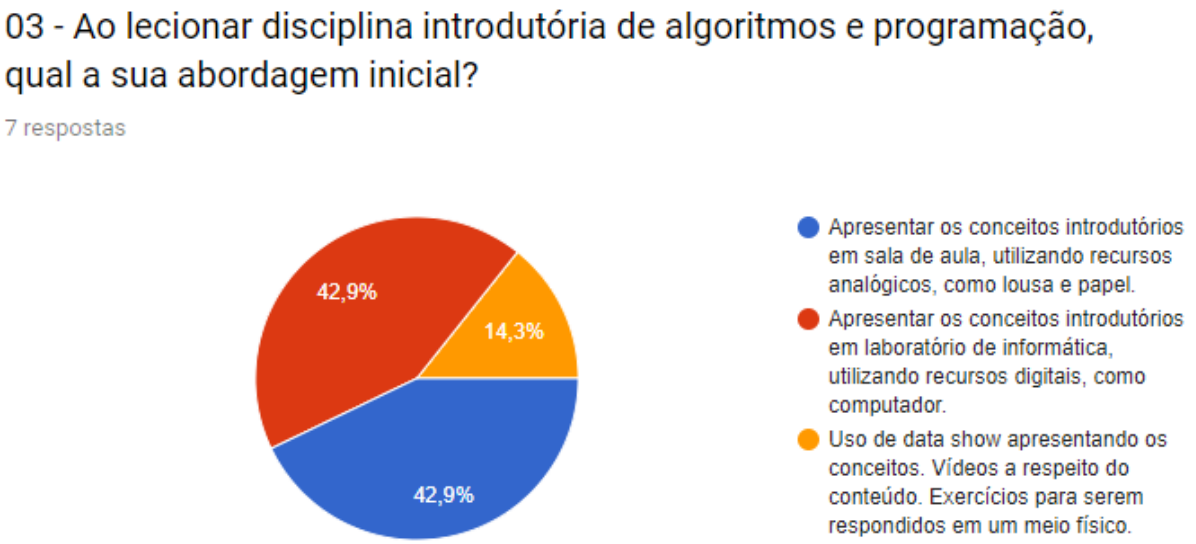
Fonte: Do autor (2019).

Enquanto 01 (um) professor ministrou disciplina para 06 (seis) turmas, o que corresponde a 14,3% dos respondentes, o restante dos professores – total de 85,7% - lecionou algoritmos e programação de computadores para quantitativo variável de 10 a 15 turmas.

Neste cenário, foi possível perceber que os respondentes, em geral, possuem vasta experiência no que tange à prática docente de algoritmos e programação e, desta forma, configuraram amostra adequada para o fornecimento de informações sobre o processo de ensino-aprendizagem.

Na terceira questão, o objetivo foi evidenciar a abordagem aplicada pelos professores para introduzir o conteúdo de algoritmos e programação. O respondente pôde selecionar entre três opções: abordagem em sala de aula, abordagem em laboratório de informática e outros (mediante justificativa desta). No Gráfico 03 abaixo, são exibidas as respostas obtidas.

Gráfico 03 – Abordagem introdutória adotada pelos respondentes



Fonte: Do autor (2019).

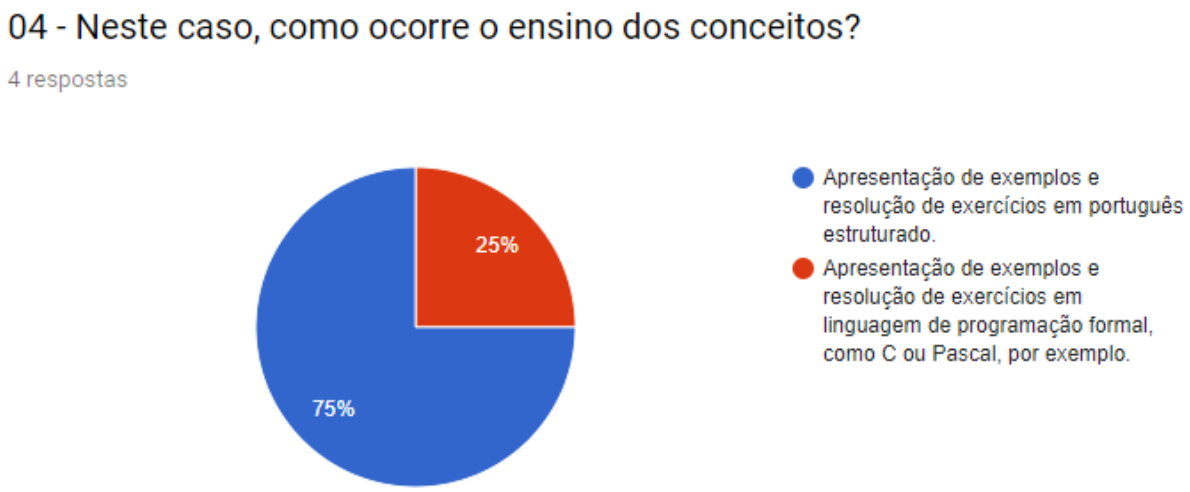
A respeito das abordagens, obteve-se igual percentual (42,9%) de introdução do conteúdo em sala de aula e em laboratório de informática. Complementarmente, 01 (um) professor - 14,3% - detalhou outra abordagem utilizada, sendo esta uma mescla de aplicação de recursos digitais e analógicos.

A partir de uma análise primária do resultado, verificou-se equilíbrio na adoção de abordagens em sala de aula e laboratório, com ausência de aplicação maciça de apenas uma destas. Entretanto, ao analisar a abordagem "mista" exemplificada por um dos respondentes, verificou-se que, mesmo com a utilização de recursos digitais – como computador e projetor -, o ensino é realizado em sala de aula e os exercícios resolvidos em meio físico, como papel. A partir desta ponderação, foi possível considerar que esta forma de ensino corresponde à abordagem inicial em sala de aula. Apesar de não ter sido evidenciado o porquê cada professor adota uma ou outra abordagem, pôde-se inferir alguns dos motivos: o ritmo de aprendizado; o nível de conhecimento da turma; a disponibilidade de recursos para aplicação e a preferência do professor.

Neste sentido, esta questão serviu de base para a elaboração da quarta questão. A partir da abordagem selecionada na pergunta anterior, o professor foi indagado sobre a estratégia adotada para ensinar os conceitos.

O Gráfico 04 revela a estratégia dos adeptos do ensino introdutório em sala de aula.

Gráfico 04 – Estratégia de ensino adotada em sala de aula



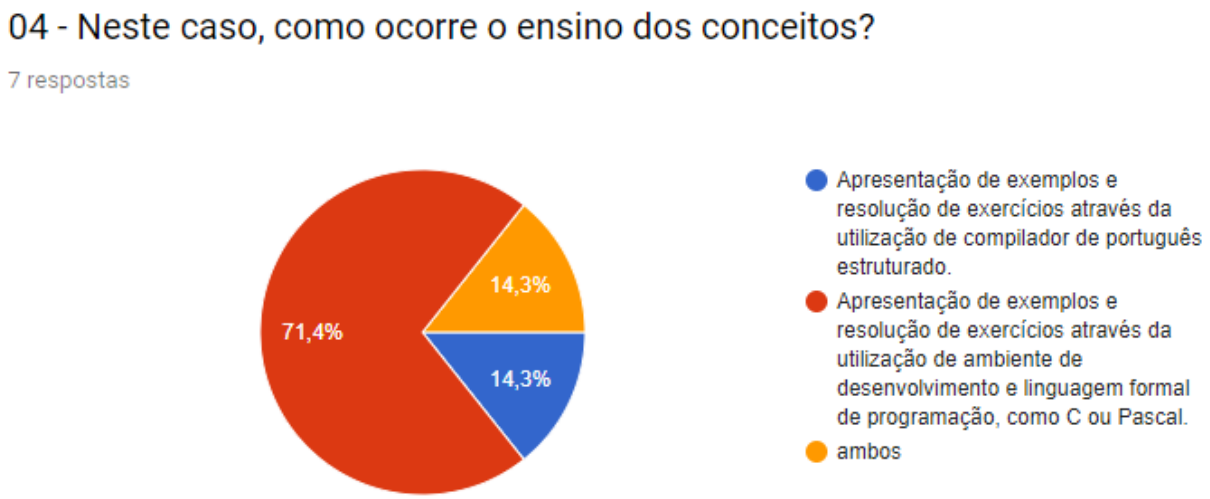
Fonte: Do autor (2019).

Para este grupo de docentes, a apresentação de exemplos e a resolução de exercícios em português estruturado tem sido a estratégia dominante, com adesão de 75%. Os 25% restantes já aplicaram e/ou aplicam em sala de aula alguma linguagem de programação, como o C ou o Pascal.

Assim, deduziu-se a existência de preocupação, por parte dos professores, em direcionar o aluno à compreensão do problema e ao desenvolvimento de meios de resolução isolados de detalhes puramente técnicos, o que os levou e/ou leva a recorrer ao uso do português estruturado. Ainda que os 25% restante dos docentes também tenha essa preocupação, estes consideram que o ensino dos conceitos básicos deve ocorrer diretamente em uma linguagem de programação.

Em relação ao ensino introdutório em laboratório de informática, o Gráfico 05 revela os resultados obtidos.

Gráfico 05 – Estratégia de ensino adotada em laboratório de informática



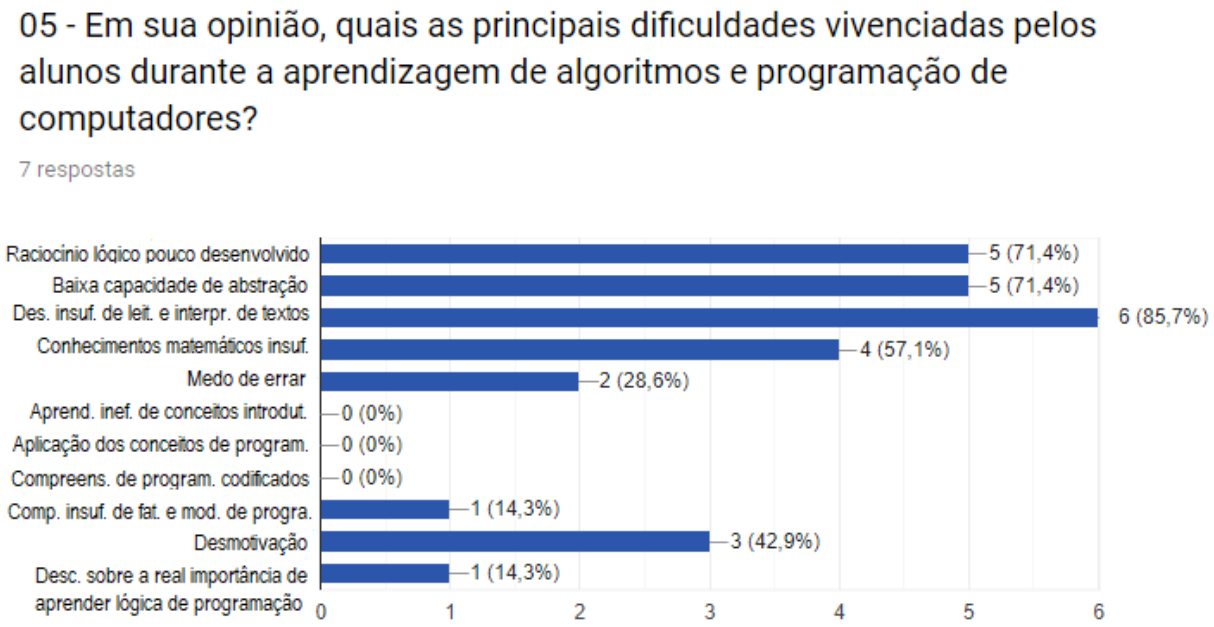
Fonte: Do autor (2019).

Já para os adeptos do ensino introdutório em laboratório de informática, a apresentação de exemplos e a resolução de exercícios em ambiente de desenvolvimento com o uso de linguagem de programação tem sido predominante – 71,4%. Por sua vez, a utilização de compilador de portugol foi e/ou é adotada por 14,3% dos professores, enquanto que os 14,3% restantes fizeram e/ou fazem uso de ambas as estratégias.

Para a estratégia dominante, inferiu-se que o docente apresentou e/ou apresenta o conceito e em seguida já o demonstrou e/ou demonstra em solução codificada, de modo a tornar o aluno apto a solucionar problemas diretamente no computador através de alguma linguagem de programação. Quando do uso de compilador de portugol, deduziu-se que a intenção do professor foi e/ou é familiarizar o aluno com o ambiente computacional ao mesmo tempo em que aplicou e/ou aplica os conceitos ensinados. No caso de recorrência a ambas as estratégias, a ideia é de que o professor iniciou e/ou inicie pelo compilador de portugol e em momento adequado migrou e/ou migre para alguma linguagem de programação formal.

A partir da identificação de algumas dificuldades vivenciadas pelos alunos no processo de ensino-aprendizagem de algoritmos e programação, a quinta pergunta centrou no levantamento das dificuldades percebidas pelos professores nas turmas lecionadas. Assim, foram colocadas à disposição para múltipla seleção um conjunto pré-definido de dificuldades e também a opção “Outras”, de modo que o docente informasse alguma outra não contemplada dentre as opções. O Gráfico 06 abaixo ilustra a percepção dos professores acerca da temática.

Gráfico 06 – Dificuldades dos alunos percebidas pelos respondentes



Fonte: Do autor (2019).

A principal dificuldade foi e/ou é o desenvolvimento insuficiente de leitura e interpretação de textos, apontada por 85,7% dos docentes. Logo em seguida, com 71,4% de indicação, foram identificados o raciocínio lógico pouco desenvolvido e a baixa capacidade de abstração. Abaixo destas, foram também elencadas: os conhecimentos matemáticos insuficientes (57,1%), a desmotivação (42,9%), o medo de errar (28,6%), a compreensão insuficiente sobre fatoração e modularização de programas e o desconhecimento sobre a real necessidade de aprender lógica de programação, cada qual com 14,3%. A aprendizagem ineficiente de conceitos introdutórios, a aplicação dos conceitos de programação e a compreensão de programas codificados não foram apontadas pelos respondentes. Este cenário apresentado pelos professores possibilitou tecer considerações interessantes sobre o panorama das dificuldades apresentadas pelos estudantes.

Em primeiro lugar, a elevada indicação acerca da leitura e interpretação de textos expôs uma realidade na qual a dificuldade em estruturar soluções algorítmicas reside nas etapas iniciais da resolução de um problema. É inegável a impossibilidade de o aluno elaborar estratégias pelas quais seja possível propor uma solução visto que, somente através da leitura e interpretação do mesmo, é possível extrair dados os quais, devidamente processados, conduzam à(s) saída(s) esperada(s).

O raciocínio lógico pouco desenvolvido e a baixa capacidade de abstração já eram esperados como sendo dificuldades marcantes no processo de ensino-aprendizagem de algoritmos e programação. O referencial teórico que sustenta este trabalho já evidenciou a manifestação destas e aqui obteve-se nova confirmação. Embora as habilidades de raciocínio lógico e abstração sejam aprimoradas ao longo da graduação, é fundamental que estas sejam moldadas nas fases antecedentes ao ensino superior nos diversos meios pelos quais o estudante transita durante seu amadurecimento pessoal e estudantil. A evolução dos alunos em direção à formação de acadêmicos e profissionais qualificados exige dos mesmos a capacidade de pensar de forma lógica, estruturada e, por vezes, isolada de aspectos não fundamentais à resolução de um problema.

A respeito dos conhecimentos matemáticos insuficientes, esta dificuldade foi e/ou é recorrente sobretudo nas etapas iniciais do aprendizado, quando o aluno é introduzido a alguns conceitos introdutórios de algoritmos e programação. Como exemplos, podem ser destacados os tipos de dados, as operações aritméticas, relacionais e lógicas e o incremento e o decremento de variáveis. Mediante a insuficiente compreensão destes conceitos, as dificuldades tornam-se ainda mais evidentes nos momentos em que é necessário combinar elementos para a resolução de exercícios oriundos do universo da Matemática, como problemas relacionados a cálculos financeiros, espaciais ou outros que exijam aplicação algorítmica ou codificada de elementos matemáticos.

A desmotivação dos alunos, embora seja entendida pelos professores como uma dificuldade face à missão de ensinar, também deve ser compreendida como resultado de um contexto multifacetado no qual o aluno está inserido. Este contexto pode ser composto por uma ou mais dificuldades acima mencionadas ou ilustradas nos parágrafos seguintes, pela abordagem dos professores, pelas ferramentas utilizadas, pelo ambiente universitário – pessoas e espaço em si – e outras não diretamente relacionadas à graduação, tais como aspectos sociais, econômicos, psicológicos e afins, não inclusos no escopo deste trabalho. Devido a esta complexidade, faz-se necessário ao professor, a partir da manifestação de desmotivação (individual ou coletiva), investigar as causas em conjunto com os demais participantes do processo de ensino-aprendizagem, tais como o próprio aluno, seus familiares, demais estudantes e corpo acadêmico, a fim de promover um ensino dotado de real significado, tanto para quem aprende quanto para quem ensina.

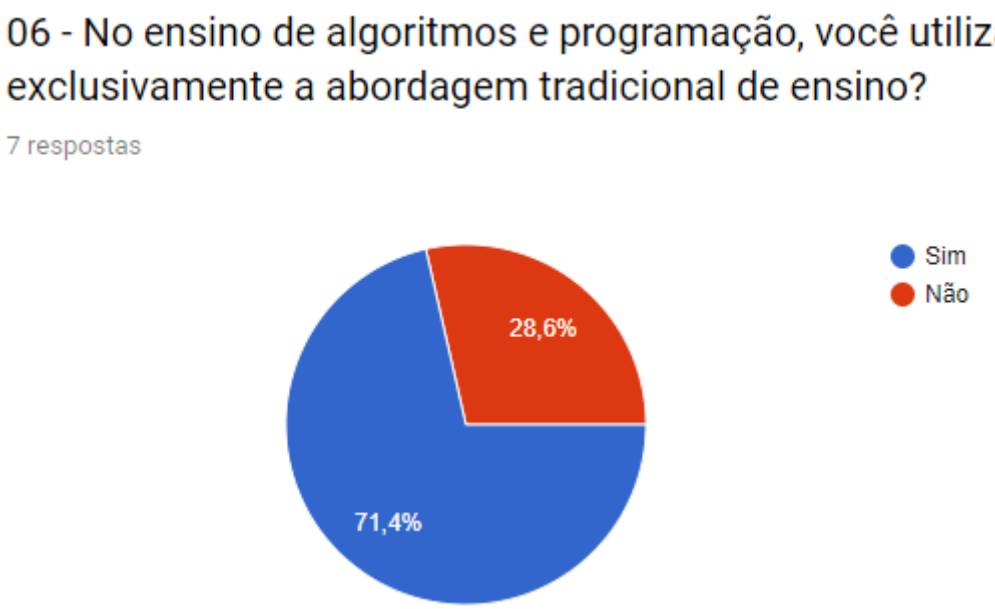
Apesar da tímida abordagem em literatura, o medo de errar foi e/ou é outra dificuldade observada pelos professores. É natural, sobretudo nas fases iniciais do aprendizado, a

ocorrência de erros nas diversas etapas que compõem o processo de construção de soluções, desde a visualização do problema até a obtenção da solução correta. Independente da natureza semântica ou sintática do erro, os estudantes sentem-se desmotivados e/ou incapazes, principalmente em situações nas quais manifestam-se múltiplos erros. Entretanto, os estudantes precisam compreender que estes equívocos tendem a ser menos frequentes à medida em que é praticada a resolução de exercícios e obtida consequente experiência. Aos professores, cabe orientar os alunos em duas frentes: subsidiar a correção dos equívocos e desestimular a proposição de soluções – algorítmicas e/ou codificadas – isentas de planejamento prévio, as quais tendem a acarretar erros sintáticos e semânticos.

Embora tenham tido menor apontamento dos docentes, a compreensão insuficiente sobre fatoração e modularização de programas e o desconhecimento sobre a real necessidade de aprender lógica de programação também compuseram e/ou compõem o leque de dificuldades vivenciadas pelos alunos. Para ambas, é interessante confrontá-las ao perfil de inexperiência dos alunos com o universo do algoritmo e da programação de computadores no início da graduação. Durante a trajetória acadêmica, sobremaneira no início desta fase, é importante que o aluno seja conscientizado sobre a importância de aprimorar as soluções construídas e dividir a solução em partes menores e bem definidas para estruturar o código e torná-lo reutilizável. Assim, o reconhecimento da importância das disciplinas introdutórias deve ser consequência de um ensino o qual permita ao aluno compreender, mediante teoria e prática, a necessidade de aprender gradualmente os conceitos introdutórios de algoritmos e programação para então obter êxito no percurso acadêmico e profissional.

Adiante, na sexta questão foi verificado o grau de aplicação da abordagem tradicional de ensino de algoritmos e programação. Aos respondentes, foi fornecida a conceituação desta abordagem, como sendo “aquela constituída por: aula expositiva de conteúdo, resolução de exercícios, trabalhos – individuais e/ou em grupo – e provas – no papel e/ou computador”. O resultado deste questionamento é exibido no Gráfico 07 abaixo.

Gráfico 07 – Uso de abordagem tradicional de ensino pelos respondentes



Fonte: Do autor (2019).

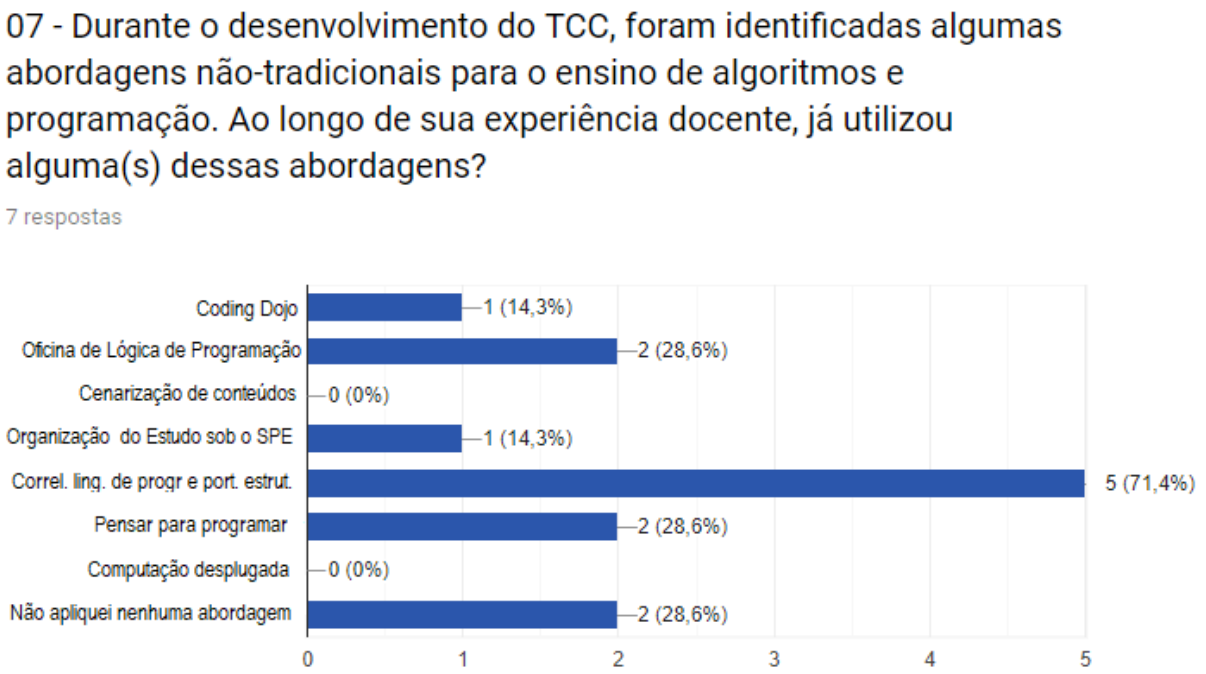
Através do resultado, verificou-se que a abordagem tradicional foi e/ou é aplicada por 71,4% dos professores, enquanto que outras abordagens somente foram e/ou são utilizadas por 28,6% dos docentes.

Portanto, já consolidada no meio acadêmico face à recorrente aplicação, a abordagem tradicional foi e/ou é largamente utilizada no ensino introdutório como abordagem exclusiva. Paralelamente, tem sido pequena a adoção de outras abordagens as quais possam contribuir para um ensino-aprendizagem mais satisfatório. Os motivos pelos quais os professores não adotam outras estratégias, como por exemplo o uso de abordagens não tradicionais e ferramentas de apoio, estão ilustrados na análise da nona questão, abordada mais a frente neste texto.

A sétima questão disponibilizou aos professores uma relação de abordagens não tradicionais de ensino – todas citadas e detalhadas neste estudo -, de modo a averiguar junto aos respondentes quais destas já haviam sido e/ou são por eles aplicadas. O resultado do apontamento realizado pelos professores é mostrado no Gráfico 08 abaixo

.

Gráfico 08 – Abordagens não tradicionais aplicadas pelos respondentes



Fonte: Do autor (2019).

A partir das respostas, constatou-se que a correlação entre linguagem de programação e o português estruturado foi a abordagem não tradicional com maior incidência de aplicação, obtendo 71,4% de apontamento. Após esta, a Oficina de Lógica de Programação e o Pensar para programar apareceram cada qual com 28,6% de participação. O Coding Dojo e a Organização do Estudo sob o Sistema Personalizado de Ensino foram indicadas por 14,3% dos respondentes. A Cenarização de Conteúdos e a Computação desplugada não foram utilizadas por nenhum dos docentes envolvidos na pesquisa. Adicionalmente, 28,6% dos participantes relataram não ter aplicado quaisquer abordagens não-tradicionais em suas aulas. Neste cenário, algumas inferências foram possíveis.

Em primeiro lugar, a maior recorrência da Correlação entre linguagem de programação e o português estruturado pode se dar em função da facilidade de implementação da mesma. O docente, durante o ministrar das aulas, necessita apenas traçar o paralelo entre as duas formas de representação do conceito utilizado, não sendo necessário reorganizar a dinâmica de funcionamento das aulas. Outro fator que poderia justificar esta elevada indicação seria a ideia de que, ao utilizar-se do português estruturado nas etapas iniciais e então migrar posteriormente para a representação do problema em linguagem de programação formal, esclarecendo a “nova” representação dos conceitos, o professor já teria feito uso desta abordagem não tradicional.

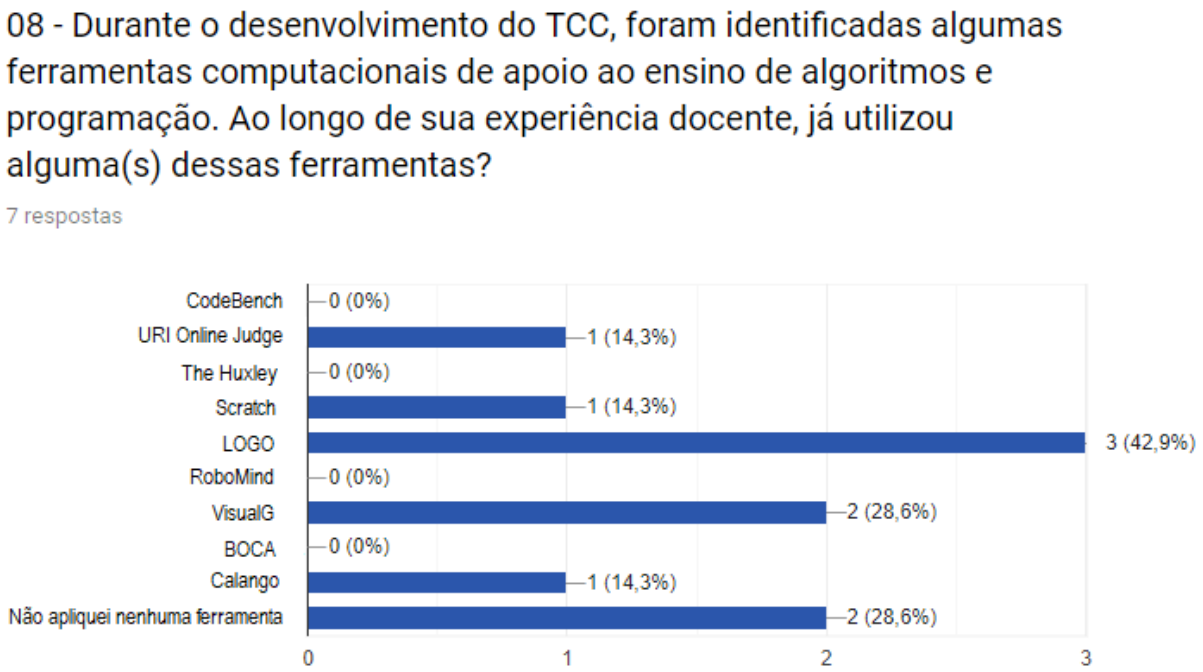
Em segundo lugar, o fato de nenhum professor ter recorrido à Cenarização de Conteúdos e à Computação desplugada poderia estar relacionado ao desconhecimento acerca destas abordagens. Neste caso, este trabalho pode atuar como meio de apresentação destas abordagens não tradicionais ao mesmo tempo que busca, a partir do levantamento das dificuldades dos alunos e professores, subsidiar a aplicação das mesmas em ambiente acadêmico.

Verificou-se também que 05 (cinco) dos 07 (sete) respondentes – 71,4% dos participantes - utilizaram ao menos 01 (uma) das abordagens não tradicionais evidenciadas neste estudo. Este resultado delineia contrastes com aquele obtido na sexta questão, através da qual 71,4% dos docentes apontaram uso exclusivo da abordagem tradicional durante as aulas. Ponderações sobre este contraste poderiam residir no fato destas abordagens não tradicionais terem sido introduzidas de maneira tímida no decorrer das aulas e, consequentemente, acarretarem na percepção de que apenas a abordagem tradicional havia sido utilizada, com estes métodos não tradicionais "diluídos" no processo de ensino-aprendizagem. Outra consideração possível poderia ser a disposição das perguntas do questionário. Quando afirmaram utilizar apenas a abordagem tradicional, os respondentes não tinham ciência das abordagens não tradicionais levantadas por este estudo e assim consideraram o escopo metodológico por eles utilizado como sendo "tradicional", mesmo com a definição daquela. Porém, ao serem questionados sobre este assunto nesta questão, avaliaram que em algum momento já fizeram uso de uma ou mais abordagens não tradicionais e então realizaram o apontamento.

De forma complementar à ideia acima, 28,6% dos participantes afirmaram não ter utilizado nenhuma abordagem não tradicional. Nestes casos, as possíveis justificativas podem ser semelhantes àquelas abordadas na nona questão, explicada posteriormente neste texto.

A oitava questão discorreu sobre a utilização de ferramentas computacionais de apoio ao processo de ensino-aprendizagem. Eventualmente, caso o docente tenha utilizado e/ou utilize outra(s) ferramenta(s) além das apresentadas, poderia informar qual(is) era(m) esta(s). No Gráfico 09 abaixo, encontra-se o resultado desta verificação.

Gráfico 09 – Ferramentas computacionais de apoio utilizadas pelos respondentes



Fonte: Do autor (2019).

A linguagem LOGO foi e/ou é empregada por 42,9% dos participantes. Logo em seguida, foram posicionadas o VisualG (28,6%), o URI Online Judge, o Scratch e o Calango (cada qual com 14,3%). O CodeBench, o The Huxley, o RoboMind e o BOCA não foram selecionados pelos docentes. Complementarmente, 28,6% dos participantes afirmaram não ter aplicado ferramentas de apoio durante as aulas.

A maior incidência da linguagem LOGO pode ser associada à sua longevidade e à variedade de implementações as quais utilizam esta linguagem como recurso de codificação. Por esta ideia, compreendeu-se que, mesmo com o progresso tecnológico e a evolução dos softwares classificados como educacionais, o poder didático da linguagem e de seu recurso visual contribuem para a "reciclagem" destas características em novos softwares.

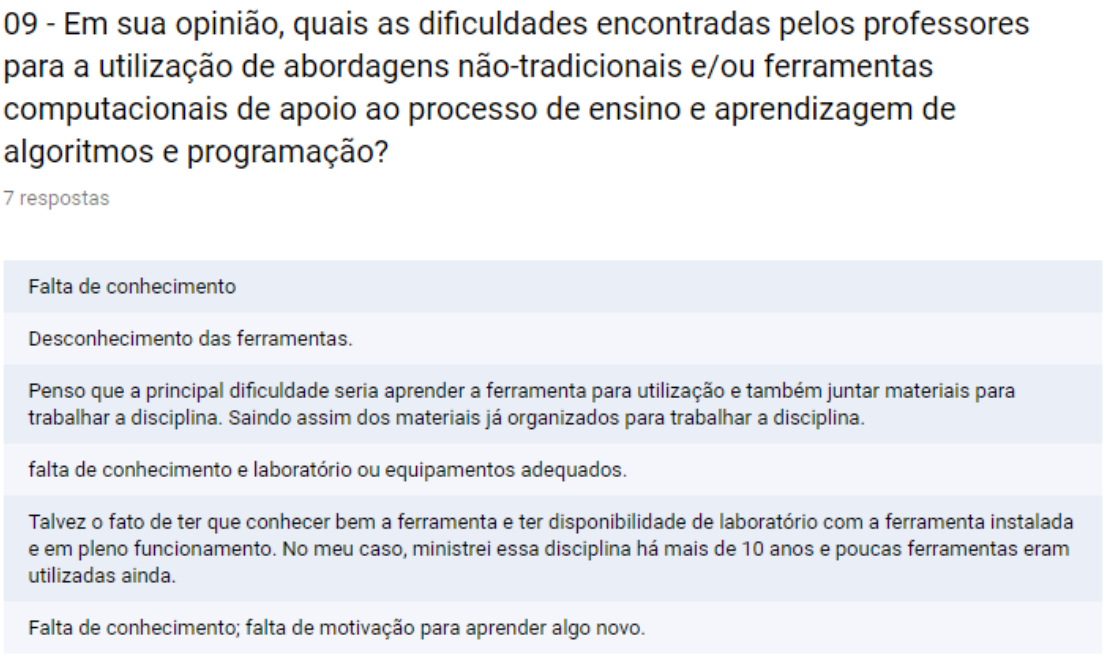
O resultado deste questionamento revelou a aplicação de todas as três categorias de ferramentas de apoio discutidas neste estudo – juiz online, ambiente de desenvolvimento visual e compilador de portugol. Ainda que os docentes possam ter feito /ou façam uso de alguma(s) dessas ferramentas isentos de detalhes sobre os benefícios por ela(s) proporcionados, é interessante avaliar que todos os três tipos já foram e/ou são aplicados em sala de aula com um ou mais objetivos.

O Boca e o Calango não foram citados no trabalho, entretanto foram incluídos no decorrer da elaboração do formulário para verificar se algum respondente utilizou e/ou utiliza alguma destas ferramentas. Assim, apesar de o BOCA não ter sido indicado, o Calango foi apontado por 01 (um) professor e, neste caso, poderia compor material de estudo para futuras investigações.

A respeito do posicionamento de alguns docentes em não ter feito e/ou não fazer uso de ao menos uma das ferramentas apresentadas, assim como não ter indicado a utilização de outra(s) não abordada(s) neste trabalho, foi possível ponderar que tal fato pode relacionar-se à exclusiva recorrência a ambientes de programação formal – aqueles cuja codificação é estruturada a partir de uma linguagem de programação formal. Outra ideia passível de reflexão é a utilização de ferramentas cujo nome não tenha sido recordado durante o responder do questionário.

Após tomar ciência do nível de aplicação das abordagens não tradicionais e das ferramentas computacionais de apoio ao processo de ensino-aprendizagem, fez-se necessário compreender as dificuldades enfrentadas pelo corpo docente no que tange à aplicação destes recursos nas turmas das disciplinas introdutórias de algoritmos e programação. A nona questão reuniu as dificuldades relatadas a partir da própria experiência docente, todas redigidas em texto livre e apresentadas na Figura 18 abaixo, sem edições.

Figura 18 – Dificuldades de aplicação relatadas pelos respondentes



Fonte: Do autor (2019).

Ao analisar o leque de dificuldades expostas pelos professores, percebeu-se que a falta de conhecimento foi e/ou é fator preponderante. Em maior ou menor grau, todos os relatos abordaram pouco ou nenhum conhecimento sobre as abordagens e/ou ferramentas. Este cenário pode estar associado, dentre outros fatores, à pequena divulgação dos instrumentos didáticos apresentados neste estudo. Esta pequena divulgação também impacta na dificuldade de reunir materiais adequados para trabalhar estes instrumentos no decorrer das aulas.

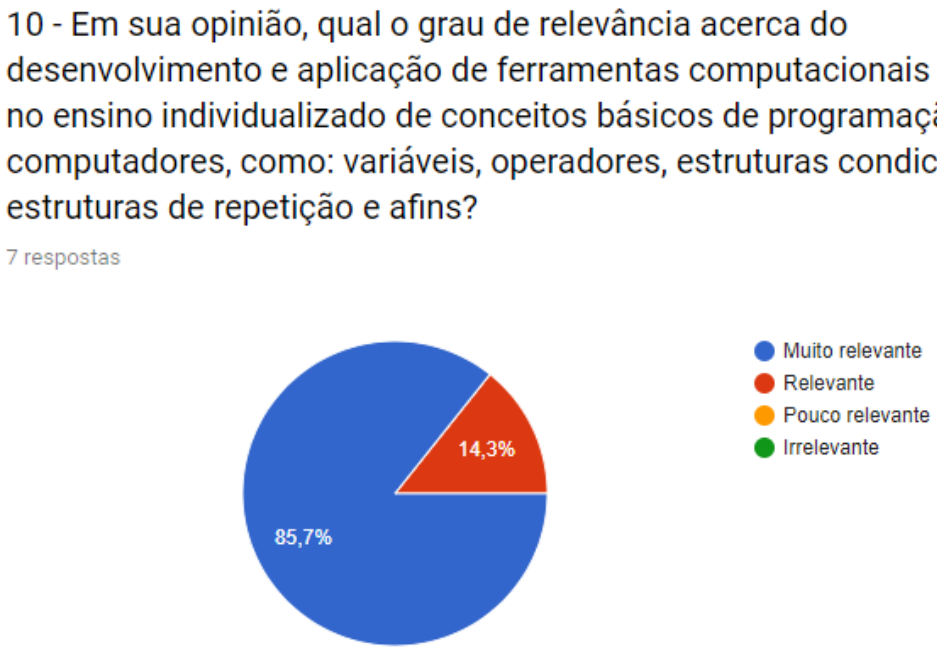
A disponibilidade de laboratórios e equipamentos adequados também foi apontada pelos docentes como limitador à aplicação de abordagens não tradicionais e ferramentas de apoio. Entretanto, o detalhamento das abordagens e ferramentas revela que as mesmas podem ser aplicadas em ambientes simplificados, isentos de robustez técnica. Ao analisar as abordagens, percebeu-se que estas apresentam atividades desenvolvidas em meios analógicos, de modo que os recursos digitais utilizados – computadores, internet e em alguns casos, projetores – para a resolução de problemas complementares às atividades analógicas são, em geral, simples e encontram-se disponíveis nos laboratórios de informática das IES. Sobre as ferramentas online, é necessária apenas a instalação de navegador e conexão à internet – eventualmente pode ser necessária a instalação de um *plugin*. Para as ferramentas offline, os requisitos necessários à instalação são baixos, a fim de possibilitar a aplicação destas em

múltiplos ambientes, com diferentes especificações técnicas. Detalhes sobre os procedimentos de instalação e configuração podem ser encontrados nos respectivos sites das ferramentas.

Ainda sobre as dificuldades, alguns respondentes também apontaram a falta de motivação em aprender algo novo, o que implica em dificuldades de aprender e explorar os instrumentos detalhados neste trabalho. Inseridos em um contexto no qual a abordagem tradicional de ensino é majoritariamente aplicada pelo corpo docente - em turmas com menor ou maior grau de dificuldade de aprendizado - e em que os alunos devem se adaptar ao padrão metodológico de ensino, é notável a atitude de parte dos professores em se manterem em zona de conforto. Nestes casos, é interessante o exercício da reflexão em avaliar, mesmo submetido a condições externas não controláveis – aspectos políticos, econômicos e sociais – a necessidade de explorar outros mecanismos através dos quais o processo de ensino-aprendizagem de algoritmos e programação seja frutífero aos alunos e aos professores.

Finalmente, a décima questão verificou junto aos professores a relevância em desenvolver e aplicar ferramentas computacionais voltadas ao ensino individualizado de conceitos introdutórios de algoritmos e programação de computadores. Ainda que não seja objetivo deste estudo propor o desenvolvimento de ferramentas, o mesmo poderá atuar como subsídio teórico face à proposição de implementação de tais ferramentas. A avaliação dos professores é exibida no Gráfico 10 abaixo.

Gráfico 10 – Relevância de ferramentas voltadas ao ensino de conceitos individualizados



Fonte: Do autor (2019).

Ao analisar o resultado, verificou-se que 85,7% dos professores consideraram muito relevante e 14,3% consideraram relevante o desenvolvimento e aplicação das ferramentas.

Por meio dos percentuais obtidos, foi evidenciado que os professores compreendem a importância do aprendizado dos conceitos introdutórios, os quais são fundamentais à construção de soluções para os problemas propostos nos meios acadêmico e profissional, independentemente da complexidade destes. Neste sentido, tais ferramentas poderão atuar como mecanismos de apoio ao ensino dos tópicos básicos de programação de computadores.

A partir do entendimento teórico e prático destes conceitos, o aluno torna-se mais independente na busca pela constante construção do próprio conhecimento, a qual é de extrema relevância no aprendizado de uma área volátil como a Computação.

CONCLUSÃO

O processo de ensino-aprendizagem de algoritmos e programação de computadores é notadamente permeado de dificuldades vivenciadas tanto pelos alunos quanto pelos professores e pela majoritária utilização da abordagem tradicional. Adicionalmente, foi possível constatar que a introdução ao universo do algoritmo e da programação de computadores tende a ocorrer somente durante a etapa inicial da graduação, de modo que o primeiro contato dos estudantes com este conteúdo fica reservado ao ambiente universitário.

As pesquisas bibliográfica e de campo conduzidas durante a execução deste trabalho reafirmaram a manifestação deste cenário, o qual, invariavelmente, contribui para acentuar as dificuldades e, consequentemente, reduzir o índice de aproveitamento do ensino.

Em síntese, a condução deste estudo constatou que, ao longo da jornada docente, os professores têm observado a manifestação de limitações e obstáculos ao aprendizado, no que tange à relação entre os estudantes e o conteúdo a ser repassado. Entretanto, ainda que cientes desta realidade, pouco tem sido feito para revertê-la quanto à utilização de abordagens não tradicionais e/ou ferramentas computacionais de apoio ao processo de ensino-aprendizagem. Conforme apontado pelos próprios professores, o principal fator para a manutenção deste cenário é a falta de conhecimento sobre recursos alternativos de ensino.

A partir desta percepção, concluiu-se que este trabalho logrou êxito tanto na identificação e discussão das dificuldades vivenciadas quanto na exposição de abordagens alternativas e ferramentas computacionais de apoio. O levantamento destas informações é importante na disseminação do conhecimento, através do qual pode ser possível reestruturar o método de ensino de algoritmos e programação de computadores, seja este realizado em sala de aula ou em laboratório de informática.

Ainda que o estudo tenha obtido sucesso quanto à sua finalidade, é importante ressaltar a necessidade de delimitação de seu escopo. Esta delimitação é fundamental para aprimorar a compreensão sobre o alcance do mesmo, assim como definir propósitos incluídos e excluídos no desenvolvimento do trabalho.

O presente trabalho não culminou na proposição explícita e no desenvolvimento de abordagens e/ou ferramentas computacionais voltadas a auxiliar o processo de ensino-aprendizagem de algoritmos e programação. O estudo em questão pode atuar como norteador para futuros desenvolvimentos de aplicações face à exploração do estado da arte quanto a métodos e/ou abordagens e ferramentas computacionais atualmente existentes.

Ainda quanto às limitações, os respondentes do questionário são todos docentes em IES do estado de Goiás e do Distrito Federal. Assim, mesmo que o estudo tenha reforçado o apontamento bibliográfico, é necessário ressaltar a restrição geográfica sob a qual este estudo foi amparado em parte de sua etapa investigativa.

Para trabalhos futuros, sugere-se a investigação de outras abordagens não tradicionais de ensino, de forma a enriquecer o leque de alternativas disponíveis. Ainda que este trabalho apresente diversas abordagens não tradicionais e ferramentas computacionais, é perceptível que existem outros recursos os quais podem ser abordados através de novos estudos.

Adicionalmente, podem ser desenvolvidos estudos para avaliar a aplicação de algum(as) abordagem(ns) e/ou ferramenta(s) computacional(is) em ambiente universitário e, a partir desta aplicação, analisar o real impacto proveniente da utilização de tais recursos em complementariedade à atual dinâmica de ensino.

Finalmente, o algoritmo e a programação de computadores são apenas um dos pilares sob os quais sustentam-se os cursos de graduação em Computação. Desta forma, estudos futuros podem ser desenvolvidos para investigar a manifestação de dificuldades em outras disciplinas e, a partir daquelas, evidenciar recursos os quais possam ser aplicados a fim de promover um ensino mais significativo e, consequentemente, conduzir à excelência acadêmica e profissional da graduação em Computação.

REFERÊNCIAS

AMARAL, Érico; *et al*. ALGO+ Uma ferramenta para o apoio ao ensino de Algoritmos e Programação para alunos iniciantes. **Anais do XXVIII Simpósio Brasileiro de Informática na Educação (SBIE 2017)**, 2017, 10 p. Disponível em: <http://br-ie.org/pub/index.php/sbie/article/view/7699/5494>. Acesso em: 10 mar. 2019.

AMBRÓSIO, Ana Paula L.; *et al*. Programação de computadores: compreender as dificuldades de aprendizagem dos alunos. **Revista Galego-Portuguesa de Psicoloxía e Educación**, v. 19, 2011, p. 185 – 197. Disponível em: < https://repositorio.bc.ufg.br/bitstream/ri/14795/5/Artigo%20-%20Ana%20Paula%20Laboissière%20Ambrósio%20-%202011.pdf>. Acesso em: 20 abr. 2019.

ANIDO, Ricardo. **Saci – ainda outro ambiente para o ensino de programação**. Instituto de Computação da Universidade Estadual de Campinas. Campinas, 2014, 10 p. Disponível em: < http://www.lbd.dcc.ufmg.br/colecoes/wei/2015/024.pdf >. Acesso em: 01 jun. 2019.

BARBOSA, Leônidas S.; FERNANDES, Teresa C. B. B.; CAMPOS, André M. C. Takkou: Uma Ferramenta Proposta ao Ensino de Algoritmos. **Anais do XXXI Congresso da Sociedade Brasileira de Computação**, 2011, 10 p. Disponível em: <https://dimap.ufrn.br/csbc2011/anais/eventos/contents/WEI/Wei_Secao_1_Artigo_3_Barbosa.pdf>. Acesso em: 10 fev. 2019.

BARCELOS, Thiago Schumacher; SILVEIRA, Ismar Frango. Teaching Computational Thinking in Initial Series: An Analysis of the Confluence among Mathematics and Computer Sciences in Elementary Education and its Implications for Higher Education. **Anais da XXXVIII Conferencia Latinoamericana en Informática**. Medellin, 2012, 08 p. Disponível em: < http://www.computacional.com.br/files/Gerais/BARCELOS%20-%20Teaching%20Computational%20Thinking%20in%20Initial%20Series.pdf >. Acesso em: 14 abr. 2019.

BELL, Tim; WITTEN, Ian H.; FELLOWS, Mike. **Computer Science Unplugged: Ensinando Ciência da Computação sem o uso do computador**. Adaptação de Robyn Adams e Jane McKenzie. Tradução de Luciano Porto Barreto. Salvador, 2011. Disponível em: <https://classic.csunplugged.org/wp-content/uploads/2014/12/CSUnpluggedTeachers-portuguese-brazil-feb-2011.pdf>. Acesso em: 11 mai. 2019.

BERSSANETTE, João Henrique; FRANCISCO, Antonio Carlos de; BARAN, Leandro Roberto. Espaços Ampliados Apoiados Por Tecnologias Digitais Para o Ensino de Programação de Computadores. **Revista Pleiade**, v. 12, n. 25, 2018, p. 39–51. Disponível em: <https://pleiade.uniamerica.br/index.php/pleiade/article/view/448>. Acesso em: 04 mai. 2019.

BUENO, Francisco da Silveira. **Minidicionário da Língua Portuguesa**. 2. ed. São Paulo: FTD, 2007.

CARVALHO, Leandro S. G.; OLIVEIRA, David B. F.; GADELHA, Bruno F. Juiz online como ferramenta de apoio a uma metodologia de ensino híbrido em programação. **Anais do**

XXVII Simpósio Brasileiro de Informática na Educação. Uberlândia, 2016, 10 p. Disponível em: <http://www.br-ie.org/pub/index.php/sbie/article/view/6694/4582>. Acesso em: 18 mai. 2019.

CAVALCANTE, Maria Cristina Tenório Cabral. Correlatos para Iniciantes em Programação. In: **Uma Modelagem Diagnóstica Multidimensional para o Entendimento do Perfil de Alunos Iniciantes em Programação**. Maceió, 2013. Disponível em: <http://www.repositorio.ufal.br/bitstream/riufal/1775/1/Uma%20modelagem%20diagnóstica%20multidimensional%20para%20o%20entendimento%20do%20perfil%20de%20alunos%20iniciantes%20em%20programação.pdf>. Acesso em: 16 abr. 2019.

COSTA, Túlio Henriques. **Análise dos problemas enfrentados por alunos de Programação**. Universidade Estadual da Paraíba, 2013. Disponível em: <http://dspace.bc.uepb.edu.br/jspui/bitstream/123456789/2312/1/PDF%20-%20Túlio%20Henriques%20Costa.pdf>. Acesso em: 19 abr. 2019.

DAGOSTINI, Jéssica; *et al*. URI Online Judge Blocks: Construindo Soluções em uma Plataforma Online de Programação. **Anais do XXIX Simpósio Brasileiro de Informática na Educação (SBIE 2018)**, 2018, 10 p. Disponível em: <http://www.br-ie.org/pub/index.php/sbie/article/view/7969/5667>. Acesso em: 28 mai. 2019.

DANTAS, Sormany Silva; *et al*. Avaliação de um software educacional de apoio à aprendizagem de programação: VisuAlg. **Anais do VII Congresso Norte Nordeste de Pesquisa e Inovação - CONNEPI**. Palmas, 2012, 09 p. Disponível em: <http://propi.ifto.edu.br/ocs/index.php/connepi/vii/paper/viewFile/1295/923>. Acesso em: 02 jun. 2019.

FALCKEMBACH, Gilse A. Morgental; ARAUJO, Fabrício Viero de. Aprendizagem de algoritmos: dificuldades na resolução de problemas. **Anais do Congresso Sul Brasileiro de Computação**, v. 2, 2006, 07 p. Disponível em: <http://periodicos.unesc.net/sulcomp/article/view/916/909>. Acesso em: 04 mar. 2019.

FERREIRA, Larissa Karollyne de Melo. Coding Dojo: O que é? Pra que serve? Como funciona? **Medium Corporation**, 2018. Disponível em: < https://medium.com/@lkmf/coding-dojo-o-que-é-pra-quê-serve-como-funciona-a84d333ea031>. Acesso em: 28 abr. 2019.

FRANZ, Luiz Paulo; SILVA, João Pablo Silva da; CHEIRAN, Jean Felipe Patikowski. O uso de Coding Dojo no aprendizado colaborativo de programação de computadores. **Revista Novas Tecnologias na Educação**, v. 12, n. 2, 2014, 09 p. Disponível em: < https://www.seer.ufrgs.br/renote/article/view/53541/33046>. Acesso em: 27 abr. 2019.

GOMES, Marina; *et al*. Um estudo sobre erros em Programação: reconhecendo as dificuldades de programadores iniciantes. **Anais dos Workshops do IV Congresso Brasileiro de Informática na Educação (CBIE 2015)**. Alagoas, 2015, 10 p. Disponível em: <http://br-ie.org/pub/index.php/wcbie/article/view/6317/4426>. Acesso em: 06 abr. 2019.

GOMES, Tancicleide; MELO, Jeane de. Jogos Digitais no Ensino de Programação. **Anais do Congresso Regional sobe Tecnologias na Educação**, 2016, 08 p. Disponível em: <http://ceur-ws.org/Vol-1667/Minicurso05.pdf>. Acesso em: 27 abr. 2019.

HENRIQUE, Mychelline Souto; TEDESCO, Patrícia C. de A. R. Uma Revisão Sistemática da Literatura sobre conhecimentos, habilidades, atitudes e competências desejáveis para auxiliar a aprendizagem de programação. **Anais dos Workshops do VI Congresso Brasileiro de Informática na Educação (WCBIE 2017)**, 2017, 10 p. Disponível em: < http://www.br-ie.org/pub/index.php/wcbie/article/view/7505/5300>. Acesso em: 07 abr. 2019.

HOLANDA, Wallace Duarte de; COUTINHO, Jarbele Cássia da Silva; FONTES, Laysa Mabel de Oliveira. Uma Intervenção Metodológica para Auxiliar a Aprendizagem de Programação Introdutória: um estudo experimental. **Anais dos Workshops do VII Congresso Brasileiro de Informática na Educação (WCBIE 2018).** Fortaleza, 2018, 10 p. Disponível em: <http://www.br-ie.org/pub/index.php/wcbie/article/view/8292/5969>. Acesso em: 06 abr. 2019.

INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA. PNAD Contínua TIC 2017: Internet chega a três em cada quatro domicílios do país. **Agência de Notícias do IBGE**. Rio de Janeiro, 2018. Disponível em: < https://agenciadenoticias.ibge.gov.br/agencia-sala-de-imprensa/2013-agencia-de-noticias/releases/23445-pnad-continua-tic-2017-internet-chega-a-tres-em-cada-quatro-domicilios-do-pais >. Acesso em: 18 mai. 2019.

INSTITUTO DE COMPUTAÇÃO DA UFAM. Screenshots das Interfaces de Aluno e Professor. **CodeBench**. [201-?]. Disponível em: < http://codebench.icomp.ufam.edu.br/index.php?r=site%2Fscreenshots >. Acesso em: 02 jun. 2019.

LABORATÓRIO DE INOVAÇÃO TECNOLÓGICA NA EDUCAÇÃO. Portugol Studio: Uma ferramenta para aprender programação. **LITE**. Disponível em: < http://lite.acad.univali.br/portugol/ >. Acesso em: 01 jun. 2019.

LEITE, Vanessa Matias; *et al*. VisuAlg: Estudo de Caso e Análise de Compilador destinado ao ensino de Programação. **Anais do Nuevas Ideas en Informática Educativa - TISE 2013.** Porto Alegre, 2013, 04 p. Disponível em: < http://www.tise.cl/volumen9/TISE2013/637-640.pdf>. Acesso em: 01 jun. 2019.

LIMA, Anderson C.; *et al*. Uma Oficina para Ensino de Algoritmos Paralelos por Meio de Computação Desplugada. **Anais dos Workshops do VII Congresso Brasileiro de Informática na Educação (WCBIE 2018)**, Ponta Porã, 2018, 10 p. Disponível em: < http://br-ie.org/pub/index.php/wcbie/article/view/8304/5981>. Acesso em: 05 mai. 2019.

LIMA JUNIOR, José Augusto Teixeira de; VIEIRA, Carlos Eduardo Costa; VIEIRA, Priscila de Paula. Dificuldades no processo de aprendizagem de Algoritmos: uma análise dos resultados na disciplina de AL1 do Curso de Sistemas de Informação da FAETERJ - Campus Paracambi. **Cadernos UniFOA**, n. 27**.** Volta Redonda, 2015, 10 p. Disponível em: <http://revistas.unifoa.edu.br/index.php/cadernos/article/view/293/346>. Acesso em: 07 abr. 2019.

MACP. **Sobre o projeto Compilador Portugol**, [201-]. Disponível em: <http://portugol.sourceforge.net>. Acesso em: 28 mai. 2019.

MAGALHÃES, Regis Pires. **Introdução à Lógica**. Piauí, 2007. Disponível em: <https://pt.slideshare.net/regispires/logica-algoritmo-02-algoritmo-presentation>. Acesso em: 14 abr. 2019.

MARCOLINO, Anderson S.; BARBOSA, Ellen Francine. Softwares Educacionais para o Ensino de Programação: Um Mapeamento Sistemático. **Anais do XXVI Simpósio Brasileiro de Informática na Educação (SBIE 2015)**, 2015, 10p. Disponível em: <http://www.br-ie.org/pub/index.php/sbie/article/view/5150/3541>. Acesso em: 04 mar. 2019.

MARQUES, Wagner dos Santos; SOUZA, Paulo Silas Severo de; MOMBACH, Jaline Gonçalves. Pensar para Programar: Projeto de Ensino no Curso Técnico em Informática. **Anais do XXXVII Congresso da Sociedade Brasileira de Computação**. Alegrete, 2017, p. 2110–2119. Disponível em: < http://portaldeconteudo.sbc.org.br/index.php/wei/article/view/3550>. Acesso em: 05 mai. 2019.

MEDEIROS, Tainá Jesus; SILVA, Thiago Reis da; ARANHA, Eduardo Henrique da Silva. Ensino de programação utilizando jogos digitais: uma revisão sistemática da literatura. **Revista Novas Tecnologias na Educação**, v. 11, 2013, 10 p. Disponível em: <https://seer.ufrgs.br/renote/article/view/44363/28025>. Acesso em: 05 mar. 2019.

MINISTÉRIO DA EDUCAÇÃO. As tecnologias digitais e a computação. In: **Base Nacional Comum Curricular**: educação é a base. Brasília, 2018, p. 473-475. Disponível em: < http://basenacionalcomum.mec.gov.br/images/BNCC_EI_EF_110518_versaofinal_site.pdf>. Acesso em: 16 abr. 2019.

MINISTÉRIO DA EDUCAÇÃO. CNE debate incluir computação na Base Nacional Comum Curricular. **Portal do Governo do Brasil**. Brasília, jul. 2018. Disponível em: <http://portal.mec.gov.br/component/content/article/211-noticias/218175739/66811-cne-debate-incluir-computacao-na-base-nacional-comum-curricular?Itemid=164>. Acesso em: 16 abr. 2019.

MINISTÉRIO DA EDUCAÇÃO. Resolução CNE/CES 5/2016. **Diário Oficial da União**. Brasília, nov. 2016. Disponível em: <http://portal.mec.gov.br/index.php?option=com_docman&view=download&alias=52101-rces005-16-pdf&category_slug=novembro-2016-pdf&Itemid=30192>. Acesso em: 22 mar. 2019.

MOREIRA, Benjamin Grando. Melhores IDEs para Portugol. **Professor Benjamin**. 2015. Disponível em: < http://www.galirows.com.br/meublog/blog/melhores-portugol/>. Acesso em: 02 jun. 2019.

MOREIRA, Gabriel Luídy; *et al*. Desafios na aprendizagem de programação introdutória em cursos de TI da UFERSA, campus Pau dos Ferros: um estudo exploratório. **Anais do Encontro de Computação do Oeste Potiguar ECOP/UFERSA 2018**. Rio Grande do Norte,

2018, 07 p. Disponível em: < https://periodicos.ufersa.edu.br/index.php/ecop/article/view/7907 >. Acesso em: 06 abr. 2019.

MOTTA, Marcelo Souza; MIRANDA, Dimas Felipe de. **Geometria da Tartaruga**: SuperLogo. Centro Universitário Norte do Espírito Santo. São Mateus, 2018. Disponível em: < https://www.slideshare.net/elciellebonomo/superlogo-94025571 >. Acesso em: 02 jun. 2019.

NASCIMENTO, João Paulo Mendes do. **Um estudo sobre avaliação de Software para o auxílio no ensino de programação**. Orientador: Adriana Zenaide Clericuzi. Rio Tinto, 2016. Disponível em: < https://repositorio.ufpb.br/jspui/bitstream/123456789/3320/1/JPMN28112016.pdf >. Acesso em: 27 mai. 2019.

NOFITASARI, Anngun; YUANA, Rosihan. Ari; MARYONO, Dwi. The Use of Robomind Application in Problem Based Learning Model to Enhance the Student's Understanding on the Conceptual Programming Algorithm. **Indonesian Journal of Informatics Education (IJIE)**, v. 1, n. 1. Surakarta, 2017, 10 p. Disponível em: < https://jurnal.uns.ac.id/ijie/article/view/4170/pdf_1 >. Acesso em: 19 mai. 2019.

NOSCHANG, Luiz F; *et al*. Portugol Studio: Uma IDE para Iniciantes em Programação. **Anais do XXXIV Congresso da Sociedade Brasileira de Computação – CSBC 2014**. Brasília, 2014, 10 p. Disponível em: < http://www.lbd.dcc.ufmg.br/colecoes/wei/2014/001.pdf >. Acesso em: 01 jun. 2019.

OLIVEIRA, Manassés Vitorino de; RODRIGUES, Luciene Cavalcanti; QUEIROGA, Ana Paula Garrido de. Material didático lúdico: uso da ferramenta Scratch para auxílio no aprendizado de lógica da programação. **Anais do XXII Workshop de Informática na Escola (WIE 2016).** Votuporanga, 2016, 10 p. Disponível em: < http://www.br-ie.org/pub/index.php/wie/article/view/6842/4720 >. Acesso em: 19 mai. 2019.

OLIVEIRA, Millena Lauyse Silva de; *et al.* Ensino de lógica de programação no ensino fundamental utilizando o Scratch: um relato de experiência. **Anais do XXXIV Congresso da Sociedade Brasileira de Computação – CSBC 2014**. Brasília, 2014, 10 p. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/wei/2014/0022.pdf>. Acesso em: 14 abr. 2019.

ORGANIZATION CODING DOJO. **What Is Coding Dojo**. 2016. Disponível em: < http://codingdojo.org/WhatIsCodingDojo/>. Acesso em: 28 abr. 2019.

PAES, Rodrigo de Barros; *et al*. Ferramenta para a Avaliação de Aprendizado de Alunos em Programação de Computadores. **Anais do II Congresso Brasileiro de Informática na Educação (CBIE 2013)**. Campinas, 2013, 10 p. Disponível em: < http://www.br-ie.org/pub/index.php/wcbie/article/view/2669 >. Acesso em: 31 mai. 2019.

PRIESNITZ FILHO, Walter; ABEGG, Ilse; SIMONETTO, Eugênio de Oliveiro. Uma abordagem diferenciada no ensino de algoritmos através da utilização de uma lousa digital. **Revista Gestão, Inovação e Tecnologias**, v. 2, n. 2. São Cristóvão, 2012, p. 129–137. Disponível em: < http://www.revistageintec.net/index.php/revista/article/view/29/89>. Acesso em: 01 mai. 2019.

QUEIROZ, Marina O.; REBOUÇAS, Ayla Débora Dantas S. Neurociência e o ensino de programação: Uma revisão sistemática da literatura. **Anais do XXIX Simpósio Brasileiro de Informática na Educação (SBIE 2018)**. Ceará, 2018, 10 p. Disponível em: < http://www.br-ie.org/pub/index.php/sbie/article/view/8063/5754 >. Acesso em: 06 abr. 2019.

RAIOL, Alberto A. C.; *et al*. Resgatando a Linguagem de Programação Logo: Uma Experiência com Calouros no Ensino Superior. **Anais do XXIII Workshop sobre Educação em Computação**. Recife, 2015, 10 p. Disponível em: < https://www.researchgate.net/publication/281209934_Resgatando_a_Linguagem_de_Programacao_Logo_Uma_Experiencia_com_Calouros_no_Ensino_Superior >. Acesso em: 18 mai. 2019.

RIBEIRO, Leila; *et al*. Computational Thinking: Possibilities and Challenges. **Anais do II Workshop - Escola de Informática Teórica**. Rio Grande, 2013, 04 p. Disponível em: < https://ieeexplore.ieee.org/abstract/document/6778560>. Acesso em: 14 abr. 2019.

ROCHA, Paulo Santana; *et al*. Ensino e Aprendizagem de Programação: Análise da Aplicação de Proposta Metodológica Baseada no Sistema Personalizado de Ensino. **Revista Novas Tecnologias na Educação**, n. 3, v. 8, 2010, 11 p. Disponível em: < https://www.seer.ufrgs.br/renote/article/view/18061/10649>. Acesso em: 01 mai. 2019.

SALAZAR, Rafael; ODAKURA, Valguima; BARVINSKI, Carla. Scratch no ensino superior: motivação. **Anais do XXVI Simpósio Brasileiro de Informática na Educação (SBIE 2015)**. Maceió, 2015, 10 p. Disponível em: < http://www.br-ie.org/pub/index.php/sbie/article/view/5470/3829 >. Acesso em: 19 mai. 2019.

SALES, Chrystian Gesteira; DANTAS, Vanessa Faria. ProGame: um jogo para o ensino de algoritmos e programação. **Anais do Simpósio Brasileiro de Informática na Educação**, 2010, 04 p. Disponível em: <http://www.br-ie.org/pub/index.php/sbie/article/view/1558/1323>. Acesso em: 15 jan. 2019.

SANTIAGO, Almir D. V.; KRONBAUER, Artur H. Um Modelo Lúdico para o Ensino de Conceitos de Programação de Computadores. **Anais do XXVII Simpósio Brasileiro de Informática na Educação (SBIE 2016)**, 2016, 10 p. Disponível em: < http://br-ie.org/pub/index.php/sbie/article/view/6723/4610 >. Acesso em: 10 mar. 2019.

SANTOS, Rodrigo Pereira; COSTA, Heitor Augusto Xavier. Análise de Metodologias e Ambientes de Ensino para Algoritmos, Estruturas de Dados e Programação aos iniciantes em Computação e Informática. **INFOCOMP Journal of Computer Science**, v. 5, 2006, 10p. Disponível em: <www.dcc.ufla.br/infocomp/index.php/INFOCOMP/article/download/121/106/0>. Acesso em: 19 jan. 2019.

SILVA, Carlos Antônio da; SILVA, Leandro Dias da; MARTINS, João Carlos Diniz. Aplicação do The Huxley no ensino de programação para alunos do curso técnico em informática para internet. **Proceedings of SBGames 2018**. Foz do Iguaçu, 2018, 04 p. Disponível em: <

http://www.sbgames.org/sbgames2018/files/papers/EducacaoShort/188353.pdf>. Acesso em: 30 mai. 2019.

SILVA, Eralyson Galdino da; *et al*. Análise de ferramentas para o ensino de Computação na Educação Básica. **Anais do XXXIV Congresso da Sociedade Brasileira de Computação – CSBC 2014**. Brasília, 2014, 10 p. Disponível em: < https://s3.amazonaws.com/academia.edu.documents/34381015/Analise__e_ferramentas_para_o_ensino_de_Computacao_na_Educacao_Basica.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1548503586&Signature=kkHjcnG5lA%2F8UtKxYb4Zp2rSU4g%3D&response-content-disposition=inline%3B%20filename%3DAnalise_deferramentas_para_o_ensino_de.pdf>. Acesso em: 26 jan. 2019.

SILVA, Italo Fernandes Amorim da; SILVA, Ivanda Maria Martins; SANTOS, Marizete Silva. Análise de problemas e soluções aplicadas ao ensino de disciplinas introdutórias de programação. **IX Jornada de Ensino, Pesquisa e Extensão da UFRPE**. Pernambuco, 2009, 03 p. Disponível em: <http://www.eventosufrpe.com.br/jepex2009/cd/resumos/r1479-1.pdf>. Acesso em: 26 mar. 2019.

SILVA JUNIOR, Leonardo Ribeiro da. **Mineração de Dados Aplicada ao Estudo da Relação Entre as Reprovações nas Disciplinas de Lógica de Programação a Evasão no Curso de Sistemas de Informação do Campus Anápolis de Ciências Exatas e Tecnológicas - Henrique Santillo** / Orientador: Noeli Antônia Pimentel Vaz – Anápolis, 2018, 83 p.

SOUZA, Draylson Micael; BATISTA, Marisa Helena da Silva; BARBOSA, Ellen Francine. Problemas e Dificuldades no Ensino e na Aprendizagem de Programação: Um Mapeamento Sistemático. **Revista Brasileira de Informática na Educação**, v. 24, 2016, 15 p. Disponível em:<https://www.researchgate.net/profile/Ellen_Barbosa/publication/306299342_Problemas_e_Dificuldades_no_Ensino_de_Programacao_Um_Mapeamento_Sistematico/links/5818a12908aee7cdc685aac9/Problemas-e-Dificuldades-no-Ensino-de-Programacao-Um-Mapeamento-Sistematico.pdf>. Acesso em: 10 mar. 2019.

STADELHOFER, Luiza Engler; *et al*. Aplicação de um questionário com professores brasileiros para investigar as disciplinas de Algoritmos e Lógica de Programação para os diferentes cursos. **Anais do IX Computer on the Beach**. Florianópolis, 2018, 10 p. Disponível em: < https://www.researchgate.net/profile/Luiza_Engler_Stadelhofer/publication/325679633_Aplicacao_de_um_questionario_com_professores_brasileiros_para_investigar_as_disciplinas_de_Algoritmos_e_Logica_de_Programacao_para_os_diferentes_cursos/links/5b1dd1beaca272021cf57ce0/Aplicacao-de-um-questionario-com-professores-brasileiros-para-investigar-as-disciplinas-de-Algoritmos-e-Logica-de-Programacao-para-os-diferentes-cursos.pdf >. Acesso em: 06 abr. 2019.

TAYLOR, Kyle; SILVER, Laura. Smartphone Ownership Is Growing Rapidly Around the World, but Not Always Equally. **Pew Research Center**. Washington DC, 2019. Disponível em: < https://www.pewglobal.org/2019/02/05/smartphone-ownership-is-growing-rapidly-around-the-world-but-not-always-equally/ >. Acesso em: 18 mai. 2019.

THE HUXLEY. Home. **The Huxley**. [201-?]. Disponível em: < https://www.thehuxley.com >. Acesso em: 02 jun. 2019.

THE SCRATCH TEAM. 3 Things To Know About Scratch 3.0. **Medium**. 2018. Disponível em: < https://medium.com/scratchteam-blog/3-things-to-know-about-scratch-3-0-18ee2f564278>. Acesso em: 02 jun. 2019.

TRENTIN, Marco Antônio Sandini; *et al*. Scratch como estratégia de ensino de algoritmos. **VIII International Conference on Engineering and Computer Education**. Luanda, 2013, 05 p. Disponível em: < http://copec.eu/congresses/icece2013/proc/works/51.pdf >. Acesso em: 19 mai. 2019.

UNIVERSIDADE DE AMSTERDÃ. Documentation Overview. **RoboMind Desktop**, [201-?]. Disponível em: < https://www.robomind.net/en/docOverview.htm >. Acesso em: 19 mai. 2019.

UNIVERSIDADE FEDERAL DE LAVRAS. **O que é LOGO**. [200-?]. Disponível em: < http://algol.dcc.ufla.br/~bruno/wxlogo/docs/oquee.html>. Acesso em: 18 mai. 2019.

URI ONLINE JUDGE. Academic. **URI Online Judge: Problems & Contests**. [201-?]. Disponível em: <https://www.urionlinejudge.com.br/academic/>. Acesso em: 29 mai. 2019.

URI ONLINE JUDGE. Soma Simples. **URI Online Judge: Problems & Contests** [201-?]. Disponível em: < https://www.urionlinejudge.com.br/judge/pt/problems/view/1003 >. Acesso em: 02 jun. 2019.

VERGARA, Sylvia Constant. **Projetos e relatórios de pesquisa em administração**. 16. ed. São Paulo: Atlas, 2016, 97 p.

VISUALG. Sobre o VisuAlg. **VisuAlg: O melhor interpretador de algoritmos**. Disponível em: < http://visualg3.com.br/sobre-o-visualg/>. Acesso em: 01 jun. 2019.

XIMENES, Sérgio. **Dicionário da língua portuguesa**. 3 ed. São Paulo: Ediouro, 2001.

YUANA, Rosihan Ari; MARYONO, Dwi. Robomind Utilization to Improve Student Motivation and Concept in Learning Programming. **Proceeding of International Conference on Teacher Training and Education (ICTTE) FKIP UNS 2015,** v. 1, n. 1. Surakarta, 2016, 05 p. Disponível em: < http://jurnal.fkip.uns.ac.id/index.php/ictte/article/view/8439 >. Acesso em: 19 mai. 2019.

APÊNDICES

Apêndice A – Cronograma previsto e realizado

Nº	Atividade	Fev.		Mar.		Abr.		Mai.		Jun.		Jul.		Ago.		Set		Out.		Nov.		Dez.	
		1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2
01	Apresentação da ideia pelo orientador		X																				
02	1ª Reunião de TC		X																				
03	Assinatura e entrega do Termo de Aceite e Compromisso		X																				
04	Pesquisa e catalogação dos materiais bibliográficos		X	X	X	X	X	X	X	X	X	X	X										
05	Discussão sobre os tópicos componentes do Referencial Teórico		X	X	X	X	X	X	X														
06	Escrita do esboço da Introdução			X	X	X	X																
07	Escrita dos tópicos iniciais do Referencial Teórico			X	X	X	X																
08	Data limite para disponibilização do Termo de Aceite no Moodle				X																		
09	2ª Reunião do TC					X																	
10	Escrita do esboço do Desenho Teórico e Metodológico da Pesquisa					X	X																
11	Escrita dos tópicos finais do Referencial Teórico						X	X	X														

12	Entrega do TC1.N1						X																
13	Correção do TC1.N1							X															
14	3ª Reunião de TC								X														
15	Elaboração da lista de links das plataformas e ferramentas								X	X	X	X	X										
15	Elaboração do questionário aos docentes								X	X													
16	Entrega e postagem do TC1.N2									X													
17	Realização da banca de qualificação									X													
18	Produção do artigo como componente TC2.N1											X	X	X	X	X							
19	Submissão do questionário aos docentes													X									
20	Análise das respostas do questionário														X								
21	Escrita da Análise dos Resultados														X	X							
22	Envio da versão preliminar do artigo ao orientador para validação														X								
23	Postagem do artigo no Moodle														X								
24	4ª Reunião de TC														X								
25	Entrega do artigo em versão impressa																X						
26	Apresentação do artigo na XI Semana de Iniciação Científica de SI																X						
27	Postagem definitiva no Moodle do artigo como																X						

	componente TC2.N1																					
28	Escrita da Conclusão																X	X				
29	Postagem da monografia no Moodle para indicação à banca final de avaliação																	X				
30	5ª Reunião de TC																	X				
31	Postagem do TC no Moodle para avaliação da banca final																		X			
32	Defesa do TC na banca final de avaliação																			X		
33	Postagem do TC com as correções indicadas no Moodle																				X	
34	Entrega do CD e termos de autorização e publicação do TC																				X	
35	Obtenção do resultado final do TC																				X	
36	Encerramento das atividades																					X

Fonte: Do autor (2019).

Apêndice B – Links para download das ferramentas e acesso às plataformas computacionais

No Quadro 08 abaixo, estão listados os links para download das ferramentas e acesso às plataformas computacionais de apoio ao processo de ensino-aprendizagem de algoritmos e programação.

É importante ressaltar que todos os links se encontram disponíveis no momento da publicação deste documento. Ao passo da atualização deste documento, os links foram validados quanto a possíveis alterações e/ou remoções.

Quadro 08 – Links das ferramentas e plataformas

Ferramenta/plataforma	Link
CodeBench	http://codebench.icomp.ufam.edu.br
LOGO (KTurtle)	appstream://org.kde.kturtle.desktop
LOGO (SuperLogo)	https://www.nied.unicamp.br/biblioteca/super-logo-3-0-para-windows-7-a-10/
LOGO (xLogo)	https://xlogo.tuxfamily.org/pt/telechargements.html
Portugol Studio	http://lite.acad.univali.br/portugol/
RoboMind	https://www.robomind.net/en/download.html
Scratch	https://scratch.mit.edu/download (*offline*) https://scratch.mit.edu/projects/editor/ (*online*)
The Huxley	https://www.thehuxley.com
URI Online Judge	https://www.urionlinejudge.com.br/judge/en/login
VisuAlg	http://visualg3.com.br/baixe-o-visualg-3-0-7/

Fonte: Do autor (2019).

Apêndice C – Aplicativos móveis para o ensino-aprendizagem de algoritmos e programação de computadores

Abaixo, estão listados alguns aplicativos voltados ao processo de ensino-aprendizagem de algoritmos e programação de computadores. Ao contrário das ferramentas computacionais de apoio abordadas ao longo deste trabalho, o intuito destes aplicativos é oferecer autonomia aos usuários no que tange ao aprendizado dos conceitos introdutórios e aplicação dos mesmos.

A seleção destes aplicativos levou em consideração os seguintes critérios:

- abordagem de ao menos 50% dos conceitos introdutórios de algoritmos e programação.
- avaliação de ao menos 04 (quatro) estrelas dos usuários na plataforma de download;
- e gratuidade da aplicação.

Mediante os critérios acima definidos, foram selecionadas as aplicações descritas no Quadro 09 abaixo. Para cada aplicação, são apresentadas as seguintes informações:

- nome;
- versão (quando disponível);
- conceitos introdutórios abordados;
- recursos;
- e link para download.

É importante ressaltar que, para todos os aplicativos detalhados abaixo, o requisito básico para adequado aproveitamento do conteúdo – teórico e prático – é basicamente a vontade de aprender algoritmo e programação de computadores. Ademais, a explicação dos conceitos introdutórios, apresentação de exemplos e proposição de exercícios de fixação são disponibilizadas pelas próprias aplicações

.

Quadro 09 – Aplicativos móveis para o ensino-aprendizagem de algoritmos e programação

Nome	Versão	Conceitos introdutórios abordados	Recursos	Link para download
Py	1.1.10	- Entrada e saída de dados. - Variáveis e constantes. - Operadores (aritméticos, lógicos e relacionais). - Estruturas condicionais e de repetição.	- Seleção de linguagem de programação (C++, Python, Java, dentre outras). - Explicação e exemplificação dos conceitos introdutórios. – Proposição de exercícios baseados em verificar a compreensão da teoria, completar trechos de código e/ou indicar a saída resultante de um trecho de código. - Idioma: inglês.	https://play.google.com/store/apps/details?id=com.py.learn&hl=pt_BR
SoloLearn	2.6.4	- Entrada e saída de dados. - Variáveis e constantes. - Operadores (aritméticos, lógicos e relacionais). - Estruturas condicionais e de repetição.	- Seleção de linguagem de programação (C, Kotlin, Java, dentre outras). - Explicação e exemplificação dos conceitos introdutórios. – Proposição de exercícios baseados em verificar a compreensão da teoria, completar trechos de código e/ou	https://play.google.com/store/apps/details?id=com.sololearn&hl=pt_BR

		- Vetores e matrizes. - Funções. - Ponteiros.	indicar a saída resultante de um trecho de código. - Idioma: inglês.	
Pseudocode Visualg Algoritmos	2.0.6	- Entrada e saída de dados. - Variáveis e constantes. - Operadores (aritméticos, lógicos e relacionais). - Estruturas condicionais e de repetição. - Funções.	- Escrita de códigos em pseudocódigo português estruturado. - Console dedicado para exibição da saída e mensagens de erro. - Tutorial de sintaxe da linguagem. - Exemplos disponíveis para análise e edição. - Compartilhamento e salvamento de códigos. - Idioma: português.	https://play.google.com/store/apps/details?id=pe.diegovelopeer.pseudocode&hl=pt_BR
Encode	4.6	- Entrada e saída de dados. - Variáveis e constantes. - Operadores (aritméticos, lógicos e relacionais). - Estruturas condicionais e de repetição. - Funções. - Arquivos.	- Seleção de linguagem de programação (Python ou JavaScript). - Explicação dos conceitos em paralelo à proposição de exercícios. Os exercícios consistem em completar trechos de código e/ou escrever novos códigos com base na explicação fornecida. - Idioma: inglês.	https://play.google.com/store/apps/details?id=com.upskew.encode&hl=pt_BR

W3schools	14	- Entrada e saída de dados. - Variáveis e constantes. - Operadores (aritméticos, lógicos e relacionais). - Estruturas condicionais e de repetição. - Vetores e matrizes. - Funções. - Arquivos.	- Seleção de linguagem de programação (C++, Java ou Python). - Ensino dos conceitos introdutórios sob o paradigma de orientação a objetos. - Disponibilização de exemplos os quais podem ser editados e executados pelos usuários. - Disponibilização de exercícios para cada conteúdo apresentado. - Idioma: inglês.	https://play.google.com/store/apps/details?id=com.W3school.Anbu&hl=en_US
Start: Ensino de Programação	1.4	- Entrada e saída de dados. - Variáveis e constantes. - Operadores (aritméticos, lógicos e relacionais). - Estruturas condicionais e de repetição.	- Explicação dos conceitos e aplicação dos mesmos na linguagem de programação Python. - Exemplificação dos conceitos apresentados. - Disponibilização de game no qual o usuário aplica os conceitos apresentados para avançar de nível. - Idioma: português.	https://play.google.com/store/apps/details?id=tk.app.start&hl=pt_BR

Fonte: Do autor (2019).