UNIVERSIDADE FEDERAL DE PERNAMBUCO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

TATYANE SOUZA CALIXTO DA SILVA

SKILLS-CT: UM MODELO PARA CLASSIFICAÇÃO DOS ESTÁGIOS COGNITIVOS DAS HABILIDADES DO PENSAMENTO COMPUTACIONAL E DESENVOLVIMENTO DA APRENDIZAGEM CRIATIVA

Recife
2021

TATYANE SOUZA CALIXTO DA SILVA

SKILLS-CT: UM MODELO PARA CLASSIFICAÇÃO DOS ESTÁGIOS COGNITIVOS DAS HABILIDADES DO PENSAMENTO COMPUTACIONAL E DESENVOLVIMENTO DA APRENDIZAGEM CRIATIVA

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para obtenção do título de Doutora em Ciência da Computação. Área de concentração: Inteligência Computacional.

Orientador (a): Patricia Cabral de Azevedo Restelli Tedesco

Coorientador (a): Jeane Cecília Bezerra de Melo

Recife
2021

Catalogação na fonte
Bibliotecária: Mônica Uchôa, CRB4-1010

S586s Silva, Tatyane Souza Calixto da.
Skills-CT: um modelo para classificação dos estágios cognitivos das habilidades do pensamento computacional e desenvolvimento da aprendizagem criativa / Tatyane Souza Calixto da Silva. – 2021.
163 f.: il., fig., tab.

Orientadora: Patricia Cabral de Azevedo Restelli Tedesco.
Tese (Doutorado) – Universidade Federal de Pernambuco. CIn. Programa de Pós-graduação em Ciência da Computação. Recife, 2021.
Inclui referências e apêndices.

1. Pensamento computacional. 2. Estágios cognitivos. 3. Aprendizagem criativa. I. Tedesco, Patricia Cabral de Azevedo Restelli. (Orientadora). II. Título.

006.31 CDD (23. ed.) UFPE- CCEN 2022 - 09

TATYANE SOUZA CALIXTO DA SILVA

SKILLS-CT: UM MODELO PARA CLASSIFICAÇÃO DOS ESTÁGIOS COGNITIVOS DAS HABILIDADES DO PENSAMENTO COMPUTACIONAL E DESENVOLVIMENTO DA APRENDIZAGEM CRIATIVA

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para obtenção do título de Doutora em Ciência da Computação. Área de concentração: Inteligência Computacional.

Aprovado em: 10/09/2021.

_____

Orientador (a): Profa. Dra. Patricia Cabral de Azevedo Restelli Tedesco

_____

Coorientador (a): Profa. Dra. Jeane Cecília Bezerra de Melo

BANCA EXAMINADORA

_____

Prof. Dr. Geber Lisboa Ramalho
Centro de Informática / UFPE

_____

Profa. Dra. Carla Taciana Lima Lourenço Silva Schuenemann
Centro de Informática / UFPE

_____

Profa. Dra. Claudia Lage Rebello da Motta
Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais / UFRJ

_____

Profa. Dra. Maria Auxiliadora Soares Padilha
Programa de Pós-Graduação em Educação Matemática e Tecnológica / UFPE

_____

Profa. Dra. Danielle Rousy Dias Ricarte
Centro de Informática / UFPB

É preciso ter esperança, mas ter esperança do verbo esperançar; porque tem gente que tem esperança do verbo esperar. E esperança do verbo esperar não é esperança, é espera. Esperançar é se levantar, esperançar é ir atrás, esperançar é construir, esperançar é não desistir! Esperançar é levar adiante, esperançar é juntar-se com outros para fazer de outro modo (FREIRE, 1992, apud CABRAL, 2015, s.p.)

AGRADECIMENTOS

Gostaria de agradecer aos meus pais por sempre me apoiarem e me incentivarem a estudar: meu pai, que me deixava fazer experimentos com os computadores e, em um momento, eu até queimei um computador, mas, mesmo assim, não perdeu a fé em mim e me liga até hoje para tirar dúvidas sobre questões tecnológicas, e minha mãe, por permitir expressar minha arte e criatividade nas paredes de casa.

Ao meu companheiro, Ricardo, que sempre esteve presente apoiando, incentivando e compreendendo o longo tempo dedicado à pesquisa e por entender os planos que tivemos que adiar por conta da minha realização. Obrigada, meu amor, sei que sua vida é mais alegre depois que cheguei nela.

À minha irmã, Amanda, que é minha amiga e parceira de aventuras. Muito obrigada pela escuta empática, pelo apoio e pelo incentivo. Sou grata por compartilhar com você minhas alegrias (e também as tristezas) e pelas boas risadas que sempre damos. Sou muito feliz por ter você em minha vida. Te amo.

À minha amiga Mari (*in memorian*), que nos deixou tão cedo este ano, mas que sempre estará em meu coração. Você era alegre, divertida e iluminava os lugares que passava. Sou grata pelas aulas de dança e por me ensinar a ser uma pessoa melhor.

Às minhas queridas orientadoras: Jeane, que foi minha professora na graduação, fui monitora em uma disciplina que ela lecionava e sua orientanda da graduação até o doutorado – obrigada pela paciência, pela empatia, pelo cuidado e por trazer leveza e afeto à academia; Patricia querida, obrigada pelas perguntas difíceis. Uma excelente professora não é aquela que sabe dar respostas, mas, sim, que faz perguntas que nos fazem pensar. Aprendi muito com você nessa jornada. Sou grata pela sua compreensão, pela empatia em momentos difíceis que compartilhamos e pelas palavras de acolhimento.

A todas as amigas e todos os amigos que me apoiaram.

À banca avaliadora pelas contribuições na minha pesquisa.

Aos meus alunos do curso de Desenvolvimento de *Software*.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo apoio financeiro para a realização desta pesquisa.

RESUMO

O Pensamento Computacional (PC) envolve resolução de problemas utilizando seus modelos, abstrações, organização e decomposição de forma algorítmica. Dessa forma, a resolução de problemas se enquadra na estrutura do Pensamento Computacional cognitivo, pois visa fornecer aos discentes uma compreensão dos principais conceitos computacionais, práticas e perspectivas, enfatizando, assim, a construção de habilidades e competências que serão úteis no ensino superior e em futuras carreiras. Estes elementos, por sua vez, requerem aos sujeitos uma habilidade pouco explorada em processos tradicionais de ensino-aprendizagem: a criatividade, que se torna relevante por promover a integração do fazer e do ser, levando o indivíduo a utilizar suas competências com o intuito de transformar para melhor o mundo que o cerca. Diante deste panorama, o presente trabalho apresenta um estudo realizado com o objetivo de compreender a relação entre os Pilares do Pensamento Computacional, os Estágios Cognitivos Neo-Piagetianos e as habilidades do Pensamento Computacional no processo de resolução de problemas para auxiliar no diagnóstico da aprendizagem do aluno de programação e estimular o desenvolvimento da aprendizagem criativa dos estudantes. Esta correlação tem o intuito de auxiliar o ensino e aprendizagem de programação, bem como fornecer aos docentes uma maneira de avaliar e compreender como os estudantes desenvolvem as habilidades do PC. O processo de ensino de programação envolve a resolução de problemas com base em conceitos como associação, avaliação, atribuição, chamada de procedimento e passagem de parâmetros. Com isto, na busca por promover a utilização de tais conceitos correlacionados, elaborou-se o *Skills-CT,* um Modelo Conceitual que relaciona os Pilares do Pensamento Computacional, classifica os Estágios Cognitivos para cada Habilidade do PC para a resolução de problemas em programação, estimulando a aprendizagem criativa dos estudantes. O *Skills-CT* foi aplicado em uma turma de um curso profissionalizante em Desenvolvimento de *Software* com o objetivo de classificar os Estágios Cognitivos dos alunos em cada habilidade do PC, identificar aquele em que os estudantes possuem menor desenvolvimento e, a partir disso, criar atividades visando amenizar essas dificuldades. Os resultados da pesquisa apontam a relevância da utilização do *Skills-CT* no desenvolvimento das Habilidades do PC, dos Estágios Cognitivos e resolução dos problemas, indicando que ele influenciou positivamente a aprendizagem de programação por parte dos estudantes, o que foi constatado analisando os diversos programas por eles desenvolvidos. Adicionalmente, apresenta-se um guia para o ensino de programação e estratégias consideradas significativas para auxiliar os alunos na aprendizagem criativa.

Palavras-chave: pensamento computacional; estágios cognitivos; aprendizagem criativa.

ABSTRACT

Computational Thinking (CT) involves problem solving using its models, abstractions, organization, and decomposition in an algorithmic way. In this manner, problem solving falls within the framework of cognitive Computational Thinking, as it aims to provide students with an understanding of the main computational concepts, practices, and perspectives, thus emphasizing the construction of skills and competencies that will be useful in higher education and future careers. These elements, in turn, require from the subjects a skill little explored in traditional teaching-learning processes: creativity, which becomes relevant for promoting the integration of doing and being, leading individuals to use their skills to transform the world around them for the better. Given this panorama, the present work presents a study carried out to understand the relationship between the Pillars of Computational Thinking, the Neo-Piagetian Cognitive Stages, and the abilities of Computational Thinking in the problem solving process to aid in diagnosing the learning process. programming students and stimulating the development of creative student learning. This correlation is intended to aid teaching and learning programming, as well as provide faculty with a way to assess and understand how students develop CT skills. The process of teaching programming involves solving problems, based on concepts such as association, evaluation, assignment, procedure call, and parameter passing. Therefore, seeking to promote the use of such correlated concepts, Skills-CT was created, a Conceptual Model that relates the Pillars of Computational Thinking, classifies the Cognitive Stages for each CT Skill, for problem solving in programming, stimulating students' creative learning. The Skills-CT was applied in a group of professional courses in Software Development to classify the students' Cognitive Stages in each CT skill, identifying the one in which students have less development, and thus creating activities to alleviate these difficulties. The research results point to the relevance of using Skills-CT in the development of CT Skills, Cognitive Stages, and problem solving, indicating that it positively influenced the learning of programming by students, which was found by analyzing the various programs developed by them. Additionally, we present a guide to teaching programming and strategies considered significant to assist students in creative learning.

Keywords: computational thinking; cognitive stages; creative learning.

LISTA DE FIGURAS

Figura 1 – Ciclo da pesquisa-ação 23
Figura 2 – Organização da Tese em capítulos 25
Figura 3 – Infográfico sobre os Pilares do Pensamento Computacional 27
Figura 4 – Perspectivas teóricas do Pensamento Computacional 29
Figura 5 – Modelo de ondas sobrepostas 34
Figura 6 – Estágios Cognitivos 38
Figura 7 – Modelo *Skills-CT* 56
Figura 8 – Relação entres os Pilares do PC e Habilidades 56
Figura 9 – Estágios Cognitivos X Níveis de Habilidades de PC para Abstração e Decomposição de Problemas e Paralelismo 57
Figura 10 – Estágios Cognitivos X Níveis de Habilidades de PC Lógica, Controle de Fluxo e Representação dos Dados 58
Figura 11 – Detalhes do Pilares e CPS (*Creative Problem Solving*) 61
Figura 12 – Etapas do Guia para Ensino de Programação 62
Figura 13 – Kanban para preparar aula (professor(a)) 66
Figura 14 – Quadro Kanban Alunos 66
Figura 15 – Guia para o Ensino de Programação 68
Figura 16 – Variáveis – Representação de Dados 69
Figura 17 – Estrutura Condicional – Lógica 70
Figura 18 – Laços de Repetição – Controle de Fluxo 70
Figura 19 – Funções – Abstração de Decomposição de Problemas e Paralelismo 70
Figura 20 – Linha do Tempo – Curso de Desenvolvimento de Software 74
Figura 21 – Jornada do Estudante 74
Figura 22 – Linha do Tempo – Módulo de Lógica de Programação 76
Figura 23 – Linha do Tempo – Módulo de *Python* 77
Figura 24 – Exemplo de saída do classificador criado pela pesquisadora 78
Figura 25 – Estágios Cognitivos de habilidades do PC – Aluno 1 80
Figura 26 – Estágios Cognitivos de habilidades do PC – Aluno 2 81
Figura 27 – Estágios Cognitivos de habilidades do PC – Aluno 3 81
Figura 28 – Estágios Cognitivos de habilidades do PC – Aluno 4 82
Figura 29 – Estágios Cognitivos de habilidades do PC – Aluno 5 82
Figura 30 – Estágios Cognitivos de habilidades do PC – Aluno 6 83

Figura 31 – Estágios Cognitivos de habilidades do PC – Aluno 7 83
Figura 32 – Estágios Cognitivos de habilidades do PC – Aluno 8 84
Figura 33 – Estágios Cognitivos de habilidades do PC – Aluno 9 84
Figura 34 – Estágios Cognitivos de habilidades do PC – Aluno 10 85
Figura 35 – Estágios Cognitivos de habilidades do PC – Aluno 11 85
Figura 36 – Estágios Cognitivos de habilidades do PC – Aluno 1 86
Figura 37 – Estágios Cognitivos de habilidades do PC – Aluno 2 86
Figura 38 – Estágios Cognitivos de habilidades do PC – Aluno 3 87
Figura 39 – Estágios Cognitivos de habilidades do PC – Aluno 4 87
Figura 40 – Estágios Cognitivos de habilidades do PC – Aluno 5 88
Figura 41 – Estágios Cognitivos de habilidades do PC – Aluno 6 88
Figura 42 – Estágios Cognitivos de habilidades do PC – Aluno 7 89
Figura 43 – Estágios Cognitivos de habilidades do PC – Aluno 8 89
Figura 44 – Estágios Cognitivos de habilidades do PC – Aluno 9 93
Figura 45 – Estágios Cognitivos de habilidades do PC – Aluno 10 90
Figura 46 – Estágios Cognitivos de habilidades do PC – Aluno 11 91
Figura 47 – Estágios Cognitivos de habilidades do PC – Aluno 1 91
Figura 48 – Estágios Cognitivos de habilidades do PC – Aluno 2 92
Figura 49 – Estágios Cognitivos de habilidades do PC – Aluno 3 92
Figura 50 – Estágios Cognitivos de habilidades do PC – Aluno 4 93
Figura 51 – Estágios Cognitivos de habilidades do PC – Aluno 5 93
Figura 52 – Estágios Cognitivos de habilidades do PC – Aluno 6 94
Figura 53 – Estágios Cognitivos de habilidades do PC – Aluno 7 94
Figura 54 – Estágios Cognitivos de habilidades do PC – Aluno 8 95
Figura 55 – Estágios Cognitivos de habilidades do PC – Aluno 9 95
Figura 56 – Estágios Cognitivos de habilidades do PC – Aluno 10 96
Figura 57 – Estágios Cognitivos de habilidades do PC – Aluno 11 96
Figura 58 – *Slide* da Aula de Lista, Tupla e Dicionário 97
Figura 59 – *Slide* da Aula Reformulada de Lista Tupla e Dicionário 98
Figura 60 – Porcentagem de respostas corretas por alunos usando *Kahoot* – *Scratch* 99
Figura 61 – Porcentagem de respostas corretas no *Kahoot* – *Scratch* 1 100
Figura 62 – Porcentagem de respostas corretas no *Kahoot* – *Scratch* 2 100
Figura 63 – Porcentagem de respostas corretas no *Kahoot* – *Scratch* 3 101
Figura 64 – Porcentagem de respostas corretas por alunos usando *Kahoot* – Portugol 104

Figura 65 – Código da Q8 do Portugol 102
Figura 66 – Porcentagem de respostas corretas no *Kahoot* – Portugol 1 102
Figura 67 – Porcentagem de respostas corretas no *Kahoot* – Portugol 2 103
Figura 68 – Porcentagem de respostas corretas no *Kahoot* – Portugol 3 103
Figura 69 – Porcentagem de respostas corretas por alunos usando *Kahoot – Python* 104
Figura 70 – Porcentagem de respostas corretas no *Kahoot – Python* 1 104
Figura 71 – Porcentagem de respostas corretas no *Kahoot – Python* 2 105
Figura 72 – Resultado do questionário de autoeficácia para todas as linguagens de programação 105

LISTA DE QUADROS

Quadro 1 – Tipo de pesquisa-ação 23
Quadro 2 – Teoria Clássica X Teoria Neo-Piagetiana 33
Quadro 3 – Tipos de comportamento do programador iniciante no Sensório-Motor 35
Quadro 4 – Tipos de Comportamento do programador iniciante no Pré-Operacional 36
Quadro 5 – Tipos de Comportamento do programador iniciante no Operacional Concreto 37
Quadro 6 – Etapas do CPS 60
Quadro 7 – Solução ótima para as atividades do *Scratch* 106
Quadro 8 – Solução ótima para as atividades do Portugol e *Python* 108

LISTA DE TABELAS

Tabela 1 – Habilidades do Pensamento Computacional para o *Scratch* 63
Tabela 2 – Habilidades do Pensamento Computacional para linguagem de programação procedural 63

LISTA DE ABREVIATURAS E SIGLAS

BBC	*British Broadcasting Corporation*
CEIE	Comissão Especial de Informática na Educação
WIE	Workshop de Informática na Escola
PC	Pensamento Computacional
STEM	*Science, Technology, Engineering and Mathematics*

SUMÁRIO

1 INTRODUÇÃO 17
1.2 MOTIVAÇÃO E JUSTIFICATIVA 17
1.3 DEFINIÇÃO DO PROBLEMA 20
1.4 OBJETIVOS 21
1.5 MÉTODO 21
1.6 ESTRUTURA DA TESE 24
2 FUNDAMENTAÇÃO TEÓRICA 26
2.1 PENSAMENTO COMPUTACIONAL 26
2.2 ENSINO E APRENDIZAGEM DE PROGRAMAÇÃO 30
2.3 TEORIA NEO-PIAGETIANA E PROGRAMAÇÃO 32
2.3.1 Modelos de Estágios 33
2.3.2 Teoria Neo-Piagetiana e Programação 34
2.3.2.1 Estágio Sensório-Motor 34
2.3.2.2 Estágio Pré-Operacional 35
2.3.2.3 Estágio Operacional Concreto 36
2.4 APRENDIZAGEM CRIATIVA 38
2.5 CONSIDERAÇÕES SOBRE O CAPÍTULO 42
3 TRABALHOS RELACIONADOS 43
3.1 DESENVOLVIMENTO DAS HABILIDADES DO PENSAMENTO COMPUTACIONAL, ENSINO E APRENDIZAGEM CRIATIVOS 43
3.1.1 Evidências Nacionais 45
3.1.2 Evidências Internacionais 46
3.2 TEORIA NEO-PIAGETIANA NO ENSINO E APRENDIZAGEM DE PROGRAMAÇÃO 51
3.3 CONSIDERAÇÕES SOBRE O CAPÍTULO 53
4 *SKILLS-CT*: UM MODELO CONCEITUAL DE APRENDIZAGEM CRIATIVA E RESOLUÇÃO DE PROBLEMAS 55
4.1 MODELO CONCEITUAL SKILLS-CT 55
4.2 GUIA PARA O ENSINO DE PROGRAMAÇÃO 62
4.2.1 Utilização do Guia para o Ensino de Programação 67
4.3 CONSIDERAÇÕES SOBRE O CAPÍTULO 72

5 JORNADA DE ENSINO E APRENDIZAGEM COM O *SKILLS-CT* 73
5.1 PERFIL DOS PARTICIPANTES DO CURSO 73
5.2 APLICANDO O MODELO 74
5.3 ESTUDO NO MÓDULO DE LÓGICA DE PROGRAMAÇÃO 75
5.4 ESTUDO NO MÓDULO DE *PYTHON* 77
5.5 FERRAMENTAS USADAS NO ESTUDOS 77
5.6 CONSIDERAÇÕES SOBRE O CAPÍTULO 79
6 RESULTADOS E DISCUSSÃO 80
6.1 RESULTADOS DO *SCRATCH* 80
6.2 RESULTADOS DO PORTUGOL 85
6.3 RESULTADOS DO *PYTHON* 91
6.4 RESULTADO DAS ATIVIDADES COM *KAHOOT* 99
6.5 RESULTADO DO QUESTIONÁRIO DE AUTOEFICÁCIA DO *SCRATCH*, PORTUGOL E *PYTHON* 105
6.6 DISCUSSÃO 106
6.7 CONSIDERAÇÕES SOBRE O CAPÍTULO 109
7 CONCLUSÃO E TRABALHOS FUTUROS 110
7.1 CONTRIBUIÇÕES DA PESQUISA 111
7.2 LIMITAÇÕES 112
7.3 TRABALHOS FUTUROS 112
REFERÊNCIAS 114
APÊNDICE A – ATIVIDADES DO SCRATCH 121
APÊNDICE B – ATIVIDADES DO *VISUALG* 123
APÊNDICE C – ATIVIDADES DE *PYTHON* 126
APÊNDICE D – *SCRATCH* – PRIMEIRO *QUIZ* 136
APÊNDICE E – *SCRATCH* – SEGUNDO *QUIZ* 139
APÊNDICE F – *SCRATCH* – TERCEIRO *QUIZ* 143
APÊNDICE G – PORTUGOL – PRIMEIRO *QUIZ* 145
APÊNDICE H – PORTUGOL – SEGUNDO *QUIZ* 149
APÊNDICE I – PORTUGOL – TERCEIRO *QUIZ* 152
APÊNDICE J – *PYTHON* – PRIMEIRO *QUIZ* 155
APÊNDICE K – *PYTHON* – SEGUNDO *QUIZ* 158
APÊNDICE L – QUESTÕES DE AUTOEFICÁCIA 163

1 INTRODUÇÃO

Este capítulo apresenta o problema de pesquisa juntamente com a motivação e a justificativa para a realização deste trabalho. A partir de tais elementos, são traçados os objetivos e a explicação da metodologia, de modo que a estrutura desta tese é apresentada no fim do presente capítulo.

1.2 MOTIVAÇÃO E JUSTIFICATIVA

Fonte: *Tumblr* Tiras Armandinho (2019)[1].

A disciplina de Programação faz parte da formação básica nos cursos de Tecnologia em geral. Seu conteúdo está voltado ao ensino de conceitos, modelos computacionais e linguagens de programação (SANTOS; COSTA, 2006).

Contudo, pode-se ressaltar que ensinar programação não é apenas ensinar uma linguagem de programação. O processo de ensino deve envolver a resolução de problemas, fundamentando-se em conceitos como associação, avaliação, atribuição, chamada de procedimento e passagem de parâmetros (SANTOS; COSTA, 2006).

Dessa maneira, a literatura apresenta alguns desafios enfrentados pelos docentes no processo de ensino de programação, destacando-se, entre eles, apresentar técnicas de resolução de problemas e trabalhar a capacidade de abstração do estudante na busca das possíveis soluções (SANTOS; COSTA, 2006; GIRAFFA; MULLER; MORAES, 2015). Além de superar tais desafios, faz-se necessário proporcionar um ambiente que estimule a criatividade, podendo, assim, favorecer a aprendizagem autônoma do estudante na resolução de problemas e ainda trabalhar a capacidade de abstração.

Em vista disso, o ensino de disciplinas relacionadas à programação, incluindo seus

[1]Disponível em: https://tirasarmandinho.tumblr.com/post/114087015594/tirinha-original. Acesso em: 16 jan. 2021.

conceitos básicos, representa um grande desafio para docentes na busca por metodologias de ensino mais adequadas. Segundo Watson e Li (2014), uma boa abordagem para o ensino de introdução à programação deve ser baseada em estudo em pequenos grupos e utilização de programação por pares ao invés de aulas tradicionais em sala de aula.

A metodologia de ensino tradicional, comumente utilizada nas aulas de programação, as quais costumam ser divididas em aulas teóricas, teóricas-práticas e/ou laboratoriais e utilização de ferramentas e materiais tradicionais de ensino estático, como livros didáticos, notas de aula, quadros negros e apresentações de *slides*, não têm se mostrado suficientes para o ensino da disciplina (BENNEDSEN; CASPERSEN, 2004). Esses recursos são relevantes para a apresentação do resultado de um processo, porém, não mostram o próprio processo de desenvolvimento, o que se deve ao fato de que o uso destas ferramentas e materiais tradicionais apresentam uma solução ideal que é resultado de um desenvolvimento não-linear (BENNEDSEN; CASPERSEN, 2004).

Esta abordagem apresenta uma questão a ser observada, isto é, quando a apresentação do produto, independente do processo de desenvolvimento, pode interferir na percepção dos alunos de que existe uma "estrada real", linear e direta da solução do problema (BENNEDSEN; CASPERSEN, 2004).

Além disso, é importante prover uma maneira de classificar os erros e fragilidades na aprendizagem desses alunos, como o desenvolvimento do raciocínio abstrato, fundamental na programação. Nesse contexto, a teoria do desenvolvimento cognitivo Neo-Piagetiana afirma que os indivíduos não nascem com habilidades de raciocínio abstrato e que elas são desenvolvidas gradual e sequencialmente em um novo domínio com a experiência adquirida (LISTER, 2011).

Para Lister (2011), a Teoria Neo-Piagetiana poderia ser usada para descrever o desenvolvimento cognitivo de programadores iniciantes. Ele reconheceu que o comportamento análogo ao descrito pelos teóricos Neo-Piagetianos pode ser evidente em programadores novatos em cada um dos estágios de desenvolvimento Sensório-Motor, Pré-Operacional, Operacional Concreto e Operacional Formal. A mudança entre os estágios se dá de forma gradual, sendo o Estágio Operacional Formal aquele em que o aluno pode raciocinar de forma lógica, generalizar ou analisar outras possibilidades de utilização de um conceito aprendido e é nele que se espera que programadores especialistas raciocinem. O foco deste estudo está nos estágios anteriores ao Operacional Formal, ou seja, nos estágios iniciais do desenvolvimento da programação. Assim, a literatura sobre especialistas em programação não é analisada neste trabalho.

Outro fator a ser considerado no processo de ensino-aprendizagem de programação é o Pensamento Criativo. Torna-se necessário engajar as pessoas de modo que possam desenvolver a criatividade, promovendo a integração do fazer e do ser, usando as suas competências com o intuito de transformar o mundo para melhor (YOUNG, 1985; MARTINEZ; TACCA, 2009). Uma outra habilidade que se destaca pela importância é o Pensamento Computacional (PC) (WING, 2006), que consta na lista de Habilidades e Competências requeridas para os profissionais no século XXI, segundo o *National Research Council* (PELLEGRINO; HILTON, 2012). O PC engloba a resolução de problemas utilizando seus modelos, abstrações, organização e decomposição de forma algorítmica, fazendo uso de processos cognitivos comuns aos profissionais da Ciência da Computação, sem que necessariamente se tenha uma formação nesta área do conhecimento, uma vez que, em muitas dessas atividades, haverá, pelo menos, o uso de tecnologias da informação relacionada a um raciocínio computacional algorítmico (WING, 2006).

Nesse cenário, o Pensamento Computacional e a criatividade são vistos como ferramentas cognitivas que expandem o conhecimento e as habilidades que se pode aplicar na obtenção de uma solução para um dado problema. Ou seja, as ferramentas computacionais, quando utilizadas de forma criativa, levam ao desenvolvimento de novas abordagens para problemas antigos e novos, observando diferentes estímulos e perspectivas que podem ser relevantes na sua solução (BLIKSTEIN, 2008; PETERANETZ et al., 2017).

Uma das primeiras abordagens sobre criatividade foi proposta por Guilford (1967) na época em que presidia a Associação Americana de Psicologia. Em seu trabalho, Guilford (1967) afirma que tal habilidade deveria ser reconhecida e desenvolvida devido à sua importância para o indivíduo e para a sociedade, ajudando diretamente na proposta de soluções de problemas enfrentados pela humanidade.

Por sua vez, Alencar e Fleith (2009) apontam o papel fundamental do desenvolvimento da criatividade em estudantes. As abordagens tradicionalmente utilizadas nem sempre favorecem as soluções criativas; assim, mudanças nas práticas educacionais, programas de treinamento e estímulo da criatividade se fazem necessárias visando desenvolver um ambiente educacional engajador e inovador, propício, portanto, ao desenvolvimento do Pensamento Computacional e da aprendizagem de programação.

Destarte, é importante obter uma melhor concepção sobre a classificação dos Estágios Cognitivos dos estudantes em cada uma das habilidades no desenvolvimento do Pensamento Computacional.

1.3 DEFINIÇÃO DO PROBLEMA

A resolução de problemas é uma habilidade metacognitiva que indica a forma que uma pessoa aprende e vivencia diferentes aspectos no processo de solução de problemas. No contexto de programação, é uma habilidade fundamental. Entretanto, muitos estudantes iniciantes de programação possuem dificuldade em formular um problema e expressar sua solução em código (VEERASAMY et al., 2019).

Conceitos e como a utilização de *loops* e condicionais na solução de problemas também figuram nas dificuldades dos discentes. Ademais, alguns estudos afirmam que os estudantes das disciplinas introdutórias de programação não focam o seu desenvolvimento nas habilidades de resolução de problemas, mas, ao invés disso, dão mais ênfase à codificação dos problemas de programação (MALIK; MATHEW; HAMMOOD, 2019).

Vê-se na literatura que, para muitos autores, o objetivo principal de uma disciplina de programação introdutória é desenvolver habilidades de solução de problemas por meio do Pensamento Computacional e dos fundamentos de programação ao invés de dar ênfase no ensino da sintaxe de uma linguagem de programação específica (MEDEIROS; RAMALHO; FALCÃO, 2019). Alinhado com essa abordagem, o *Computer Science Curricula* (JOINT TASK FORCE ON COMPUTING CURRICULA, 2013) sugere que as disciplinas introdutórias de programação deem ênfase nos fundamentos de programação no lugar de uma linguagem de programação.

Segundo Medeiros (2019), as definições são genéricas, ausentes ou inconsistentes no que diz respeito à solução de problemas presente em diversas pesquisas. As razões para as limitações dos alunos na resolução de problemas em programação não são detalhadas e, por este motivo, Medeiros, Ramalho e Falcão (2019) levantaram alguns questionamentos sobre o assunto:

- A dificuldade dos estudantes em resolver problemas é porque eles não entendem a afirmação do problema, não conhecem a estratégia para resolvê-lo ou não conhecem um plano passo a passo para implementar a estratégia?
- Como a solução de problemas está relacionada ao pensamento algorítmico, ao Pensamento Computacional ou à abstração?

Portanto, é importante examinar a relação entre as habilidades de resolução de problemas do aluno e o desenvolvimento do Pensamento Computacional, pois a consciência precoce das habilidades de resolução de problemas dos discentes permite ajudá-los estrategicamente a desenvolverem ainda mais essas habilidades, influenciando no seu

aprendizado de programação.

Diante do exposto, esta pesquisa busca contribuir para as seguintes lacunas detectadas na literatura: i) identificar qual é a relação entre as habilidades de resolução de problemas do aluno e desenvolvimento do Pensamento Computacional; ii) classificar as dificuldades dos alunos nas habilidades do PC e propor atividades direcionadas para ajudá-los a desenvolvê-las; iii) propor atividades que promovam a aprendizagem criativa.

Com o objetivo de abordar o problema de aplicação dos conceitos de programação na resolução de problemas, lançando mão dos Pilares, das habilidades do Pensamento Computacional e dos Estágios Cognitivos da Teoria Neo-Piagetiana, tem-se, a seguir, uma questão que norteará esta pesquisa:

- Como a relação entre os Estágios Cognitivos Neo-Piagetianos e o Pensamento Computacional pode auxiliar no diagnóstico da aprendizagem do aluno de programação e estimular a aprendizagem criativa?

1.4 OBJETIVOS

Este estudo tem o intuito de desenvolver uma abordagem que promova o desenvolvimento do Pensamento Computacional para auxiliar na resolução de problemas por meio da classificação dos Estágios Cognitivos dos estudantes em cada habilidade do PC e estimular o desenvolvimento de aprendizagem criativa deles.

Para tal, se faz necessário alcançar os seguintes objetivos específicos:

I. Propor um modelo *Skills-CT* para ajudar o(a) professor(a) a avaliar os Pilares e as habilidades de Pensamento Computacional e Estágios Cognitivos e, com base nos resultados, propor atividades para facilitar o emprego de conhecimentos na Resolução de Problemas, bem como auxiliar no desenvolvimento do estágio cognitivo do estudante;

II. Desenvolver, a partir do modelo *Skills-CT*, um guia para o ensino de programação e estratégias consideradas significativas para auxiliar os alunos na solução de problemas e desenvolvimento da aprendizagem criativa;

III. Avaliar a aprendizagem dos discentes diante da utilização do modelo *Skills-CT*.

1.5 MÉTODO

A pesquisa aqui apresentada foi desenvolvida por etapas que são descritas a seguir. No

primeiro momento, a literatura sobre Pensamento Computacional e ensino e aprendizagem de programação, incluindo artigos, livros, teses e dissertações, foi consultada e avaliada com o intuito de detectar lacunas nos estudos que poderiam ser contempladas nesta pesquisa. Partindo do estudo da literatura, um modelo inicial foi desenvolvido e os resultados obtidos foram publicados no artigo intitulado *The Creative Process in the Development of Computational Thinking in Higher Education* no *International Conference on Computer Supported Education* (CSEDU) de 2020.

Para a criação do modelo *Skills-CT* proposto nesta tese, foram consultadas pesquisas sobre outros campos de conhecimento, incluindo a Teoria Neo-Piagetiana em Programação e Aprendizagem Criativa, com o objetivo de desenvolver o aporte teórico sobre as questões envolvidas na classificação e na identificação dos Estágios Cognitivos dos alunos.

Após definir o problema de pesquisa, uma nova revisão da literatura foi efetuada, desta vez com o intuito de analisar a classificação dos Estágios Cognitivos e a sua relação com as estratégias de ensino para o desenvolvimento do Pensamento Computação ou aprendizagem de programação dos estudantes. Nesse sentido, foram realizadas investigações de estudos sobre o problema de ensino e aprendizagem de programação e desenvolvimento das habilidades do PC na resolução de problemas. É importante destacar que tais elementos foram estudados de forma separada, uma vez que não foram encontradas pesquisas que fizessem a relação desses assuntos. O processo de pesquisa e seleção de estudos foi realizado de forma sistemática, utilizando-se, para isto, bases de dados nacionais e internacionais.

Dessa maneira, as bases de dados nacionais, onde as *strings* de busca, apresentadas no Capítulo 3, foram usadas, abrangem o Portal de Publicações da Comissão Especial de Informática na Educação (CEIE), o Simpósio Brasileiro de Informática na Educação (SBIE) e o Workshop de Informática na Escola (WIE).

No contexto internacional, as bases consideradas englobam ACM *Digital Library*, IEEE *Xplore* e *ScienceDirec*t. A partir dos resultados da revisão, o modelo denominado *Skills-CT* foi criado e, posteriormente, aplicado através de pesquisa-ação, ou seja, um processo no qual os participantes examinam sua própria prática docente de forma sistemática e cuidadosa, usando as técnicas de pesquisa (FERRANCE, 2000). Suas características são apresentadas no Quadro 1.

Quadro 1 – Tipo de pesquisa-ação.

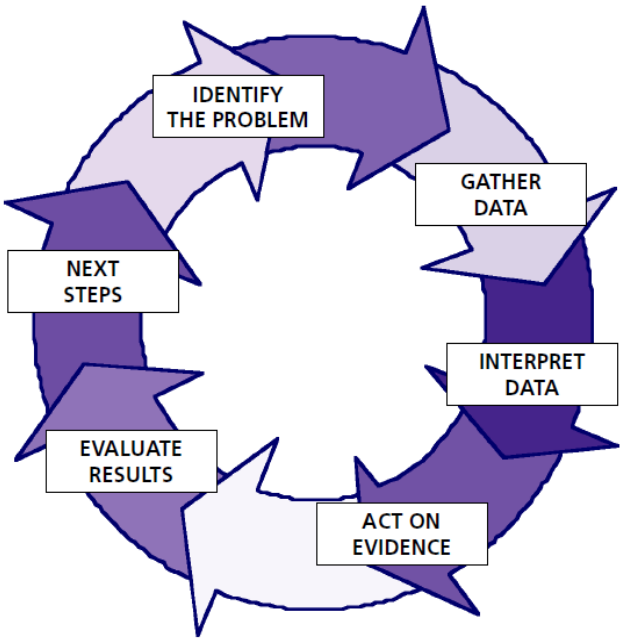
	Pesquisa Individual

Foco	Problema de uma turma
Possível suporte necessário	Mentoria Acesso à tecnologia Assistência com organização e análise de dados
Impacto Potencial	Instrução Avaliação
Resultados	Prática baseada em dados

Fonte: Adaptado de Ferrance (2000).

Para a criação do modelo, utilizou-se o ciclo da pesquisa-ação apresentada na Figura 1.

Figura 1 – Ciclo da pesquisa-ação.



Fonte: Ferrance (2000, p. 15).

A primeira etapa consiste na identificação do problema (*identify a problem area*), enquanto a segunda etapa envolve a coleta de dados (*gather data*), na qual múltiplas fontes de dados são usadas visando auxiliar o entendimento do escopo dos acontecimentos na sala de aula. A fase de interpretação de dados (*interpret data*), por sua vez, está diretamente relacionada à identificação do problema – os docentes podem desejar usar dados coletados nas aulas, dados individuais ou dados de subgrupo, podendo ser quantitativos ou qualitativos. A etapa seguinte consiste em elaborar um plano de ação que propiciará uma mudança na prática docente (*act on evidence*) a partir das informações obtidas na coleta de dados, bem como na revisão da literatura. A penúltima etapa compreende a avaliação dos resultados (*evaluate results*), cujo

intuito é analisar os efeitos da intervenção para determinar se ocorreu melhora. Por fim, os próximos passos (*next steps*) têm o objetivo de identificar as questões adicionais levantadas com base nos dados e planejar melhorias adicionais, revisões e próximas etapas (FERRANCE, 2000).

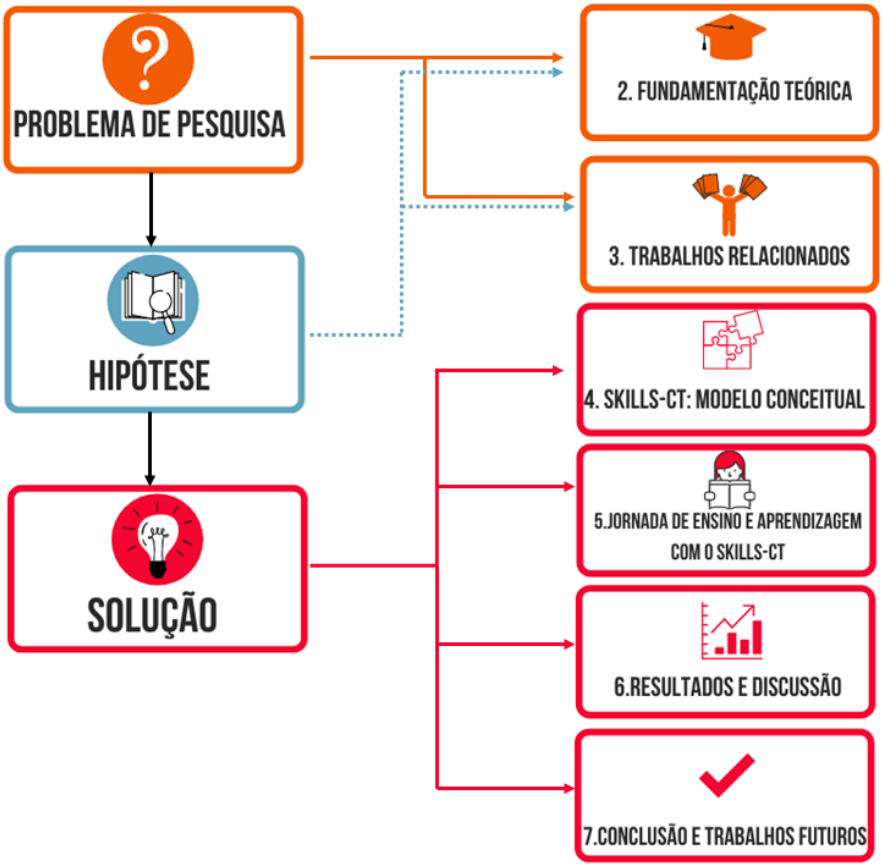
O modelo *Skills-CT* foi criado para auxiliar no desenvolvimento do Pensamento Computacional e propor a relação entre os Pilares e as habilidades do PC e os Estágios Cognitivos dos estudantes. A etapa seguinte consiste na criação do guia para o ensino de programação com o objetivo de materializar o modelo *Skill-CT* e propor ferramentas e abordagens de ensino de programação. A fase final compreende a Jornada de Aprendizagem de Programação que apresenta uma maneira de avaliar a aprendizagem do aluno baseando-se no Guia de Ensino de Programação.

Para avaliação do modelo *Skills-CT*, um quase-experimento foi realizado com os estudantes tendo em vista classificar os Estágios Cognitivos e habilidades do Pensamento Computacional através dos códigos criados por eles na resolução de problemas. As atividades propostas seguiram o Guia de Ensino de Programação e os resultados da aprendizagem usaram a Jornada de Aprendizagem de Programação. Além disso, a análise da aprendizagem criativa foi baseada na teoria de Martinez (2009) que será apresentada no próximo capítulo.

1.6 ESTRUTURA DA TESE

Objetivando facilitar a localização de conceitos e demais assuntos abordados na pesquisa realizada, a estrutura da presente tese encontra-se descrita na Figura 2.

Figura 2 – Organização da Tese em capítulos.



Fonte: A autora (2021).

2 FUNDAMENTAÇÃO TEÓRICA

Fonte: *Instagram* Tirinha de Armandinho (2018)[2].

Este capítulo apresenta os fundamentos teóricos dos elementos básicos utilizados no presente trabalho: Pensamento Computacional, Ensino e Aprendizagem de Programação e Teoria Neo-piagetiana no contexto de Programação e Aprendizagem Criativa.

2.1 PENSAMENTO COMPUTACIONAL

O conceito de Pensamento Computacional (PC) foi proposto em 2006 por Jeannette Wing e está relacionado com a resolução de problemas e percepção do comportamento humano guiados por definições dos fundamentos da Ciência da Computação. O PC aborda um conjunto de definições, habilidades e práticas da Computação que podem ser aplicadas tanto em atividades do cotidiano, como em outras áreas do conhecimento (WING, 2006).

Segundo Barr e Stephenson (2011), o Pensamento Computacional possui nove conceitos que o definem: (i) Coleta de dados: processo de coleta de dados ou informações sobre um problema; (ii) Análise de dados: encontrar padrões resultando em conclusões; (iii) Representação de dados: representar e organizar dados em gráficos, tabelas, textos ou figuras; (iv) Decomposição de problemas: dividir um problema complexo em tarefas menores e gerenciáveis; v) Abstração: reduzir a complexidade para definir ideias principais; (vi) Algoritmos e Procedimentos: sequência de passos para resolução de um problema ou para atingir algum fim; (vii) Automação: usar computadores para realizar tarefas repetitivas; (viii) Paralelização: organizar recursos para, simultaneamente, realizar tarefas para alcançar um objetivo comum; (ix) Simulação: representar ou criar um modelo de um processo.

Adicionalmente, de acordo com um texto publicado pela BBC (2017), o Pensamento

[2]Disponível em: https://www.instagram.com/tirinhadearmandinho. Acesso em: 16 jan. 2021.

Computacional possui quatro pilares que auxiliam na resolução de problemas complexos (FIGURA 3):

Figura 3 – Infográfico sobre os Pilares do Pensamento Computacional.

Fonte: A autora (2021).

1) Decomposição: consiste em decompor um problema ou sistema complexo em partes menores e mais gerenciáveis;

2) Reconhecimento de Padrões: caracterizado por procurar semelhanças entre e dentro de problemas;

3) Abstração: possui a finalidade de concentrar-se apenas na informação importante, ignorando detalhes irrelevantes;

4) Algoritmos: têm o intuito de desenvolver uma solução passo a passo para o problema ou as regras que devem ser seguidas para resolver o problema.

A utilização dos quatro pilares visa auxiliar na programação e na resolução de problemas complexos que são aqueles que, à primeira vista, não se sabe como resolver facilmente. Ou seja, essas etapas ou regras simples são usadas na programação para ajudar a resolver problemas da melhor maneira (BBC, 2017).

Blikstein (2008) explica que o Pensamento Computacional utiliza o computador como uma forma de implementar o pensamento, distinguindo-o de uma ferramenta, e considerando a Ciência da Computação como um elemento para desenvolver a capacidade de resolução de problemas pelos estudantes. Nessa perspectiva, o PC tem o intuito de explicar o pensamento na concepção da computação, utilizando, por exemplo, a abordagem "dividir para conquistar" para decompor uma tarefa complexa em algumas tarefas simples; empregando a ideia de recursão para traduzir o código em dados; aplicando o pensamento da divisão para abstrair o problema real em algumas etapas para resolução; e usando a iteração para executar um mesmo processo ou processos similares.

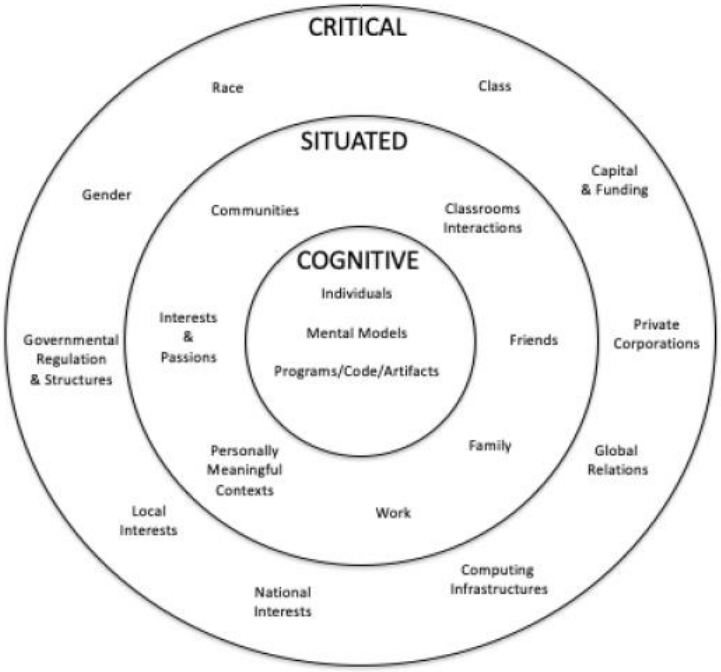
Atualmente, as escolas, de modo geral, buscam trazer elementos de tecnologia através da utilização de ferramentas e aplicativos, tais como navegadores, planilhas eletrônicas e editores de texto. Tal abordagem propaga nos estudantes uma ideia de que a tecnologia serve apenas para recombinar informações existentes e não para criar um conhecimento novo, conforme proposto por Blikstein (2008), lançando mão de elementos de Ciência da Computação aliados à criatividade.

Dessa maneira, ao falar em Computação enquanto ciência, deve-se levar em consideração os seus princípios fundamentais, como a abstração e o raciocínio lógico, que podem ser aplicados na solução de problemas e no desenvolvimento do conhecimento (BLIKSTEIN, 2008). Para Resnick et al. (2009), o estudo de programação de computadores é uma maneira de entender as ideias centrais relacionadas ao PC, uma vez que pode auxiliar no desenvolvimento de habilidades como raciocínio lógico, resolução de problemas e pensamento algorítmico.

Adicionalmente, Kafai, Proctor e Lui (2019) dividem o Pensamento Computacional em três perspectivas teóricas: (1) Pensamento Computacional Cognitivo, que possui o intuito de apresentar aos estudantes uma compreensão dos fundamentos da computação, práticas e perspectivas, com foco no desenvolvimento das habilidades e competências que serão relevantes na formação acadêmica e em carreiras futuras; (2) o Pensamento Computacional Situado, que evidencia a expressão criativa individual e o envolvimento social como um meio para aprender os fundamentos da computação, baseando-se no interesse dos jovens em mídia e produção digital; e (3) o Pensamento Computacional Crítico, que utiliza a computação com

intuito mais amplo de educação para justiça e recomenda uma abordagem analítica aos valores, às práticas e à estrutura da sociedade (FIGURA 4).

Figura 4 – Perspectivas teóricas do Pensamento Computacional.



Fonte: Kafai, Proctor e Lui (2019).

Na perspectiva Cognitiva, o Pensamento Computacional é considerado como uma maneira de os estudantes solucionarem problemas. As atividades instrucionais são desenvolvidas para apresentar aos estudantes os fundamentos de programação como *loops*, recursão, condicionais e estruturas de dados e práticas como iteração, abstração e automação.

As pesquisas que abrangem a perspectiva Cognitiva enfatizam a aprendizagem do discente pelo uso do *Scratch*, observando o entendimento dos estudantes sobre os conceitos básicos de Ciência da Computação, além de analisar a capacidade de programação dos discentes e a compreensão de construções de codificação básicas e avançadas (por exemplo, variáveis, lógica condicional) por meio de atividades como entrevistas em voz alta, criação de projetos funcionais, entre outros. Ademais, há estudos que focam nos contextos específicos de aprendizagem com *Scratch* para propiciar ganhos cognitivos, por exemplo, observando como as linguagens de programação baseadas em texto ou blocos podem apoiar a aprendizagem dos fundamentos de computação de diversas formas (KAFAI; PROCTOR; LUI, 2020).

A perspectiva teórica Situada do PC tem o foco em atividades orientadas por interesse e apoiadas por pares. Nesta concepção, o Pensamento Computacional é entendido como um canal para expressão pessoal e conexão entre as pessoas. Embora a compreensão dos estudantes

dos fundamentos de Computação e habilidades para programar sejam importantes, eles são meios para expressar os interesses e identidades dos discentes para outras pessoas em suas comunidades (KAFAI; PROCTOR; LUI, 2020).

Por fim, o Pensamento Computacional Crítico é entendido como um veículo potencial para se envolver com os desafios políticos, morais e éticos do mundo, por exemplo: considerando questões como o preconceito implícito em *software* de previsão de crimes que a polícia usa ou como a vigilância em massa pelas mídias sociais pode criar aberturas para "hackeamento" eleitoral. No PC Crítico, o intuito principal é o conteúdo crítico – o desenvolvimento de habilidades em programação é um resultado desejável, porém secundário, enquanto o foco é a compreensão cognitiva dos conceitos de PC e aplicação nos problemas sociais como a chave para se tornar um praticante mais crítico da computação (KAFAI; PROCTOR; LUI, 2020).

Nesta pesquisa serão utilizados os quatro pilares do Pensamento Computacional proposto pela BBC (2017) para resolução de problemas e a perspectiva teórica Cognitiva do PC, corroborando com os objetivos da atual proposta. Na próxima seção, será abordado o assunto de ensino e aprendizagem de programação trazendo pontos de reflexão sobre as dificuldades dos estudantes de aprender a resolver problemas algoritmicamente e dos professores a ensinarem esta habilidade aos discentes. O desenvolvimento do Pensamento Computacional auxilia na resolução de problemas e, nesta tese, será usado para dar suporte à aprendizagem de programação.

2.2 ENSINO E APRENDIZAGEM DE PROGRAMAÇÃO

As disciplinas introdutórias de programação apresentam um conjunto de conceitos básicos, sintaxe de uma linguagem de programação para que os estudantes possam aprender. Os discentes devem desenvolver habilidades de resolução de problemas usando a linguagem de programação para expressar soluções (GOMES; MENDES, 2014).

Nesse cenário, a resolução de problemas é uma habilidade fundamental na disciplina de programação e a sintaxe pode dificultar o desenvolvimento dos fundamentos da resolução de problemas. Adicionalmente, o estudo de Koulouri, Lauria e Macredie (2014) apresentou que, no processo iterativo, a falta de habilidades de resolução de problemas pode ser uma das razões tácitas para os estudantes terem dificuldade em usar conceitos como *loops* e condicionais.

A literatura apresenta, então, um conjunto de dificuldades associadas ao ensino e aprendizagem de programação (GOMES; MENDES, 2014), dividindo as causas das

dificuldades de muitos alunos em quatro categorias: estratégias de ensino, atitudes do aluno, métodos de estudo e dificuldades de programação natural (GOMES et al., 2017). Muitos estudantes estão acostumados à estratégia de memorização (ler, ver exercícios resolvidos), o que não é suficiente para aprender programação – faz-se necessário ainda se envolver em uma prática intensiva de solução de problemas, enfrentando as dificuldades de cada problema e procurando solucioná-los. Isto deve ser baseado em habilidades genéricas de resolução de problemas previamente adquiridas que geralmente os discentes não possuem. Inclusive, a programação também é uma causa de dificuldades, posto que sua natureza abstrata, as características e a sintaxe das linguagens (criadas para profissionais e não para alunos) não facilitam a tarefa dos estudantes (GOMES; MENDES, 2013; GOMES et al., 2017) .

O desenvolvimento das habilidades de resolução de problemas em programação é o aspecto mais complexo para os programadores iniciantes, uma vez que exige um conhecimento sólido dos fundamentos de programação que está relacionado à experiência em programar (GOMES; MENDES, 2013). Por conseguinte, é importante conhecer o estágio cognitivo do aluno para entender suas fragilidades nos fundamentos de programação e auxiliá-lo em seu aprendizado.

Outra questão a ser observada na aprendizagem de programação é o aumento do desinteresse dos estudantes que pode estar relacionado aos métodos de ensino que, por vezes, ainda se baseiam em aulas expositivas, conforme Piteira e Haddad (2011) descrevem. Embora o modelo seja eficaz em termos de custos, está longe de ser ideal do ponto de vista pedagógico. Segundo Scaico, Dantas e Farias (2013), as dificuldades em aprender programação estão associadas à utilização de meios que possam motivar e engajar os estudantes no processo, bem como ao ajuste de diferentes emoções ligadas ao processo de aprender, como é o caso da frustração.

Santiago e Kronbauer (2016), dessa forma, relatam que o estudo sobre o ensino e aprendizagem de programação apresenta evidências de que:

- Uma metodologia de ensino de conceitos de programação baseada tão somente em sintaxe é negativo para o processo de aprendizado dos alunos iniciantes;
- Os estudantes são desestimulados devido às dificuldades com as matérias introdutórias de programação, pois não conseguem criar os corretos modelos mentais acerca do conhecimento que está sendo aprendido;
- Há a necessidade de novas propostas de metodologias instrucionais que não sejam baseadas puramente em sintaxe.

No que refere ao ensino de programação, Nobre e Menezes (2002) destacam alguns

desafios do professor: i) apresentar técnicas de soluções de problemas; e ii) trabalhar a capacidade de abstração do estudante, tanto na busca de possíveis soluções, quanto na escolha da estrutura de dados a ser utilizada.

Diante deste panorama, algumas estratégias criativas podem ser utilizadas no ensino de programação: diversificar as tarefas propostas por meio dos métodos de educação, transformação, simulação e outros; utilização do Pensamento Computacional para resolução de problemas, técnicas e avaliação; criar um espaço de divulgação de trabalhos dos estudantes; compartilhar experiências pessoais relacionadas ao assunto estudado; orientar o estudante a buscar informações adicionais sobre os tópicos do seu interesse (ALENCAR; FLEITH, 2009). Outrossim, a aprendizagem criativa pode ser usada para amenizar as dificuldades dos estudantes na resolução de problemas, considerando que promove a transformação pessoal a partir da obtenção de novas habilidades e conhecimentos.

Sternberg (2003), assim, aponta algumas estratégias para proporcionar um ambiente educacional criativo: encorajar o estudante a correr risco; aceitar o erro como parte do processo de aprendizagem; propiciar oportunidades para exploração do ambiente, para questionamentos de pressupostos e gerar múltiplas hipóteses.

Diante dos desafios apresentados, é preciso conhecer metodologias para ensino e também os Estágios Cognitivos dos alunos para propor atividades adequadas. A subseção a seguir apresenta a Teoria Neo-Piagetiana aplicada à programação que será usada como aporte teórico para identificar os Estágios Cognitivos dos alunos.

2.3 TEORIA NEO-PIAGETIANA E PROGRAMAÇÃO

A Teoria Neo-Piagetiana baseia-se no trabalho de Piaget, porém foi modificada. Conforme Knight e Sutton (2004), os fundamentos de Piaget apresentados na teoria são:

- Propriedades organizacionais da cognição, por exemplo, esquemas e estágios;
- As estruturas cognitivas tornam-se mais complexas com maturidade e experiência;
- Níveis mais abstratos e complexos de conhecimento e compreensão fundamentados nos níveis anteriores;
- Os estágios aparecem em sequências universais relacionadas, mas não são definidos pela idade.

O Quadro 2 apresenta as principais diferenças entre a Teoria Clássica Piagetiana e a

Neo-Piagetiana.

Quadro 2 – Teoria Clássica X Teoria Neo-Piagetiana.

Teoria Clássica	Teoria Neo-Piagetiana
Preocupa-se com o desenvolvimento cognitivo geral das crianças.	Preocupa-se com o desenvolvimento cognitivo de pessoas de qualquer idade à medida em que aprendem qualquer nova tarefa cognitiva.
Uma criança em um determinado estágio piagetiano aplica o mesmo tipo de raciocínio a todas as tarefas cognitivas (por exemplo, matemática e xadrez).	Uma vez que a habilidade cognitiva de uma pessoa em qualquer domínio é uma função de seu conhecimento de domínio, uma pessoa frequentemente exibirá diferentes estágios Piagetianos em diferentes domínios de conhecimento.
Testes gerais, como o teste do pêndulo, podem determinar o estágio piagetiano de um indivíduo.	Não há testes gerais, portanto, há falha em encontrar fortes correlações entre a capacidade de programação e o teste de pêndulo.
Prescreve faixas etárias típicas para cada estágio piagetiano, mas as evidências empíricas mostram grande flexibilidade nas faixas etárias devido a fatores culturais e ambientais.	O tempo que os indivíduos passam em qualquer estágio é livre para variar e varia de acordo com sua taxa de aquisição de conhecimento em um domínio de conhecimento específico.
As crianças passam um longo período em um estágio antes de passar por uma rápida mudança para o próximo estágio – a "metáfora da escada".	A metáfora da escada é, às vezes, aplicada, mas também o é a metáfora da "onda sobreposta" (a ser explanada na subseção 2.3.1).

Fonte: Teague (2015).

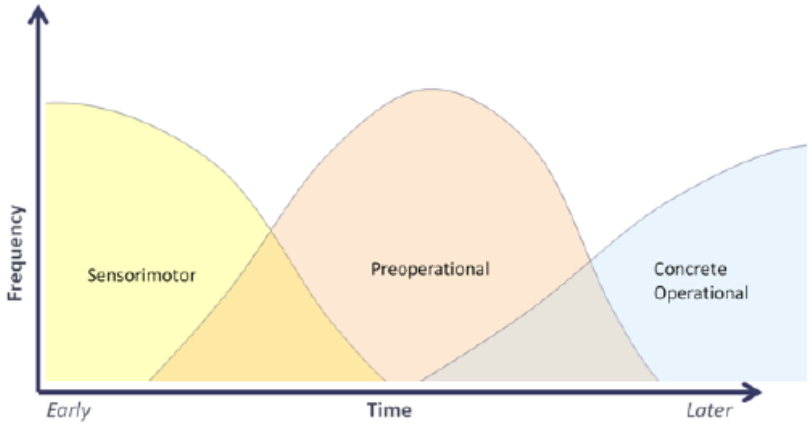
2.3.1 Modelos de Estágios

Para os autores Neo-Piagetianos, o tempo que os indivíduos levam para avançar do estágio Sensório-Motor para o Operacional Formal pode variar. Contudo, esses autores diferem sobre a maneira de progredir por meio desses estágios. Alguns teóricos Neo-Piagetianos estão de acordo com o modelo piagetiano clássico no qual os Estágios Cognitivos são apresentados como um modelo em escada, isto é, os indivíduos ficam um determinado período em um nível antes de passar por uma breve mudança para o próximo nível. Após a modificação de estágio, os indivíduos não voltam ao nível anterior.

O modelo da escada apresenta algumas questões a serem observadas. O primeiro ponto é empírico – constatou-se que as pessoas apresentam, simultaneamente, os padrões de raciocínio em mais de um nível. O segundo ponto é filosófico – como e por que um indivíduo dá o salto de um nível para o próximo? (SIEGLER, 1996).

A transição dos níveis pode ser explicada através das ondas sobrepostas quando uma pessoa apresenta características de dois ou mais níveis à medida em que desenvolve habilidades em um domínio de conhecimento (SIEGLER, 1996; BOOM, 2004; FELDMAN, 2004). No modelo de ondas sobrepostas, as características do nível inicial dominam os comportamentos inicialmente ao trabalhar em um novo domínio de conhecimento. Conforme os esquemas são construídos e o progresso cognitivo é feito, há um aumento no uso do próximo nível de raciocínio mais maduro e uma diminuição no nível menos maduro. Dessa forma, há uso concomitante de estratégias de raciocínio de diferentes estágios (FIGURA 5).

Figura 5 – Modelo de ondas sobrepostas.



Fonte: Teague (2015).

2.3.2 Teoria Neo-Piagetiana e Programação

A abstração é um dos fundamentos da programação (BENNEDSEN; CASPERSEN, 2008). Entretanto, a Teoria Neo-Piagetiana afirma que não nascemos com habilidades de raciocínio abstrato, mas elas são desenvolvidas progressiva e sequencialmente em um novo domínio por meio da experiência neste domínio. O resumo dos Estágios Cognitivos pode ser visto no infográfico da Figura 5 e, nas subseções seguintes, esses estágios Neo-Piagetianos serão abordados.

De acordo com Lister (2020), há quatro etapas do desenvolvimento cognitivo do programador iniciante, com base na Teoria Neo-Piagetiana. As subseções a seguir apresentam a literatura sobre essas quatro etapas e as dificuldades de aprender a programar, categorizadas por comportamentos característicos de cada um dos estágios Neo-Piagetianos de desenvolvimento cognitivo.

2.3.2.1 Estágio Sensório-Motor

O primeiro estágio Neo-piagetiano é o Sensório-Motor. O iniciante em programação tem pouca capacidade de produzir resultados consistentes de maneira confiável por meio do rastreamento de código. Em outras palavras, os recém-chegados ao estágio Sensório-Motor veem o código como algo “mágico”. Ou seja, eles não experimentam um programa sendo executado como uma máquina determinística e ainda têm dificuldade em diferenciar entre palavras reservadas, embutidos, variáveis, literais de lista, pontuação, entre outros, e regras de sintaxe de compreensão. A manipulação de código é por tentativa e erro (LISTER, 2011).

Há ainda a dificuldade em entender o código de outra pessoa. O programador iniciante pode ser capaz de replicar o código, mas como o conhecimento da linguagem é frágil, é difícil aplicá-lo em novas situações. Além disso, ele tem dificuldade em entender a relação entre as variáveis e seus valores e a ordem de execução das linhas de código (LISTER, 2020) (QUADRO 3).

Quadro 3 –Tipos de comportamento do programador iniciante no Sensório-Motor.

Indicadores Comportamentais	**Descrição**
Desenvolvimento da Linguagem de Programação	Distinguir entre palavras-chave, *built-ins*, variáveis e compreensão das regras de sintaxe.
Manipulação de objetos por tentativa e erro	Fazer alterações no código sem entender o que as alterações fazem ou o que estava errado em primeiro lugar.
Aprendendo por meio da interação	Dificuldade em entender qualquer coisa sem extensa experimentação. Olhar para o código de outra pessoa não ajuda – pode acontecer de estar em conformidade com uma série de equívocos no momento, mas não há um entendimento subjacente. Pode ser capaz de replicar o mesmo código, mas, como o conhecimento da linguagem é frágil, é incapaz de se aplicar a novas situações.
Capacidade de vincular as variáveis a seus valores	Compreender a relação entre as variáveis e seus valores e o conceito de ordem de execução das linhas de código.

Fonte: Teague (2015).

2.3.2.2 Estágio Pré-Operacional

No estágio Pré-operacional, os programadores iniciantes podem rastrear o código de forma confiável, mas eles não abstraem rotineiramente do código para ver um cálculo significativo executado por esse código (LISTER, 2020). Eles têm dificuldade em alterar os valores de uma matriz se eles forem diferentes no início ou no final dela. Uma outra dificuldade detectada consiste em raciocinar sobre como alterar o valor de uma variável em um *loop*, avaliar a primeira e a última iteração de um *loop*, além de ignorar as iterações intermediárias.

Os programadores que estão no Pré-operacional têm dificuldade em alterar a implementação de um programa enquanto retêm as especificações e abstraem corretamente além do próprio código. Além disso, eles têm dificuldade em ver a aplicação de um conceito em novas situações (TEAGUE, 2015).

Quadro 4 – Tipos de comportamento do programador iniciante no Pré-Operacional.

Indicadores Comportamentais	**Descrição**
Estados / centralização temporal (ou seja, foco em estados estáticos)	Dificuldade de abstração sobre a mudança de valor de uma variável em um *loop*; olhando para a primeira e última iteração de um *loop* e ignorando as iterações intermediárias.
Irreversibilidade	Dificuldade em escrever código sistematicamente para desfazer um efeito ou reverter uma ação (por exemplo, mudar valores em uma matriz na direção oposta).
Argumento “transdutivo” (relacionar fatos que não mantêm ligação entre si)	Raciocinar sobre o código é restrito a suposições indutivas com base na saída de um programa; presume-se que os programas que produzem a mesma saída tenham o mesmo propósito.
Aplicar conhecimento	Dificuldade em ver a aplicação de um conceito em novas situações ou a relação entre partes do código (em qualquer nível: métodos; blocos / sequência de código; variáveis).
Incapacidade de distinguir todo o grupo e cada uma de suas partes ao mesmo tempo	Dificuldade em compreender a visão geral do programa e a relação entre cada uma das partes do código (em qualquer nível: métodos; blocos / sequência de código; variáveis).

Fonte: Teague (2015).

2.3.2.3 Estágio Operacional Concreto

Os programadores de estágio Operacional Concreto podem raciocinar sobre abstrações de seu código. Eles podem, por exemplo, relacionar código a diagramas e também podem ver como as linhas individuais em um fragmento de código funcionam juntas para realizar alguns cálculos gerais.

No Estágio Operacional Concreto, os programadores podem relacionar as partes do código e entender como todos eles se encaixam para formar o quadro geral. Eles não se concentram apenas nos estados inicial e final dos *loops*, mas podem raciocinar sobre a mudança de variáveis nas iterações do *loop*, ou seja, há a capacidade de raciocinar sobre a relação entre duas variáveis por meio de um intermediário.

Além disso, eles podem escrever um código para executar ações conhecidas, mas esta capacidade é limitada ao considerar a extensão dessas ações a novas situações. Podem ainda raciocinar sobre o código como uma combinação de partes menores que executam funções.

Quadro 5 – Tipos de comportamento do programador iniciante no Operacional Concreto.

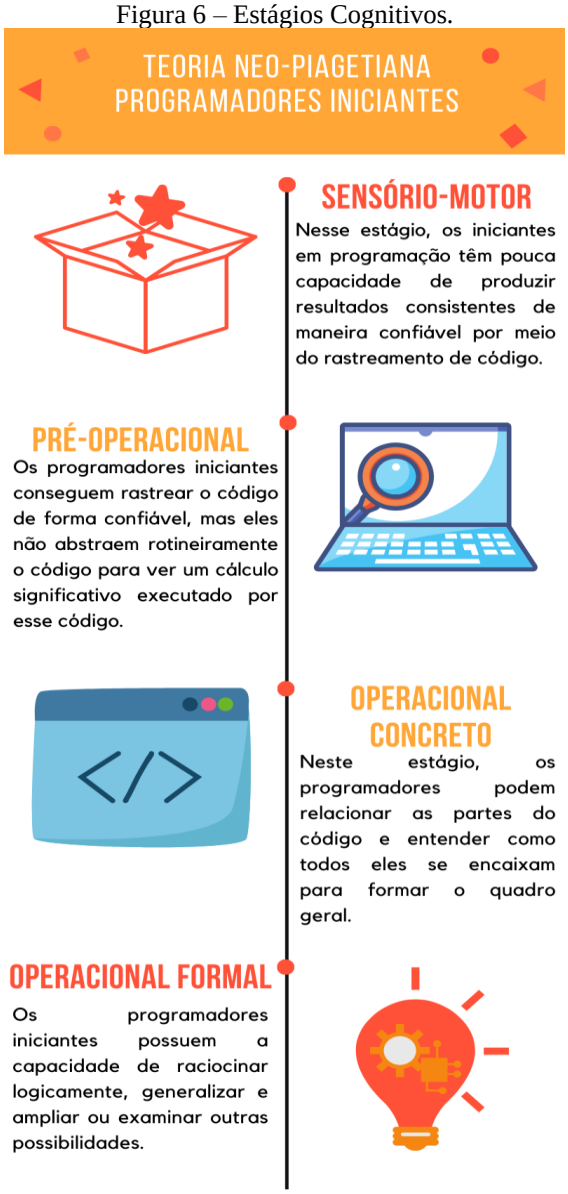
Indicadores Comportamentais	**Descrição**
A realidade inferida	Compreender o conceito de invariantes (por exemplo, um invariante de *loop*: algo que é consistente a cada iteração do *loop*).
Descentralização espacial (ou seja, um foco mais amplo, no todo ou em várias partes) – leva em consideração todos os dados perceptuais	Capacidade de ver os relacionamentos entre as partes do código (em qualquer nível: métodos; blocos / sequência de código; variáveis) e entender como todos eles se encaixam para formar o quadro geral.
Transformações	Não dá ênfase aos estados inicial e final dos *loops*. É capaz de raciocinar sobre a transformação das variáveis em todas as iterações do *loop.*
Pode realizar inferência transitiva	Capacidade de raciocinar sobre a relação entre duas variáveis por meio de uma intermediária.
Não pode estender e examinar possibilidades	Capaz de escrever código para realizar ações familiares, mas a capacidade é limitada para estender essas ações para problemas desconhecidos.

Fonte: Teague (2015).

Adicionalmente, para ajudar o estudante a avançar no Estágio Cognitivo no contexto da Teoria Neo-Piagetiana (FIGURA 6) para programação, o docente pode desenvolver atividades adequadas para analisar os Estágios Cognitivos, contudo, essas atividades podem ser

fundamentadas na aprendizagem criativa com o intuito de desenvolver esta habilidade no aluno, uma vez que ela está relacionada com a resolução de problemas. A próxima seção apresenta a definição e a característica da aprendizagem criativa que serão usadas no capítulo 4 no modelo *Skills-CT*.

Figura 6 – Estágios Cognitivos.



Fonte: A autora (2021).

2.4 APRENDIZAGEM CRIATIVA

Há diversas definições sobre o termo criatividade, não havendo, no entanto, um consenso entre os autores quanto a seu significado exato. Assim, o tema é abordado sob diferentes pontos de vista, por exemplo: a Teoria de Investimento em criatividade de Sternberg

(2003), o Modelo Componencial de criatividade de Amabile (1995) e a Perspectiva de Sistemas de Csikszentmihalyi (1988). Isto está relacionado ao fato de que a criatividade, do mesmo modo que a inteligência, é um constructo complexo, com aspectos diversos, tais como características do indivíduo, processo criativo, itens presentes no produto criativo ou ainda aspectos do ambiente onde as pessoas se encontram inseridas que pode influenciar na expressão criativa (ALENCAR; FLEITH, 2009).

Segundo a Teoria de Investimento em criatividade de Sternberg (2003), haverá criatividade a partir de seis elementos inter-relacionados: habilidades intelectuais, conhecimento, estilos de pensamento, personalidade, motivação e ambiente adequado. Esta perspectiva múltipla da criatividade está relacionada com combinações de aspectos inerentes ao indivíduo, dependendo de fatores cognitivos, emocionais e ambientais.

Já o Modelo Componencial da criatividade de Amabile (1995, p. 30) define um produto ou resposta que será julgado como criativo na medida em que: " a) é novo e apropriado, útil, correto ou de valor para a tarefa em questão, e b) a tarefa é heurística e não algorítmica". Para este autor, um produto é considerado criativo quando há três elementos interagindo: habilidades de domínio, processos criativos relevantes e motivação intrínseca. As habilidades estão relacionadas com o extenso conhecimento e técnicas em um determinado domínio, ao passo que os processos criativos são relativos à aplicação de uma heurística para percorrer novos caminhos cognitivos e a motivação intrínseca, por sua vez, aborda o comprometimento do indivíduo com uma atividade.

De acordo com a Perspectiva de Sistema de Csikszentmihalyi (1988), a criatividade é consequência da interação de dois aspectos: o domínio e o indivíduo. O domínio está relacionado com as informações que o indivíduo possui em um determinado conhecimento e este produz transformações que podem ou não ser acrescentadas a esse domínio. Contudo, essa produção criativa apenas será aceita como tal quando submetida ao julgamento de especialistas na área correlata, influenciado por uma cultura, podendo, portanto, deixar de ser considerada criativa em outra época.

Para MacKinnon (1962), três condições básicas são necessárias para que haja criatividade: a resposta deve ser nova ou, pelo menos, estatisticamente infrequente, devendo adaptar-se à realidade e servir para resolver um problema ou alcançar uma meta reconhecível, além de incluir avaliação, elaboração e desenvolvimento do *insight* original. Para esta pesquisa, será usada a definição de criatividade de MacKinnon (1962), pois corrobora com os objetivos da tese.

O Pensamento Criativo é definido e dividido por Epstein (2005), através da Teoria da

Generatividade, em quatro competências: capturar, desafiar, ampliar e envolver. A competência de capturar refere-se à capacidade de reconhecer e notar ideias únicas à medida em que elas ocorrem. A capacidade de desafiar o pensamento estabelecido e padrões de comportamento relaciona-se com a capacidade de gerar novas abordagens aos problemas. A competência de ampliar o conhecimento de alguém além da própria disciplina permite a integração inovadora de ideias. Por último, envolver relaciona-se ao estímulo que pode ser social ou ambiental, podendo levar a novas experiências e ideias.

A criatividade e o Processo Criativo são temáticas pesquisadas sob vários aspectos, entre eles: métodos, processos e cognição. A criatividade é também uma capacidade cognitiva e, como tal, pode ser desenvolvida, aplicada para criar algo novo e provido de valor para um determinado contexto (TSCHIMMEL, 2010).

Nesse contexto, os termos "criatividade", "Pensamento Criativo" e "Processo Criativo", ainda que algumas vezes utilizados como sinônimos, neste estudo são usados com significados diferentes, quais sejam: a criatividade é a capacidade sistêmica que se manifesta nas soluções (ideias, produtos, conceitos, questões, processos, etc.) novas e providas de valor, influenciada por diversos fatores contextuais do ambiente social e cultural (TSCHIMMEL, 2010; ZAVADIL; SILVA, 2016). Parte desta capacidade está relacionada ao pensamento criativo do sujeito, com suas operações cognitivas e conhecimentos adquiridos, que irão criar novas conexões entre ideias.

No entanto, como esse indivíduo atua em conjunto com outros sujeitos, seus pensamentos serão também impulsionados e influenciados por contextos sociais e trabalhos em grupos nos quais as ideias são construídas a partir da expressão e da comunicação do pensamento do outro (TSCHIMMEL, 2010). Isto acontece por intermédio do Processo Criativo, constituído de métodos, técnicas, instrumentos e o próprio conhecimento processual que pode facilitar o desenvolvimento de novas concepções, lidando com os diversos fatores de influência da criatividade, estimulando o Pensamento Criativo e facilitando a comunicação e a interação entre os indivíduos (ZAVADIL; SILVA, 2016).

Adicionalmente, a aprendizagem pode ser uma forma pela qual a criatividade se expressa em um processo. Para Adalberto (2016), a aprendizagem criativa pode ser definida como uma transformação pessoal a partir da obtenção de novas habilidades e conhecimentos que ocorrem através do engajamento direto na realização de projetos que sejam significativos para o indivíduo, tornando-os criativos e capazes de desenvolver produtos em qualquer contexto.

Martinez e Tacca (2009) explicam que a Aprendizagem Criativa propõe que o estudante

utilize processos de personalização das informações, de comparação com os dados a partir de interpretação e pontos de vistas próprios e de geração de novas ideias que vão além do que foi posto inicialmente. Dessa forma, o discente se transforma em sujeito do seu processo de aprender incorporando e produzindo o conhecimento de maneira personalizada, ativa e criativa (MARTINEZ; TACCA, 2009).

Para os autores, a aprendizagem criativa possui três características marcantes:

- A personalização da informação, referente à maneira pela qual a informação se integra à subjetividade do aprendiz. Assim, a informação personalizada é aquela que está relacionada às motivações do indivíduo, determinando que o próprio processo de recebimento da informação seja essencialmente ativo, pois a pessoa individualiza a informação baseada em suas vivências pessoais. A informação recebida é "modificada" a partir dos conhecimentos que o estudante possui, suas fontes e os recursos subjetivos de que dispõe. Ou seja, a personalização não se refere a informações que são aprendidas de maneira geral, mas a informações que se modificam ao se vincularem ao sistema subjetivo do indivíduo. A personalização da informação é essencial para a criação de representações próprias do objeto de conhecimento. Por sua natureza subjetiva, essa personalização permite ao estudante aplicar o conhecimento em condições diferentes daquelas em que a informação foi ensinada;
- A confrontação com o dado, relacionada ao questionamento, à não aceitação do dado como verdade única, o que permite ao aprendiz identificar incongruências, lacunas e contradições. O questionamento, a problematização da informação, a não aceitação do que é dado como verdade ou como única alternativa são maneiras de expressar o caráter "transgressor" da Aprendizagem Criativa. Isto, entre outros fatores, é o que permite ao estudante identificar falhas, lacunas e contradições nas informações que recebe. Esses processos são importantes para produção e geração de novas ideias;
- A produção de ideias próprias e "novas", relacionada à transcendência do dado, a ir além do que está posto e construir novas relações. A Aprendizagem Criativa é identificada pela geração de ideias próprias sobre o que foi estudado, ideias que, ao transcender o que é dado, expressam a novidade como uma característica essencial da criatividade. Deste modo, novas possibilidades e hipóteses são desenvolvidas e novas ideias surgem sobre o objeto de conhecimento (MARTINEZ; TACCA, 2009).

Estudos sobre a Aprendizagem Criativa têm destacado a pouca frequência deste tipo de aprendizagem nas instituições do sistema educacional brasileiro que, pela forma como se

constituiu historicamente, caracterizou-se e caracteriza-se por uma educação homogeneizante, orientada para a concepção assimilativo-reprodutiva do conhecimento (ALENCAR, 2009; AMARAL, 2011).

Para Martinez e Tacca (2009), apesar de pouco estimulada nas instituições de ensino, a Aprendizagem Criativa deveria ser priorizada por, no mínimo, duas razões: (1) a estabilidade do aprendido e suas possibilidades de transferência a novos contextos; (2) o seu potencial como unidade de desenvolvimento da condição de sujeito no processo de aprender ou em alguma atividade que o aprendiz desenvolva.

2.5 CONSIDERAÇÕES SOBRE O CAPÍTULO

Neste capítulo, foram destacados os quatro Pilares do Pensamento Computacional e observou-se que o objetivo do ensino do PC está relacionado com a sua perspectiva teórica, a qual pode ser Cognitiva, Situada ou Crítica.

Adicionalmente, compreendeu-se que a Teoria Neo-Piagetiana, no contexto de Programação, classifica os programadores iniciantes em Estágios Cognitivos de acordo com as fragilidades do aprendizado de programação.

A partir destes conceitos, entende-se que os Pilares do Pensamento Computacional podem ser usados em conjunto com a perspectiva teórica para o desenvolvimento do PC e na resolução de problemas em conjunto com a Aprendizagem Criativa, uma vez que ela está relacionada com a solução de problemas.

3 TRABALHOS RELACIONADOS

Fonte: *Tumblr* Tiras Armandinho (2015)[3].

Diversas abordagens têm sido usadas para o desenvolvimento do Pensamento Computacional no contexto da Resolução de Problemas. Dentro dos contextos nacional e internacional, a pesquisa realizada detectou estratégias como: uso de linguagem visual no desenvolvimento do PC, classificação de Estágios Cognitivos através dos códigos dos estudantes e utilização da criatividade no ensino.

Uma visão abrangente dos trabalhos relacionados ao problema discorrido nesta tese compõe o presente capítulo, que se encontra estruturado em duas seções, de acordo com o assunto sobre o qual cada trabalho se relaciona: Desenvolvimento das Habilidades do Pensamento Computacional, Ensino e Aprendizagem Criativos e Teoria Neo-Piagetiana.

3.1 DESENVOLVIMENTO DAS HABILIDADES DO PENSAMENTO COMPUTACIONAL, ENSINO E APRENDIZAGEM CRIATIVOS

Diferentes estratégias têm sido usadas para ajudar no desenvolvimento do Pensamento Computacional em estudantes. O estudo de Troiano et al. (2019) se concentrou em investigar até que ponto as métricas do Pensamento Computacional podem ser usadas para examinar o seu desenvolvimento e fornecer *insights* sobre como os alunos desenvolvem as habilidades do Pensamento Computacional ao longo dos projetos criado com *Scratch*. As métricas usadas no estudo para avaliar as competências de programação dos alunos foram: 1) conceitos de programação (por exemplo, uso de expressões condicionais); (2) organização de código (por exemplo, nomes de *sprites*); e (3) *design* para usabilidade (por exemplo, entrada de dados pelo usuário).

[3]Disponível em: https://tirasarmandinho.tumblr.com/post/113819416619/tirinha-original. Acesso em: 16 jan. 2021.

Assim, a análise verificou quais dimensões são desenvolvidas com mais proficiência e quais parecem mais difíceis de serem adquiridas através do desenvolvimento de jogos sérios criados pelos alunos. De acordo com os resultados do estudo de Troiano et al. (2019), o *Game Design* pode ser eficaz para apoiar os alunos no desenvolvimento da proficiência em PC nos currículos STEM (*Science, Technology, Engineering e Mathematics*). Ou seja, os jogos projetados pelos alunos mostraram alta proficiência e desenvolvimento do Pensamento Computacional. Embora a avaliação tenha sido feita ao longo do projeto, ela não usa os pilares do Pensamento Computacional para avaliar a solução de problemas e não a relaciona às habilidades do PC.

O trabalho de Araujo, Primo e Pernas (2020) teve o intuito de armazenar de forma estruturada o percurso dos estudantes durante o desenvolvimento dos projetos no *Scratch*, com o objetivo de investigar seu processo de aprendizagem das habilidades do Pensamento Computacional. Ao final do estudo, os autores concluíram que sua proposta seria viável para representação dessas habilidades com resultados equivalentes a outras ferramentas.

A pesquisa de Gal et al. (2017) tem o objetivo de explorar as características da criatividade e analisá-las por meio de indicadores automatizados entre pessoas na faixa etária de 6 a 18 anos, em um ambiente *on-line* para aprendizagem do Pensamento Computacional. Segundo a análise, a criatividade é influenciada por traços de personalidade e pela motivação intrínseca da tarefa, apresentando maior indicativo de criatividade.

O estudo realizado por Shell et al. (2017) abordou a integração de Pensamento Computacional e Pensamento Criativo em cursos de Ciência da Computação para melhorar a aprendizagem e o desempenho de alunos do ensino superior usando Exercícios de Criatividade Computacional (CCEs). Esta pesquisa usa a Teoria da Geração de Epstein (2005) para apoiar a definição de Pensamento Criativo, que o divide em quatro competências: Capturar, Desafiar, Ampliar e Engajar. A primeira refere-se à capacidade de reconhecer e anotar ideias únicas à medida em que ocorrem, enquanto a segunda, como o nome sugere, refere-se à capacidade de desafiar o pensamento e os padrões de comportamento estabelecidos, estando relacionado à capacidade de gerar novas abordagens para os problemas. A terceira, por sua vez, identifica a capacidade de estender o conhecimento para além da disciplina, permitindo a integração inovadora de ideias e a última envolve um estímulo que pode ser social ou ambiental, podendo levar a novas experiências e ideias.

Os princípios dos Exercícios de Criatividade Computacional (CCEs) são (1) equilíbrio de atributos entre Pensamento Computacional e Criativo e (2) mapeamento entre conceitos e habilidades computacionais e criativas, conforme manifestado em diferentes disciplinas. Para

cada exercício, o estudo tem um conjunto de objetivos criativos, objetivos computacionais e objetivos colaborativos de resolução de problemas. Para o conjunto de objetivos computacionais, são utilizados dois aspectos: conceitos de PC, como classificação e condição lógica, bem como habilidades de Pensamento Computacional. As habilidades de Pensamento Computacional utilizadas no estudo foram: Decomposição de Problemas, Reconhecimento de Padrões, Abstração, Generalização, *Design* Algorítmico e Avaliação.

Assim, o estudo concluiu que a integração de exercícios de Criatividade Computacional baseados nas Competências Criativas de Epstein (2005) melhorou o aprendizado dos alunos nos cursos de Pensamento Computacional em Ciência da Computação. O trabalho de Shell et al. (2017) aponta a necessidade de relacionar o ensino de Pensamento Computacional e Criatividade na disciplina de Computação a fim de ajudar os alunos a aprenderem e desenvolverem sua capacidade de aplicar, de forma criativa, os conhecimentos do Pensamento Computacional na resolução de problemas. Todavia, a pesquisa não mostra claramente a relação entre a resolução de problemas e os pilares do Pensamento Computacional, não identificando em que nível de desenvolvimento de PC os alunos se encontram. Dessa forma, devido à falta de experiência de ensino, vários problemas foram encontrados no desenvolvimento do curso, tais como falta de controle do tempo de conclusão do projeto, além da não garantia de imparcialidade das realizações dos alunos.

3.1.1 Evidências Nacionais

Com o objetivo de identificar os estudos relacionados no contexto nacional, realizou-se uma busca por artigos, considerando o Portal de Publicações da CEIE e do WEI, no período de 2017-2021 por meio da *string* “pensamento computacional” *OR* “raciocínio computacional” *OR* “programação” *AND* “resolver problemas” *OR* “solucionar problemas” *OR* “criatividade”. Adotando estes requisitos, nove trabalhos foram encontrados. O critério de exclusão utilizado removeu os estudos que não falavam sobre resolução de problemas para o desenvolvimento das habilidades do Pensamento Computacional. Com isso, restaram três trabalhos no contexto nacional, os quais são discorridos a seguir.

Em relação a estudos sobre resolução de problemas através de questões a serem respondidas e desenvolvimento das habilidades do Pensamento Computacional, o trabalho de Pessoa et al. (2017) propõe um aplicativo para celular “gamificado”. O público-alvo é composto por estudantes do ensino médio e as questões abordam as habilidades de PC: abstração, decomposição, coleta de dados e análise de dados.

A utilização de ferramentas para avaliar as habilidades do PC é tratada na pesquisa de Santana e Oliveira (2019). Eles utilizaram o Dr. *Scratch* para avaliar essas habilidades de 137 projetos de *quizzes* criados pelos alunos do ensino fundamental II.

Os resultados da pesquisa de Santana e Oliveira (2019) mostraram que houve desempenho significativo em aspectos importantes para o desenvolvimento do PC, tais como: lógica, interatividade com o usuário, representação de dados e sincronização, apresentando a necessidade de melhorar as habilidades do PC como paralelismo, controle de fluxo e abstração.

Oliveira et al. (2019) apresentam uma proposta de desenvolvimento do PC através da resolução de problemas para 98 alunos da disciplina de Introdução à Ciência da Computação do curso de Bacharelado em Ciência da Computação. Os desafios propostos tiveram o intuito de exercitar as habilidades do Pensamento Computacional, considerando abstração, decomposição, reconhecimento de padrões, formulação de algoritmos, simulação e otimização. As atividades foram desplugadas e, devido a isto, nem todos os alunos tiveram a correção realizada em sala.

3.1.2 Evidências Internacionais

No contexto internacional foram consideradas as bases ACM *Digital Library*, IEEE *Xplore* e *ScienceDirec*t. A *string* usada na busca foi "*computational thinking*" *OR* "*programming*" *AND* "*solving problems*" *OR* "*creativity*" *OR* "*problem-solving*" para artigos dos últimos cinco anos (2017-2021), tendo sido aplicada nos campos título e resumo, com a obtenção de 110 estudos no resultado. Destes, adotando o critério de exclusão utilizado, foram removidos os estudos que não falavam sobre resolução de problemas para o desenvolvimento das habilidades do Pensamento Computacional ou programação ou criatividade, selecionando-se doze, os quais são descritos a seguir.

A pesquisa de Wagh, Gravel e Tucker-Raymond (2017) investiga as atividades de um grupo de jovens criadores de artefatos digitais interativos. O estudo teve o objetivo de responder às perguntas: quais as formas de Pensamento Computacional dos jovens no projeto e de que maneiras essas práticas são desenvolvidas em seu trabalho? Os dados incluem dados etnográficos qualitativos, tais como observações, gravações e entrevistas. O artigo analisou as maneiras de práticas do Pensamento Computacional dos aprendizes e descreveu algumas formas pelas quais essas práticas foram desenvolvidas em seu trabalho. Os resultados mostraram que os alunos usaram decomposição, simulação e solução de problemas e depuração. Ao implementar essas práticas, o grupo também se envolveu na construção da lógica por meios

de comandos condicionais envolvidos no funcionamento de seu projeto.

O estudo de Boom et al. (2018) tem o objetivo de investigar a relação entre PC e inteligência como uma habilidade de resolução de problemas. Para tanto, 71 alunos fizeram vinte tarefas *on-line* com base no *Hallenge Bebras* e no Teste de Inteligência Não-verbal (3ª edição). Os resultados apresentaram uma relação linear positiva significativa entre PC e inteligência. À medida em que a capacidade do Pensamento Computacional aumentou, a inteligência também aumentou. Diante destes fatos, a pesquisa de Boom et al. (2018) apresentou algumas suposições: i) existe uma forte relação entre PC e inteligência, pois a definição e a avaliação dos construtos não são suficientemente diferenciadas e, consequentemente, o PC pode ser reformulado para se distinguir mais de outros conceitos cognitivos; ii) o relacionamento não é novidade, uma vez que ambos os conceitos estão naturalmente relacionados. A inteligência apenas abrange diferentes habilidades cognitivas e o PC pode ser parte de uma inteligência geral, o que pode ter um impacto em como se pensa o PC e seu desenvolvimento. O estudo informa que é preciso investigar esta última relação mais complexa e pesquisas com instrumentos mais sofisticados são necessárias para examinar mais extensivamente o lugar do PC entre outras habilidades cognitivas.

A pesquisa de Tyrén et al. (2018) apresenta o BBC micro: bit, um microcontrolador, que pode ser programado e conectado a outros dispositivos ou sensores como *Arduino* e *Raspberry Pi*. O estudo teve o intuito de estimular os jovens a serem criativos no mundo digital, desenvolvendo habilidades essenciais em assuntos STEM e produzindo uma nova geração de inventores e criadores. Foram realizadas 21 intervenções em *workshops* para coletar dados qualitativos e obter *insights* sobre as interações de estudantes e docentes com materiais de ensino micro: bit da BBC. O resultado foram algumas formas de intervenções consideradas relevantes para o ensino de Pensamento Computacional:

- *Tinkering*: permite familiarizar-se livremente com o novo conteúdo antes do exercício;
- *Stupid Computers*: esclarece que os computadores só fazem o que é instruído;
- *Text based instructions*: instruções baseadas apenas em texto podem não ser a melhor maneira de transmitir exercícios. Dar instruções por outros meios pode ser potencialmente mais bem-sucedido;
- *Editor navigation*: um passo a passo inicial do editor pode ajudar os alunos a entenderem como ele funciona e como usá-lo;
- *Self instructing materials*: alternativa para fornecer ajuda individual a grandes

grupos;

- *End on a positive note:* refere-se a sessões de *workshop* de conclusão com um exercício que deixa os alunos em um estado de espírito positivo, em vez de deixá-los confusos ou frustrados.

Os autores esperam que os resultados possam ser úteis para professores que estão desenvolvendo materiais de ensino para o Pensamento Computacional no novo currículo da Suécia (TYRÉN et al., 2018).

Carlborg et al. (2018) apresentam um estudo colaborativo entre pesquisadores universitários e um instituto de pesquisa da Suécia com o objetivo de investigar o que é importante considerar ao projetar materiais de ensino com o micro: bit para o treinamento de habilidades de Pensamento Computacional de estudantes do ensino fundamental. Por meio de um processo iterativo, um total de 21 intervenções em oficinas foram conduzidas para coletar dados qualitativos e obter *insights* sobre as interações dos estudantes do ensino fundamental e de seus professores com materiais de ensino micro: bit.

Os resultados do projeto sugerem que é importante que os docentes considerem a quantidade de escolhas livres que são dadas aos estudantes em qualquer atividade ao projetarem materiais de ensino com a plataforma usada para treinar habilidades de PC dos alunos do ensino fundamental. A pesquisa propõe um modelo de Escopo de Autonomia que se baseia nas relações observadas entre o potencial de aprendizagem dos alunos, o risco de se sentirem sobrecarregados e a quantidade de opções oferecidas nas atividades atenderem aos requisitos de programação do novo currículo.

O trabalho de Izu e Alexander (2018) apresentou o desenho de um curso de programação com ênfase na solução de problemas. Neste curso, os alunos têm uma grande quantidade de prática de programação desafiadora. Além disso, foram utilizados métodos de regressão não linear estocástica para identificar fatores que são mais fortemente associados de forma confiável com a fluência nas atividades práticas. A variedade de problemas e a necessidade de melhorar a fluência do código foi um aspecto principal que motivou os alunos a fazerem as atividades.

Esta prática teve um impacto claro nas percepções dos discentes sobre sua abordagem para a resolução de problemas. De acordo com as reflexões de Izu e Alexander (2018), muitos estudantes adquiriram uma forte apreciação do valor do planejamento da solução durante o curso. Para os autores, por causa da natureza repetitiva da prática e a variedade de problemas que resolveram, os discentes precisaram reavaliar seus hábitos de codificação e usar o planejamento como a primeira etapa de qualquer desenvolvimento de *software*. Os dados de submissão durante os exames práticos indicaram que a fluência na codificação melhorou ao

longo do curso, também indicando como aspectos da prática nas semanas próximas à prova e no início do curso estão associados à fluência nas avaliações.

Vourletsis e Politis (2020) investigaram as percepções dos alunos do ensino fundamental sobre suas habilidades de resolução de problemas, antes e depois de um curso experimental de PC, com o intuito de analisar o efeito sobre autopercepção, autoconfiança, autocontrole e indicadores que mostram quanto os discentes evitam enfrentar problemas e a correlação entre as variáveis das percepções dos alunos sobre sua habilidade de resolução de problemas. Segundo a análise dos dados, o curso de PC pode ajudar a melhorar as percepções gerais dos estudantes sobre suas habilidades de resolução de problemas, mas parece não ter qualquer efeito estatisticamente significativo em sua confiança na resolução de problemas. Além disso, o curso pode ser benéfico para os níveis de autocontrole dos estudantes quando eles são confrontados com problemas e incentivá-los a não os evitar. Ademais, aumentos nas pontuações das variáveis de autocontrole dos participantes foram correlacionados com aumentos em sua vontade de enfrentar seus problemas ao invés de evitá-los.

Cachero et al. (2020) apresentam um estudo que demonstra empiricamente o impacto que um curso de programação pode ter na autopercepção Pensamento e objetivos do PC. A pesquisa afirma que as capacidades do PC não são desenvolvidas naturalmente, mas precisam ser treinadas. Os dados também apoiam a ideia difundida de que a inscrição em cursos de programação aumenta as habilidades do PC. Estes resultados podem ser explicados por dois fatos amplamente aceitos: as pessoas tendem a superestimar suas habilidades e os jovens têm lacunas de habilidades digitais.

Além disso, o estudo apresentou evidências do impacto da alfabetização em informática na experiência de aprendizagem do aluno e como essas habilidades e do PC são necessárias para poder estudar, trabalhar e viver em um mundo digital. Segundo Cachero et al. (2020), a comunidade de pesquisa ainda não foi capaz de fornecer orientação para o processo de decidir qual curso ou abordagem de atividade seria mais eficaz, tanto em termos de melhoria do PC, quanto de aumento da motivação do aluno em diferentes contextos. Em vista disso, os autores defendem o uso de um modelo orientado a problemas com ênfase na descoberta científica e princípios computacionais, que inclui problemas que estão diretamente relacionados para a área de interesse do aluno.

A pesquisa de Lin e Chen (2020) apresenta a aprendizagem de uma linguagem de programação através de um sistema de Realidade Aumentada (RA) baseado em recomendação de aprendizagem com estudantes universitários. No estudo, analisou-se o desempenho da aprendizagem e a capacidade de raciocínio computacional dos alunos após as atividades. Para

os autores, esta abordagem de aprendizagem pode resolver efetivamente as dificuldades de linguagem de programação dos estudantes em sintaxe e a lógica pode ser aprimorada nas atividades. Através da recomendação de aprendizagem, pode-se refletir sobre os mitos e mal-entendidos na aprendizagem e permitir que os discentes mudem da aprendizagem passiva do conhecimento para a compreensão ativa dos problemas, pensando sobre os problemas e resolvendo-os.

Destarte, Jin, Johnson e Dekhane (2020) realizaram um estudo com o intuito de ajudar os alunos a desenvolverem uma melhor compreensão das estratégias de resolução de problemas. Para isto, foram criadas atividades que envolveram os alunos diretamente em todos os aspectos da resolução de problemas. Primeiro, os estudantes analisaram completamente programas que foram usados de exemplo e identificaram as estratégias de resolução de problemas usadas nesses programas. Dessa maneira, os discentes resumiram suas descobertas e registraram suas estratégias. Em seguida, aplicaram essas estratégias para resolver outros problemas semelhantes. Este processo iterativo requer que os alunos analisem, reflitam e sintetizem os níveis mais elevados de habilidades cognitivas na Taxonomia de Bloom, que levam a uma aprendizagem mais relevante e a uma obtenção de conhecimentos e habilidades para uma maior variedade de tarefas e contextos.

O estudo escolheu três tópicos principais para abordar nas aulas e atividades de introdução à programação. Os três tópicos são básicos (variáveis, entrada e saída), instruções if e *loop*, este último tratando-se de um tópico difícil para os estudantes, sendo observado que é por meio desse assunto que os alunos parecem começar a ter dificuldades ou até mesmo pensar em desistir.

Como conclusão, Jin, Johnson e Dekhane (2020) afirmaram que uma melhor compreensão das estratégias de resolução de problemas deve contribuir mais diretamente para a motivação na conclusão das tarefas e confiança na tentativa de resolver problemas de programação do que outras medidas atitudinais. Além disso, os dados de desempenho mostraram que a abordagem é muito eficaz para o assunto de *looping* e que os alunos também apresentaram maior motivação e confiança.

Enquanto isso, Kovalkov, Segal e Gal (2021) desenvolveram um *Combined Creativity Score* (CCS) em sua pesquisa para medir a criatividade em projetos *Scratch*, com base na combinação de análise de dados e aspectos de criatividade da literatura. Para o estudo, os autores aplicaram o CCS em mais de 200 projetos *Scratch* e compararam essa medida com a pontuação de criatividade realizada por professores de *Scratch*. Os resultados indicaram uma concordância de classificação de t = 0: 435 ou superior entre os professores do CCS.

Já Castro et al. (2021) investigaram a progressão das práticas computacionais, através de rubricas, no contexto do primeiro ano de um curso de Engenharia. Neste estudo, obtém-se a caracterização de quatro perfis dos estudantes: perfil 1) alto desempenho e nível baixo na solução de problemas; perfil 2) desempenho intermediário e nível avançado para solução de problemas; perfil 3) desempenho intermediário e nível baixo de resolução de problemas; e perfil 4) alunos com nível baixo solução de problemas. De acordo com esta classificação, a pesquisa permitiu identificar como os estudantes progrediram ao longo do semestre.

Castro et al. (2021) ainda apresentaram que os alunos que não dominavam as práticas iniciais associadas à preparação de problemas para soluções computacionais tiveram dificuldade, posteriormente, em desenvolver práticas relacionadas à implementação real das soluções computacionais. No entanto, o estudo vai além dessa constatação para também identificar uma prática específica na solução de problemas e depuração que criou uma lacuna crítica em alunos que tiveram um desempenho semelhante na fase inicial do curso (Perfil 2 e Perfil 3).

Portanto, esta descoberta sugere que os alunos devem compreender a prática de resolução de problemas computacionais e depuração no início dos cursos introdutórios. Ademais, com informações do início do semestre, foi possível identificar os estudantes que pertenciam a um grupo de baixo desempenho e que apresentaram progressão de baixo desempenho nas práticas avaliadas posteriormente. Considerar as experiências anteriores dos alunos, em uma primeira aula de programação, ajudará os instrutores a alavancarem o conhecimento de práticas individuais que podem incluir diferentes experiências de aprendizagem.

3.2 TEORIA NEO-PIAGETIANA NO ENSINO E APRENDIZAGEM DE PROGRAMAÇÃO

Diversas estratégias têm sido usadas para compreender e classificar as fragilidades no aprendizado de programação de estudantes. A Teoria Neo-Piagetiana aplicada no contexto da programação é uma delas e será apresentada a seguir.

A pesquisa desenvolvida por Culligan e Casey (2021) utilizou dados de aproximadamente 300 estudantes na disciplina de Introdução à Programação através da plataforma *MULE*. Estes dados incluem o desempenho em atividades práticas e questões de pseudocódigo. As avaliações são realizadas automaticamente e o *feedback* é dado aos alunos sobre seu código. As questões de pseudocódigo são usadas para examinar a compreensão dos alunos dos conceitos de programação, removendo o uso do compilador e comparando seu

desempenho em questões de pseudocódigo com os dados da plataforma. Usando uma estrutura Neo-Piagetiana, foram analisados os desempenhos do pseudocódigo e do exame de laboratório, bem como o comportamento do programador através da plataforma. Avaliou-se ainda a progressão dos estudantes por meio dos três Estágios Cognitivos neo-piagetianos: Sensório-Motor, Pré-Operacional e Operacional Concreto e criou classificadores binários *Deep Neural Net* para testar se esses dados comportamentais coletados passivamente podem ser usados para detectar alunos em risco de reprovação. A conclusão do estudo é que a divergência em comportamento entre alunos de alto e baixo desempenho acontece na semana 4. Estudantes que não estão apresentando indícios de progressão para o estágio Pré-Operacional de aprendizagem Neo-Piagetiana não se dão bem nas atividades do laboratório no final do semestre.

Recentemente, Rubio (2019) apresentou um estudo que tem o intuito de categorizar automaticamente estudantes de Introdução à Programação usando técnicas de aprendizado de máquina. Neste estudo, foram utilizados exercícios de programação para agrupar os discentes em diferentes *clusters* de acordo com suas pontuações e associar esses *clusters* aos diferentes estágios Neo-Piagetianos do programador iniciante. Foram criados três *clusters* que apresentaram características que propiciaram associá-los a diferentes estágios Neo-Piagetianos. Os estudantes que pertencem ao *cluster* Operacional Concreto são capazes de usar a inferência transitiva e o código reverso. Possuir essas habilidades apoia fortemente a ideia de que os discentes alcançaram o estágio Operacional Concreto. O resultado foi a configuração de *clusters* altamente estáveis, o que é consistente com o modelo Neo-Piagetiano. Os *clusters* mostram taxas de conclusão significativamente diferentes para diferentes tipos de perguntas. Essas diferenças permitiram vincular cada *cluster* a um estágio Neo-Piagetiano. Os resultados sugerem que o modelo Neo-Piagetiano pode ser uma ferramenta útil para projetar exercícios para analisar automaticamente o aprendizado de programadores iniciantes ao longo do curso.

Kutscha (2017), por sua vez, usou o método de categorização, bem como um catálogo de exercícios para o professor, com o intuito de classificar o nível de raciocínio dos estudantes por meio da utilização de testes escritos. Esta abordagem é investigada em um estudo de caso. Mostra-se que os testes escritos, que são elaborados com base na proposta de abordagem, nem sempre conduzem à determinação correta do nível de raciocínio. As razões para isto, ou seja, estratégias criativas dos alunos para ganhar mais pontos nos exercícios, são discutidas e melhorias a essa abordagem são propostas.

Em resumo, esse estudo de caso mostra um exemplo de como a Teoria de Lister pode ser utilizada para determinar o neo Estágio piagetiano de um novato. O método descrito implica o seguinte: os primeiros exercícios existentes ou recém-escritos são categorizados usando o

método de categorização proposto. Então um teste escrito é criado a partir desses exercícios para os alunos. Os resultados desta prova escrita são, então, analisados e categorizados da forma proposta de forma a acolher a fase neo-piagetiana dos alunos. Com essas informações, os alunos podem ser auxiliados individualmente com exercícios e explicações correspondentes ao seu nível de raciocínio.

A estrutura Neo-Piagetiana de desenvolvimento cognitivo é abordada por Chuechote et al. (2020), que fornecem uma explicação teórica sobre como estudantes do ensino médio estabelecem algoritmos de classificação como resultado da jogabilidade. Vinte e três estudantes jogaram o *game* com questões sobre programação em várias fases. Para refletir posteriormente sobre seu raciocínio operacional e, portanto, a tomada de decisão, a série de ações do jogo foi registrada para dados empíricos individuais.

Os resultados das relações entre ações e raciocínio esquemático para resolver problemas de classificação sugere o impacto de um jogo digital no desenvolvimento do pensamento algorítmico e, em geral, o uso de um jogo para auto-aprendizagem de conceitos computacionais. O jogo forneceu a experiência de aprendizagem, onde foram atribuídos significados às mudanças nos Estágios Cognitivos, propiciando uma experiência de classificação em que os próprios participantes aprenderam em seu próprio ritmo. A limitação das ações, apenas trocar, comparar ou submeter, criou um quadro de pensamento que era voltado para o pensamento sistemático para a organização do esquema.

3.3 CONSIDERAÇÕES SOBRE O CAPÍTULO

A partir do levantamento realizado, foi possível traçar caminhos para a realização desta pesquisa. Alguns pontos em comum nos artigos foram verificados como: i) a falta de clareza na relação entre a resolução de problemas e os pilares do Pensamento Computacional, não identificando em que nível de desenvolvimento do PC os estudantes se encontram; ii) a falta de intervenções e atividades direcionadas a melhorar o desenvolvimento das habilidades do PC dos estudantes, detalhando em qual habilidade possuem dificuldades; iii) a falta de mapeamento das dificuldades encontradas pelos estudantes na resolução de problemas e a sua relação com o pilares do PC; iv) a falta de análise dos códigos desenvolvidos pelos estudante com o intuito de entender em qual estágio cognitivo eles estão e quais níveis das habilidade do PC os estudantes estão.

Diante das possibilidades e limitações dos estudos apresentados, esta tese apresenta o Modelo Conceitual *Skills-CT* que trata algumas lacunas observadas e citadas anteriormente.

Adicionalmente, o *Skills-CT* tem o objetivo de ajudar o(a) professor(a) a avaliar os pilares e as habilidades do Pensamento Computacional e Estágios Cognitivos e estimular o desenvolvimento da aprendizagem criativa dos estudantes.

4 *SKILLS-CT*: UM MODELO CONCEITUAL DE APRENDIZAGEM CRIATIVA E RESOLUÇÃO DE PROBLEMAS

Fonte: *Instagram* de Guilherme Bandeira (2021)[4].

Esse capítulo apresenta as etapas do Modelo Conceitual, que se baseia nos pilares do Pensamento Computacional, nas habilidades de PC e Estágios Cognitivos, e sua utilização através do Guia de Ensino de Programação para professores, que explica a forma de aplicação do modelo, bem como sugere ferramentas para cada uma das etapas.

4.1 MODELO CONCEITUAL SKILLS-CT

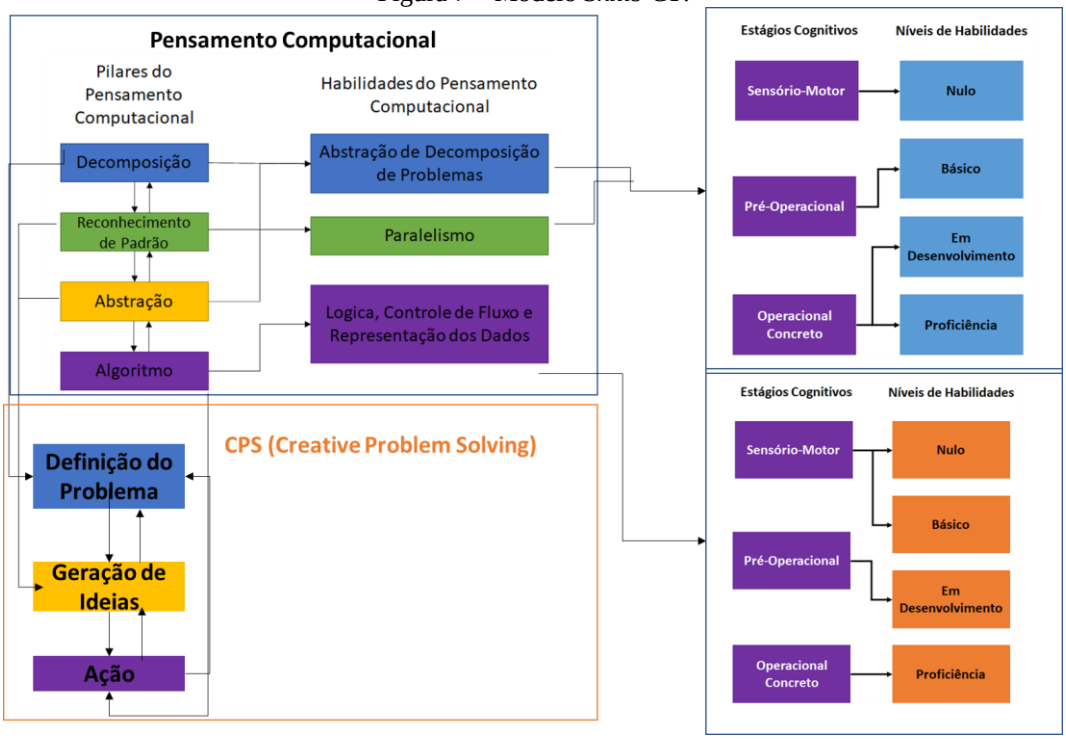
O Modelo *Skills-CT* objetiva auxiliar o(a) professor(a) a avaliar pilares e habilidades de Pensamento Computacional e Estágios Cognitivos de forma personalizada, estabelecendo suas classificações. A partir dos resultados, a ideia é propor atividades e preparar aulas de acordo com os níveis e habilidades identificados, visando diminuir a dificuldade de aplicar os conhecimentos na Resolução de Problemas, além de promover o incremento do Estágio Cognitivo do estudante.

O Modelo *Skills-CT* (FIGURA 7) propõe três relações: a dos Pilares do PC com as habilidades, os Estágios Cognitivos com os Níveis das Habilidades do PC e dos Pilares do Pensamento Computacional com o CPS (*Creative Problem Solving*). Os detalhes das relações do Modelo

[4]Disponível em: https://www.instagram.com/p/CMzlzq9n6qm/?igshid=8a822pw5oy0. Acesso em: 16 jan. 2021.

Skills-CT são apresentados nas Figuras 7, 8, 9 e 10.

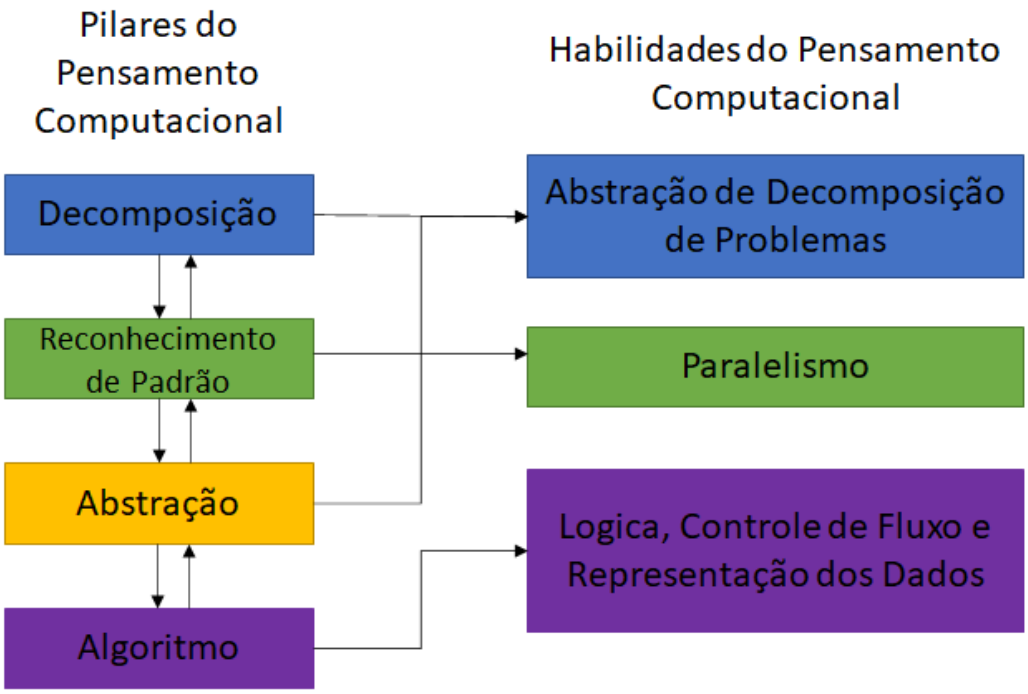
Figura 7 – Modelo *Skills-CT*.



Fonte: A autora (2021).

A Figura 8 refere-se à Decomposição, a primeira relação do Modelo *Skills-CT*, tendo como objetivo dividir os problemas em partes menores para resolvê-los.

Figura 8 – Relação entres os Pilares do PC e Habilidades.



Fonte: A autora (2021).

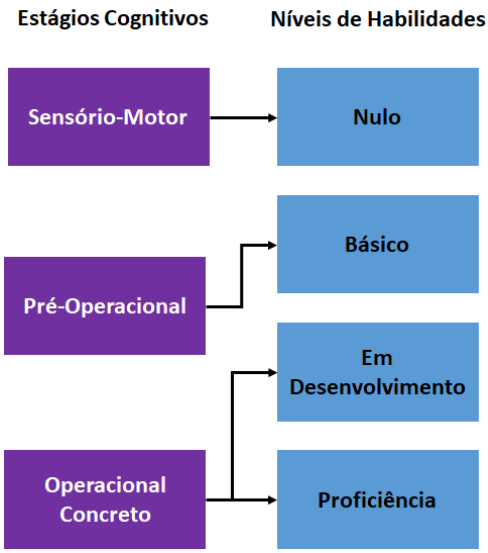
O Reconhecimento de Padrão visa observar semelhanças entre problemas e subproblemas (FIGURA 8). Ele está relacionado ao Paralelismo, que permite a execução de um *script* simultaneamente. A Abstração refere-se a isolar aspectos importantes de um todo e considerá-los individualmente. Na resolução do problema, é avaliada e extraída uma representação dele levando em consideração um aspecto para propor a solução. Por fim, o Algoritmo está relacionado à solução. Desse modo, as habilidades de Pensamento Computacional que estão vinculadas são lógica, controle de fluxo e representação de dados por estarem relacionados à criação do algoritmo usando condicionais, *loop*, variáveis, listas e *arrays*.

A Figura 9 apresenta a relação dos Estágios Cognitivos com os níveis das Habilidades de Abstração e Decomposição de Problemas e Paralelismo. O nível nulo da habilidade de Abstração e Decomposição de Problemas está relacionado ao Estágio Cognitivo Sensório-Motor. Os discentes deste Estágio Cognitivo têm dificuldade em decompor a solução em problemas menores por não conseguirem entender a sequência do código e rastreá-la.

A Abstração e a Decomposição de Problemas, se estiverem no nível básico, estão relacionadas ao Estágio Cognitivo Pré-Operacional em decorrência da dificuldade de abstrair corretamente além do próprio código.

Já se a Abstração e a Decomposição do Problema estiverem no nível de habilidade em desenvolvimento e proficiência, o aluno está no Estágio Cognitivo Operacional Concreto. Isto significa que o estudante pode raciocinar sobre o código tendo consciência de que pode ser uma combinação da divisão de partes menores que executam funções específicas.

Figura 9 – Estágios Cognitivos X Níveis de Habilidades de PC
para Abstração e Decomposição de Problemas e Paralelismo.



Fonte: A autora (2021).

Na relação entre a habilidade de Paralelismo e o Estágio Cognitivo (FIGURA 8), se o nível for nulo, ele se refere ao Estágio Cognitivo Sensório-Motor, pois o aluno que se encontra nesse estágio não cria programas que possam ser divididos em partes que possam ser executadas simultaneamente.

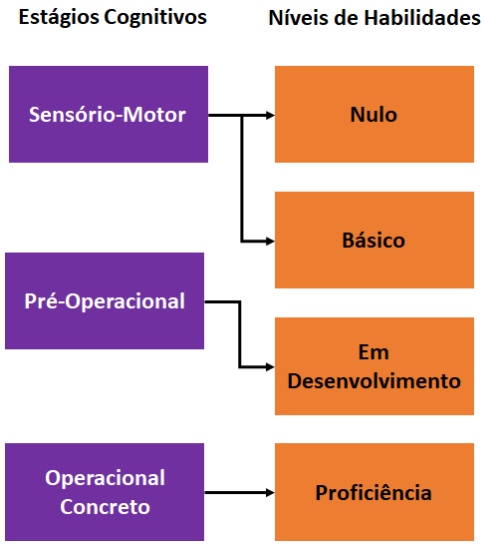
O Estágio Cognitivo do aluno é classificado como Pré-Operacional na habilidade de Paralelismo (FIGURA 9) quando ele pode entender a divisão do código em partes e usa função.

O Estágio Cognitivo Operacional Concreto e sua relação com o Paralelismo (FIGURA 9) acontecem quando o nível de habilidade está em desenvolvimento e proficiência – neste o aprendiz usa o retorno de uma função como parâmetro de outra função e/ou uma função chamando outra.

A Figura 10 mostra a relação entre os Estágios Cognitivos e a Lógica, Controle de Fluxo e Representação dos Dados. Na habilidade de Lógica, o nível do discente é Sensório-Motor quando o nível é nulo ou básico, ou seja, o estudante utiliza estrutura condicional apenas para comparar valores, mas não usa operadores lógicos em sua solução.

O Estágio Cognitivo Pré-Operacional está relacionado à Lógica no nível de desenvolvimento (FIGURA 10) quando o discente pode compreender e utilizar um recurso que indica quais instruções o programa deve processar de acordo com uma expressão booleana. Por fim, quando o estudante está no nível de proficiência em Lógica, pode usar operadores lógicos e seu Estágio Cognitivo é Operacional Concreto. Neste estágio, o estudante pode raciocinar sobre a relação entre duas ou mais variáveis.

Figura 10 – Estágios Cognitivos X Níveis de Habilidades de PC Lógica, Controle de Fluxo e Representação dos Dados.



Fonte: A autora (2021).

Para a relação dos níveis de habilidade de Controle de Fluxo (FIGURA 9) com os Estágios Cognitivos, considera-se que o estudante está no Estágio Cognitivo Sensório-Motor quando a habilidade é nula ou básica. Assim, o código produzido por um aluno neste Estágio Cognitivo não possui *loops*, visto que existe uma dificuldade de visualização da estrutura de dados e do conceito de uma única célula dentro dela. Além disso, existe uma dificuldade em compreender a diferença entre o índice e o valor (ou conteúdo) do elemento em um vetor.

Sobre o Estágio Cognitivo Pré-Operacional e sua relação com o Controle de Fluxo, se o aluno está em nível de desenvolvimento, ele usa a repetição mesmo com dificuldade em modificar valores dentro de um *loop* e rastrear essa mudança; o discente pode usar *loops* quando não há mudança de valores do início ao fim da iteração.

Por fim, o Estágio Cognitivo Operacional Concreto está presente quando o aluno usa a estrutura de repetição. Isto se deve ao fato de que os estudantes deste nível têm uma compreensão mais avançada de *loop* e compreendem os estados inicial e final, podendo raciocinar sobre a transformação de variáveis em iterações gerais de *loop*.

A classificação do Estágio Cognitivo Sensório-Motor para Representação de Dados é apresentada quando o nível é Nulo ou Básico. Assim, a utilização, a compreensão e a relação entre as variáveis e seus valores e o conceito da ordem de execução das linhas de código é uma dificuldade encontrada pelo estudante que se encontra neste Estágio Cognitivo (FIGURA 10).

Os alunos que estão no Pré-Operacional, para Representação de Dados, conseguem entender a atribuição de valores de uma variável apesar de ainda terem dificuldades de compreender a alteração de uma variável em um *loop*. Assim, considera-se que estejam no nível em desenvolvimento. Os estudantes do Pré-Operacional conseguem analisar a primeira e a última iteração de um *loop* e possuem dificuldade nas iterações intermediárias.

O Estágio Cognitivo Operacional Concreto do discente está relacionado à Representação dos Dados caso utilize uma lista e/ou *array*. Um discente nesse estágio consegue modificar os valores em uma lista e/ou *array* e é capaz de modificar variáveis em todas as iterações do *loop*.

Adicionalmente, o Modelo *Skills-CT* apresenta a relação entre os pilares do Pensamento Computacional e CPS (*Creative Problem Solving*) que destina-se a auxiliar na solução de problemas para facilitar o aprendizado da programação de forma criativa, adicionada de forma implícita na atividade proposta no Guia para Ensino de Programação apresentado na seção 4.2.

Para melhor compreensão, o *Creative Problem Solving* (CPS) foi criado por Osborn (2008) e consiste em um paradigma metodológico composto por métodos e técnicas para analisar, identificar e solucionar problemas (OSBORN, 2008).

Este método é formado por três etapas: a Investigação, a Descoberta de Ideias e a Descoberta de Soluções (OSBORN, 2008). A estratégia do modelo é obter uma clara e precisa definição do problema e gerar várias soluções. Seus elementos e fases são apresentados no Quadro 6.

Na fase da Investigação, é delimitado o problema. Segundo Osborn (2008), a definição do problema é fundamental para propor novas questões e possibilidades. A preparação do problema envolve a pesquisa e recolha de informações relacionadas com o problema a resolver e a sua posterior análise.

No estágio da Descoberta de Ideias, geram-se e desenvolvem-se ideias. Nesta etapa, o julgamento deve ser posto à parte e devem ser produzidas ideias em quantidade considerando que quantidade gera qualidade. As ideias mais promissoras são depois selecionadas e desenvolvidas na atividade projetual (OSBORN, 2008).

Quadro 6 – Etapas do CPS.

A Definição do Problema	
1 -Construção de Oportunidades	Gerar possíveis oportunidades e desafios a considerar.
2 - Exploração de Dados	Examinar várias fontes de dados e diferentes pontos de vista. Identificar os pontos chave ou os dados mais importantes.
3 - Formulação de Problemas	Gerar muitos e variados caminhos para resolver o problema. Selecionar um problema específico.
Geração de Ideias	
4 - Geração de ideias	Produzir, muitas e variadas ideias. Identificar as ideias com potencial a desenvolver.
Ação	
5 - Desenvolvimento de Soluções	Encontrar caminhos a seguir e fortalecer possibilidades promissoras. Analisar, avaliar, priorizar e lapidar soluções promissoras.
6 - Implementação da Solução	Considerar várias fontes de assistência e resistência e possíveis ações a implementar. Formular planos específicos para que ganhem suporte, para construir e avaliar as ações.

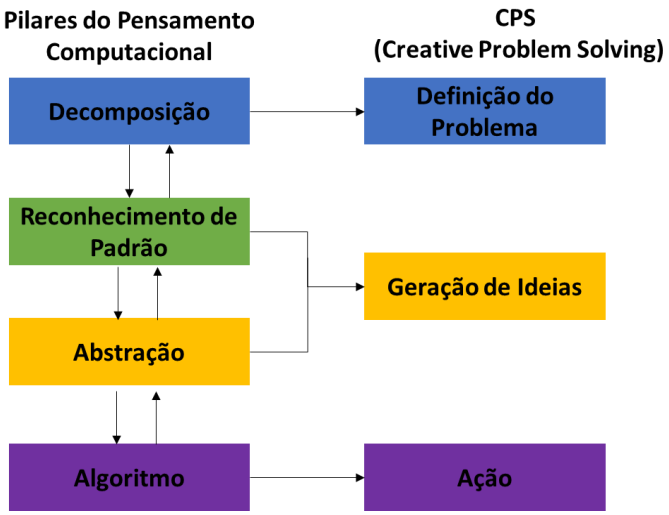
Fonte: Osborn (2008).

A fase da Descoberta de Soluções engloba a avaliação das ideias provisórias para a

escolha da solução final e a sua implementação. O processo de avaliação deve ser ponderado, permitindo o melhoramento das soluções prometedoras. A avaliação das ideias põe em evidência a inteligência crítica, analítica e o pensamento convergente (OSBORN, 2008).

Estas fases conduzem o processo do Pensamento Criativo. O método do CPS estrutura todo o processo por etapas sequenciais para que as pessoas compreendam o que fazer, passo a passo, para produzir uma ou mais soluções criativas e significativas. Neste processo são utilizados o pensamento divergente e técnicas de criatividade para a geração de muitas ideias e oportunidades que, conforme a fase, podem ser dados ou informações, definições do problema, ideias ou estratégias de implementação, deixando o pensamento fluir de forma livre. Nesta fase de divagação do pensamento, o julgamento é suspenso. O pensamento convergente também é utilizado quando se avalia e se efetuam as opções entre as várias ideias ou possibilidades originadas na fase divergente. Na fase convergente, são efetuadas escolhas dos dados e informações mais importantes, das ideias mais promissoras e das estratégias mais apropriadas (OSBORN, 2008).

Figura 11 – Detalhes do Pilares e CPS (*Creative Problem Solving*).



Fonte: Silva, Melo e Tedesco (2020).

A fase de Decomposição que compõe os pilares do Pensamento Computacional está relacionada à fase de definição do problema CPS (FIGURA 11), pois esta técnica ajuda a dividir o problema e observá-lo de diferentes perspectivas. O estágio de Reconhecimento de Padrões visa reconhecer os padrões para criar algo novo ou inovador e diz respeito à fase de geração do CPS, que é a etapa selecionada para essa função. A fase de Abstração está relacionada à fase de geração de ideias, uma vez que visa treinar no indivíduo a habilidade de observar conceitos apenas de forma genérica na busca pelos mais relevantes. Além disso, está situado no estágio

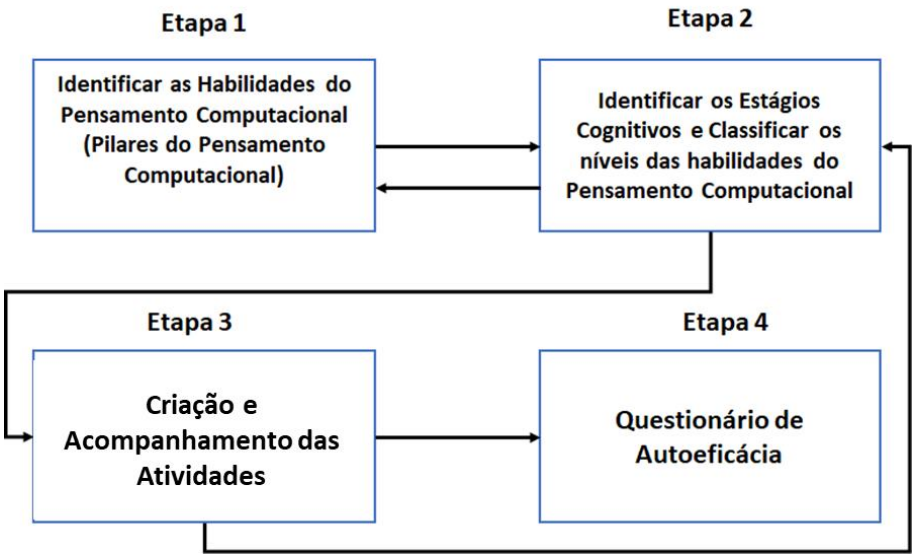
de geração de ideias, um momento de convergência.

Por fim, o pilar Algoritmo está relacionado ao código, UML ou qualquer representação algorítmica da solução e está localizado na fase Ação do modelo CPS, por ser a fase de desenvolvimento da solução.

4.2 GUIA PARA O ENSINO DE PROGRAMAÇÃO

O Guia para o Ensino de Programação (GEP) foi baseado no Modelo *Skills-CT* apresentado na seção 4.1. Ele tem o objetivo de auxiliar o professor a planejar as aulas e atividades através de diretrizes e sugestões de uso de ferramentas para o ensino de programação para uma turma iniciante. O GEP possui quatro etapas que são exibidas na Figura 12. Ao aplicar o Modelo *Skills-CT*, pode-se utilizar ferramentas de Programação Visual como o *Scratch*, por exemplo, e, em relação ao ensino de programação, utiliza-se uma linguagem com paradigma procedural.

Figura 12 – Etapas do Guia para Ensino de Programação.



Fonte: A autora (2021).

Na primeira etapa do GEP, o objetivo é conhecer as habilidades de Pensamento Computacional de cada aluno. Para tanto, foram realizados exercícios (APÊNDICE A) para os alunos resolverem. Para avaliar as habilidades do Pensamento Computacional, foi utilizado o Dr. *Scratch*, ferramenta que é uma aplicação *web* gratuita que visa analisar os projetos realizados no *Scratch* resultando em *feedback* sobre a qualidade dos programas (MORENO-LEÓN; ROBLES; ROMÁN-GONZÁLEZ, 2015).

No que diz respeito ao desenvolvimento do Pensamento Computacional, o Dr. *Scratch* avalia o código dos programas e atribui uma pontuação em diferentes aspectos dessa habilidade, como Abstração e Decomposição de Problemas, Lógica, Sincronização, Paralelismo, Controle de Fluxo, Interatividade do Usuário e Representação de Dados. Cada um desses aspectos recebe um score entre 0 e 3 pontos e uma pontuação geral do Pensamento Computacional é dada pela soma das pontuações parciais.

Para este trabalho foram consideradas apenas as habilidades apresentadas na Tabela 1 para o uso do *Scratch* e na Tabela 2 para uso de linguagem de programação procedural. Esta escolha ocorreu por estarem relacionadas aos pilares do Pensamento Computacional.

Tabela 1 – Habilidades do Pensamento Computacional para o *Scratch*.

Conceito de Pensamento Computacional	**Níveis de Habilidades**			
	(0) Nulo	**(1)Básico**	**(2)Em Desenvolvimento**	**(3) Proficiência**
Abstração e Decomposição de Problemas	-	Mais de um *script* e mais de um *sprite*	Definição de blocos	Uso de clones
Paralelismo	-	Dois *scripts* na bandeira verde	Dois *scripts* na tecla pressionada, dois *scripts* no *sprite* e ação de clique no mesmo *sprite*	Dois *scripts* quando recebo a mensagem, criar clone, dois *scripts* quando o cenário muda
Lógica	-	*if*	*if else*	Operações lógicas
Controle de Fluxo	-	Sequência de blocos	Repetir, Sempre	Repetir até
Representação de Dados	-	Modificadores de propriedades	Operações sobre variáveis	Operações em lista

Fonte: Moreno-León, Robles e Román-González (2015).

Tabela 2 – Habilidades do Pensamento Computacional para linguagem de programação procedural.

Conceito de Pensamento Computacional	**Níveis de Habilidades**			
	(0) Nulo	**(1)Básico**	**(2)Em Desenvolvimento**	**(3) Proficiência**
Abstração e	Não possui	Função sem	Função com	Uso de um

Decomposição de Problemas	função	retorno	retorno	recurso da linguagem de programação
Paralelismo	Não possui função	Uma função chamando outra	Usar retorno de uma função como parâmetro de outra função	Usa de um recurso da linguagem de programação
Lógica	Não possui *if*	*if*	*if else*	Operações lógicas
Controle de Fluxo	Não possui estrutura de repetição	Não possui estrutura de repetição	*For*	*While*
Representação de Dados	-	Não possui variáveis	Operações sobre variáveis	Operações em lista ou *array*

Fonte: A autora (2021).

Após a análise de uma atividade, na segunda etapa do GEP ocorre a identificação dos Estágios Cognitivos dos alunos – nas Figuras 8 e 9 pode ser vista a relação entre os Estágios Cognitivos dos alunos e as habilidades de Pensamento Computacional para o *Scratch* e para qualquer Linguagem de Programação que use o paradigma de programação procedural.

Na terceira etapa do GEP, ocorre o acompanhamento e criação de atividades práticas que visam compreender o conhecimento do discente. Além das atividades práticas, para esta etapa foi utilizado o *Kahoot*, que é uma aplicação *web* gamificada, na qual é possível aplicar *quizzes* virtuais animados e musicados com o objetivo de compreender o conhecimento dos estudantes sobre rastreamento de código, funcionalidades dos blocos do *Scratch* e conhecimento teórico sobre os fundamentos de programação. O *Kahoot* foi escolhido porque é gratuito e fornece relatórios sobre sucessos e erros individuais e da classe.

Ainda sobre as atividades criadas, elas devem apresentar situações em que os alunos precisem resolver um problema usando uma linguagem de programação pedida no enunciado. O objetivo é apresentar um problema cuja solução não seja direta, fazendo com que os alunos desenvolvam soluções mais criativas. Para isso, utiliza-se a relação entre o modelo dos Pilares do PC com o CPS (*Creative Problem Solving*) (FIGURA 10) implicitamente no problema para criar o cenário de aprendizagem, empregando-se a estrutura:

- Definição do problema: com o objetivo de o estudante identificar o problema e dividi-lo em partes para solução;

- Reconhecimento de Padrão e Abstração na Geração de ideias para que o discente possa pensar na solução usando esses elementos do PC;
- Algoritmo relacionado à ação (CPS), uma vez que criará uma solução usando programação para verificar se está correta.

Exemplo de enunciado (APÊNDICE A): “Cremildo foi fazer um saque no caixa eletrônico do Banco. Mas, para fazer o saque, precisa ter dinheiro no banco, o valor do saque deve ser menor ou igual ao que tem na conta, debitar (subtrair) o valor do saque do que já possui na conta. Faça no *scratch* um algoritmo que simule um caixa eletrônico”. Observando a estrutura do enunciado, pode-se classificar assim:

- Definição do problema: simular um caixa eletrônico;
- Reconhecimento de padrão e Abstração: saber que o problema pode ser resolvido usando condicional com operadores lógicos;
- Algoritmo: criar a solução usando *Scratch.*

Esta estrutura é usada para que o professor crie o exercício e pode ser usada na solução criada pelo estudante, ajudando o discente na aprendizagem criativa de problemas.

Nesta etapa, também foi utilizado o quadro Kanban (ANDERSON, 2010) que é um Gerenciador de Atividades Visual e foi usado pelo professor com o objetivo de ajudá-lo a gerenciar as atividades, pois, para o Modelo, as atividades e assuntos serão direcionados segundo os resultados das análises das habilidades dos alunos e de seus Estágios Cognitivos.

Tendo em vista a utilização do modelo pelo(a) professor(a), a Figura 13 apresenta um quadro que facilita a criação de atividades, bem como a preparação das aulas. O quadro Kanban foi dividido em seis cartões. A ordem dos cartões é: Tarefa, Problematizar/Conceituar, Praticar, Verificação, Avaliação e Conclusão. Para o cartão “Tarefa”, o professor vai adicionar a lista de atividades a serem realizadas (planejadas).

No cartão de “Problematiza/Conceituar”, deve-se especificar o assunto e a forma que vai ser abordado. Para esta etapa, pode-se usar diversas técnicas de criatividade de domínio do professor para criar um cenário de aprendizagem criativa, como, por exemplo, o D2D (*Dominate to Destroy*), que consiste em dominar os padrões antes de desconstrui-los. O termo dominar, nesse contexto, significa entender e mapear os padrões básicos de algum domínio (GUN, 2017). Outro exemplo é o *Zoom Out* – olhar criativo, que tem o intuito de fazer com que o sujeito observe um conceito genérico e o aplique em uma situação para resolver um problema de outro contexto (GUN, 2017).

No cartão “Praticar”, deve-se planejar as atividades práticas que os alunos deverão

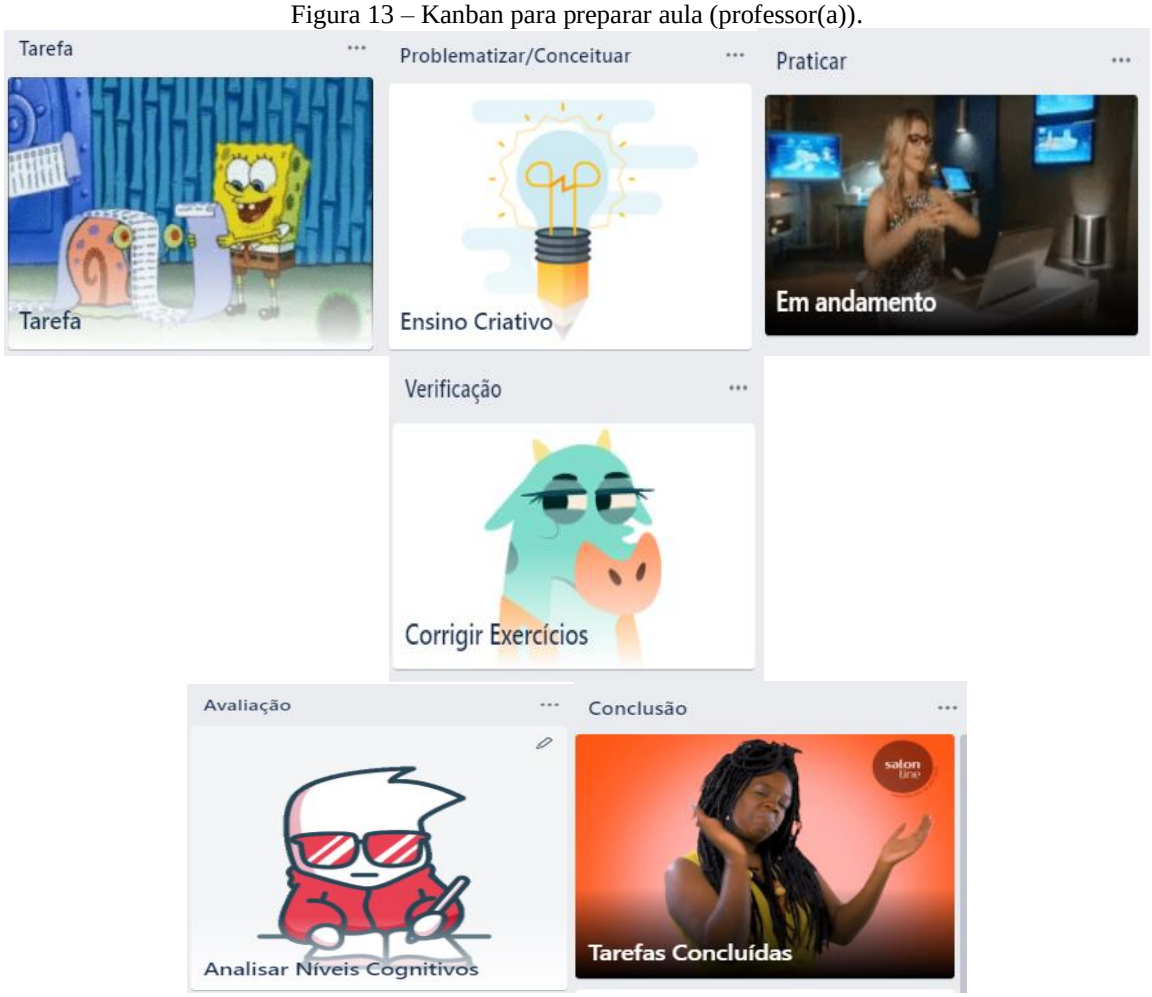
realizar usando os pilares e as habilidades do Pensamento Computacional.

No cartão “Verificação”, o professor deve corrigir os exercícios e dar *feedback* aos alunos. Para este cartão será usada a relação entre as habilidades do Pensamento Computacional usando o Dr. *Scratch* e/ou outro classificador para linguagem de programação procedural e os Estágios Cognitivos para avaliar os discentes.

Por fim, no cartão de “Conclusão” ficam as tarefas concluídas pelo professor.

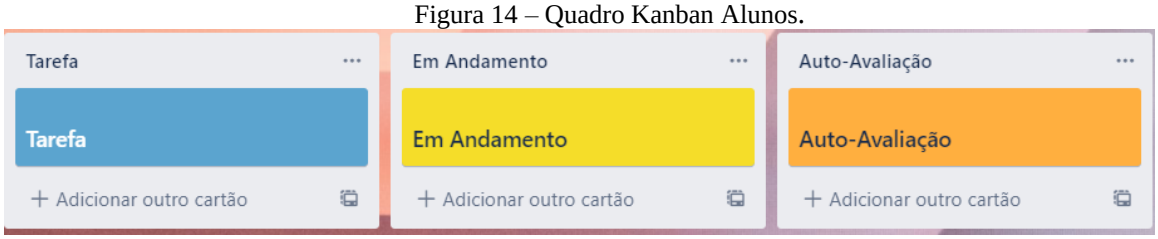
O quadro Kanban para os alunos é diferente e pode ser visto na Figura 14. No cartão “Tarefa”, o aluno vai listar as tarefas relacionadas às atividades passadas pelo professor. No cartão “Em andamento”, o discente vai adicionar as atividades que está trabalhando no momento. No cartão “Auto-avaliação”, o aluno vai se autoavaliar, adicionar dificuldades encontradas no desenvolvimento do projeto, desafio ou exercício e também escrever sobre os pontos de melhoria da atividade concluída.

Figura 13 – Kanban para preparar aula (professor(a)).



Fonte: A autora (2021).

Figura 14 – Quadro Kanban Alunos.



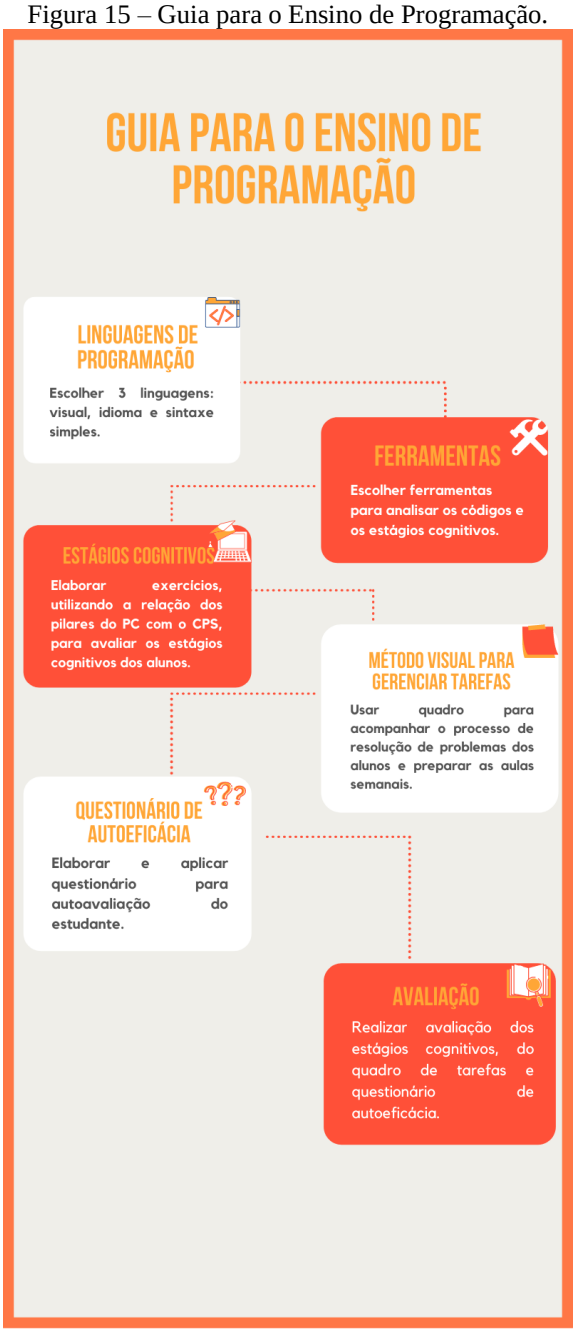
Fonte: A autora (2021).

Na quarta etapa do GEP foi realizado o questionário de Autoeficácia (APÊNDICE L) (ASKAR; DAVENPORT, 2009) para saber se os alunos achavam que eram capazes de realizar tarefas usando o *Scratch*. Esta etapa foi realizada no final de cada módulo ou disciplina.

4.2.1 Utilização do Guia para o Ensino de Programação

Para aplicar as etapas do GEP (FIGURA 15) no Guia de Ensino de Programação foi proposto não ir diretamente para uma linguagem de programação por *script* e o uso de uma IDE profissional para os iniciantes. Sendo assim, o GEP propõe o uso de três linguagens de programação: visual, intermediária (que use o idioma dos discentes) e linguagem procedural. Contudo, fica a critério do(a) professor(a) seguir os passos passando pelas três linguagens ou ir diretamente para uma linguagem de *script*.

Figura 15 – Guia para o Ensino de Programação.



Fonte: A autora (2021).

O GEP propõe ao professor, ao iniciar uma disciplina de programação para alunos iniciantes, que a primeira linguagem de programação seja a visual, como o *Scratch*, com o intuito de dar atenção ao ensino dos fundamentos de programação e as habilidades de PC já descritas na seção 4.1. Como as linguagens de programação visual são intuitivas, o aprendizado da sua utilização é rápido.

Para a linguagem intermediária, busca-se utilizar uma linguagem de *script*, cuja sintaxe seja no idioma do aluno – no caso, foi utilizada a Portugol, uma linguagem que ajuda na transição da programação visual para o *script*, pois também possui uma IDE simples e o

professor pode fazer a relação entre os blocos do *Scratch* e o Portugol, uma vez que as duas linguagens estão no idioma do discente. Por fim, o GEP propõe o uso de uma linguagem de programação procedural com uma sintaxe simples, como é o *Python*.

Um critério de escolha das linguagens de programação é que elas tenham as estruturas de programação apresentadas nas Figuras 16, 17, 18 e 19. Para este estudo, utilizou-se *Scratch*, Portugol (*Visualg*) e *Python*. No entanto, o professor pode utilizar as linguagens de programação que preferir, desde que tenha as principais estruturas que contemplem as habilidades do Pensamento Computacional.

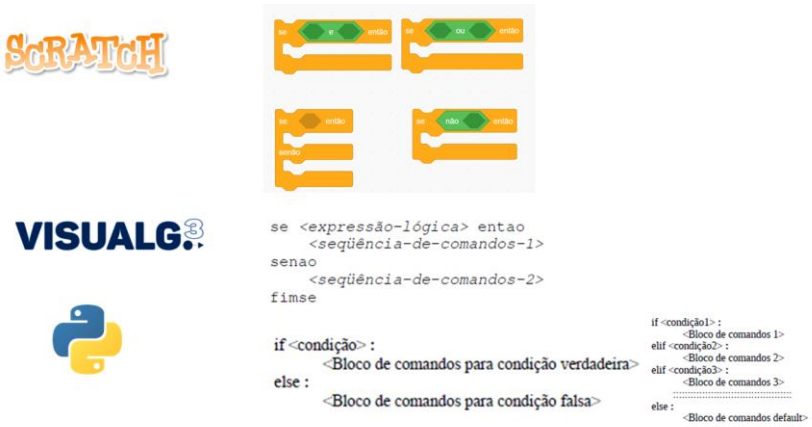
A escolha da ferramenta de programação visual deve se dar pela facilidade de uso e foco nas habilidades de PC vistas na Seção 4.1. Na escolha da linguagem de programação intermediária, deve-se escolher uma que seja no idioma dos alunos e de fácil utilização, sem muitas configurações, não precisando nesse momento ser um linguagem usada no mercado. Finalmente, a última linguagem pode ser procedural com uma sintaxe simples.

A utilização do *Python* ocorreu por esta ser uma linguagem de programação tipada dinamicamente, o que reduz a sobrecarga de notação e aumenta a facilidade para um programador iniciante se concentrar na resolução de problemas ao invés da sintaxe. Além disso, o *Python* oferece melhor expressividade, introduzindo tipos de dados como listas e dicionários que podem ser ensinados simultaneamente com outros tipos de dados básicos com facilidade (ATEEQ, 2014). Sem qualquer uso de colchetes e pontos-e-vírgulas extras, o *Python* reforça o *design* estrutural ao impor uma forma estruturada e recuada de escrever programas e seu ambiente de programação (IDE) é de código aberto e gratuito (ATEEQ, 2014).

Figura 16 – Variáveis – Representação de Dados.

Fonte: Scratch (2021); Visualg (2021); Python (2021).

Figura 17 – Estrutura Condicional – Lógica.



Fonte: Scratch (2021); Visualg (2021); Python (2021).

Figura 18 – Laços de Repetição – Controle de Fluxo.

Fonte: Scratch (2021); Visualg (2021); Python (2021).

Figura 19 – Funções – Abstração de Decomposição de Problemas e Paralelismo.

Fonte: Scratch (2021); Visualg (2021); Python (2021).

Sobre o quadro de ferramentas do Guia de Ensino de Programação, o professor pode escolher as que podem classificar automaticamente as habilidades do Pensamento Computacional do código dos estudantes. Com as informações das habilidades de PC, o professor deve realizar a classificação dos Estágios Cognitivos dos discentes.

No método de visual de gerenciamento de atividades apresentado no GEP, o professor pode escolher a ferramenta de sua preferência. Como já mencionado, o método usado é o Kanban.

A avaliação das atividades deve ser realizada constantemente e também a avaliação do quadro Kanban de atividade dos alunos. Ademais, o professor deve lançar mão do questionário de autoeficácia para entender o que o aluno julga saber sobre o aprendizado da linguagem de programação. O GEP não precisa ser usado em sequência, pois o Modelo apresenta etapas e o Guia de Ensino de Programação foi baseado no Modelo.

A Teoria da Autoeficácia foi um construto criado por Albert Bandura (2006) e é definida como a convicção que um indivíduo possui sobre sua capacidade de realizar com sucesso determinada atividade. Portanto, sua crença pode interferir em suas preferências e desempenho profissional. A autoeficácia faz a mediação entre o conhecimento de um indivíduo e suas ações. Adicionalmente, a autoeficácia reflete na maneira como os estudantes abordam os problemas (ASKAR; DAVENPORT, 2009), sendo a utilização da autoeficácia na aprendizagem um dos elementos que ajudam na resolução de problemas.

Ou seja, o indivíduo pode ter o conhecimento ou as habilidades necessárias para realizar uma tarefa, mas pode falhar devido a fatores externos. A autoeficácia, então, é uma medida da convicção de uma pessoa em sua capacidade de agir com sucesso em um contexto específico. Logo, tem uma influência poderosa sobre o que um indivíduo escolhe fazer e os cursos de ação que ele ou ela toma (BANDURA, 2006).

A autoeficácia é muito relevante no contexto educacional, pois essa teoria identifica que o desempenho de uma pessoa interfere em sua autoeficácia e seu desempenho futuro. Assim, alunos com alta autoeficácia são mais propensos a cumprir tarefas desafiadoras e estar mais engajados para concluí-las mesmo com dificuldades do que alunos com baixa autoeficácia (BANDURA, 2006).

De acordo com Bandura (2006), as pessoas baseiam suas crenças de autoeficácia em quatro fontes de informação: i) experiência pessoal; ii) experiência vicária, ou seja, ver outros iguais com a mesma habilidade; iii) persuasão verbal; e iv) somática e estados emocionais. O autor entende que a compreensão da eficácia se desenvolve gradativamente por meio da conquista de habilidades e experiências.

4.3 CONSIDERAÇÕES SOBRE O CAPÍTULO

Neste capítulo, apresentou-se o Modelo Conceitual *Skills-CT*, criado a partir das lacunas identificadas na literatura e discorridas nos Capítulos 1 e 3 deste trabalho. Dentre estas, buscou-se tratar a classificação dos Estágios Cognitivos dos estudantes para ajudar no desenvolvimento do Pensamento Computacional e como este se relaciona com a solução de problemas.

A partir da proposta do Modelo Conceitual *Skills-CT*, um estudo foi realizado para identificar se o *Skills-CT* trouxe resultados positivos no desenvolvimento do Pensamento Computacional e mudanças nos Estágios Cognitivos. O próximo capítulo apresenta o estudo realizado com o Modelo Conceitual *Skills-CT* e a utilização do GEP em uma turma de um curso profissionalizante de Desenvolvimento de *Software*.

5 JORNADA DE ENSINO E APRENDIZAGEM COM O *SKILLS-CT*

Fonte: *Tumblr* Tiras Armandinho (2016)[5].

Este capítulo apresenta o estudo realizado através da aplicação do Modelo Conceitual usando o Guia de Ensino de Programação, as características da turma do estudo e a Jornada de Aprendizagem de Programação (JAP) que foi criada e baseada na aprendizagem criativa. Adicionalmente, são apresentadas as ferramentas usadas na pesquisa e as especificidades do classificador de códigos em Portugol e *Python* desenvolvido pela pesquisadora.

5.1 PERFIL DOS PARTICIPANTES DO CURSO

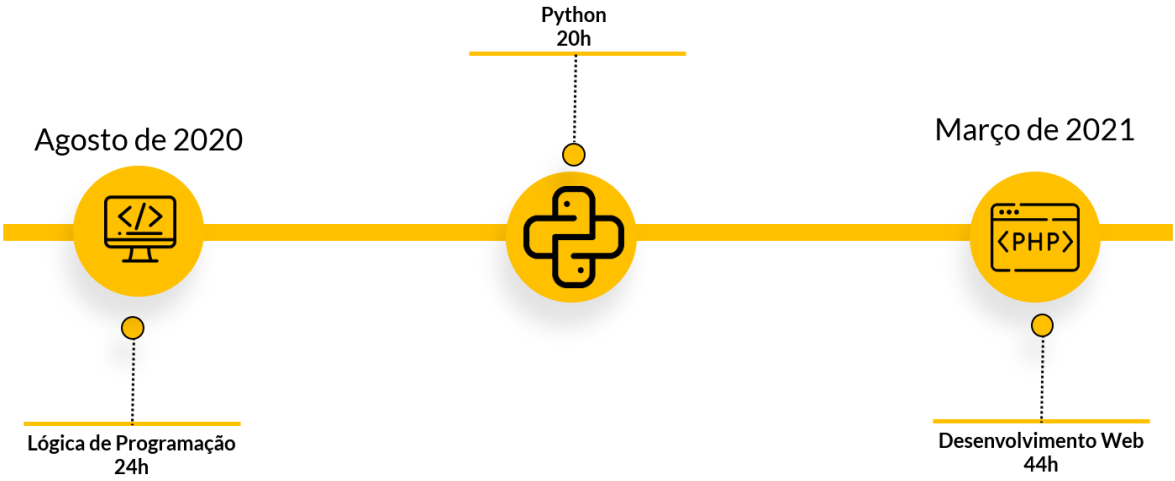
O estudo foi realizado com uma turma de onze alunos de um curso profissional *on-line* de Desenvolvimento de *Software*. Destes estudantes, sete eram do gênero masculino (A1, A2, A3, A5, A8, A10, A11) e quatro do gênero feminino (A4, A6, A7, A9). Apenas cinco alunos (A1, A2, A5, A6, A11) tinham conhecimento prévio de introdução à programação. A pesquisadora deste estudo foi a professora da turma.

O curso de Desenvolvimento de *Software* tem a duração de sete meses (FIGURA 20), iniciando em meados de agosto de 2020 e finalizado em março de 2021. O curso possuía três módulos que foram abordados na seguinte ordem: Lógica de Programação (carga horária de 24h e duração de seis semanas), *Python* (carga horária de 20h e duração de cinco semanas) e Desenvolvimento *Web* (carga horária de 44h e duração de dez semanas).

Os módulos utilizados neste estudo foram o de Lógica de Programação e o de Python. As aulas do curso ocorreram duas vezes por semana, com duração de duas horas cada. A ferramenta usada nas aulas *on-line* foram o *Meet*, o *Classroom* para envio de material e atividades e o *GitHub* onde os alunos submetiam seus códigos.

[5]Disponível em: https://tirasarmandinho.tumblr.com/post/141632600754/tirinha-original. Acesso em: 16 jan. 2021.

Figura 20 – Linha do Tempo – Curso de Desenvolvimento de *Software*.



Fonte: A autora (2021).

5.2 APLICANDO O MODELO

Para aplicação do modelo, o Guia de Ensino de Programação foi usado para planejar as aulas, as ferramentas utilizadas para classificar as habilidades do PC, as linguagens de programação visual e de *script* dos módulos de Lógica de Programação e *Python*.

A Jornada de Aprendizagem de Programação (JAP) (FIGURA 21) foi desenvolvida com objetivo de criar uma experiência de aprendizagem focada na resolução criativa de problemas, análise da autoavaliação realizada pelos alunos e, finalmente, o questionário de autoeficácia.

Figura 21 – Jornada do Estudante.

Fonte: A autora (2021).

Para a JAP, primeiro são criados exercícios que devem ser feitos pelos estudantes e, em seguida, o professor deve avaliar a solução no contexto da aprendizagem criativa, onde o código desenvolvido deve produzir algo novo e relevante, essencialmente entendido como um processo de apropriação e produção de conhecimento.

Após a análise do exercício, deve-se verificar o resultado da atividade de rastreamento de código e a autoavaliação usando o quadro Kanban seguido do questionário de autoeficácia que deve ser aplicado no final do módulo. As etapas 1 e 2 serão baseadas nos resultados dos exercícios de rastreamento de código e autoavaliação. Após passar uma vez pelas etapas 1 e 2, os próximos exercícios serão baseados nos resultados do rastreamento do código e autoavaliação e a fase 2 será baseada nos resultados dos exercícios. O objetivo é avaliar constantemente os estudantes em todo o processo do seu aprendizado através do código e das suas soluções.

Sendo assim, o GEP dá atenção ao planejamento das aulas, classificação das habilidades do PC e Estágios Cognitivos e a JAP foca nas soluções criativas dos alunos e na autoavaliação. Este processo fica transparente para o aluno, ficando à cargo do professor utilizá-lo ou não. Porém, toda a decisão tomada é baseada nos resultados dos códigos dos discentes através das atividades realizadas por eles. O GEP e a JAP foram criados de modo a permitir deixar o planejamento aberto a mudanças decorrente das necessidades e dificuldades dos estudantes, com o objetivo de fazer com que eles melhorem as habilidades do PC e alcancem o Estágio Cognitivo Operacional Concreto.

Sobre a fase de perguntas para rastreamento do código, utilizou-se o *Kahoot* para o *quiz*, o quadro Kanban para a autoavaliação e, finalizando, o uso da autoeficácia. Foram usadas duas autoavaliações: a primeira na etapa 2, onde em cada atividade realizada os alunos usavam o quadro Kanban para autoavaliação, e a segunda no questionário de autoeficácia no final do módulo.

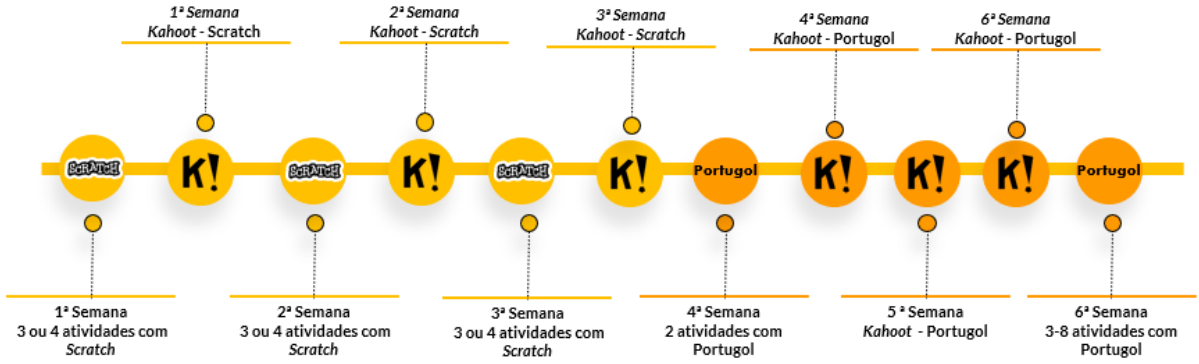
5.3 ESTUDO NO MÓDULO DE LÓGICA DE PROGRAMAÇÃO

No módulo de Lógica de Programação, durante as três primeiras semanas foi utilizada a ferramenta *Scratch* para a introdução de noções de programação. Nas três semanas subsequentes, passou-se a utilizar o *Visualg*, cuja Linguagem de Programação é Portugol (FIGURA 22).

O *Scratch* é uma ferramenta visual usada para programação em bloco, criada pelo *Massachusetts Institute of Technology* (MIT) para ensinar programação para crianças e PC de

forma lúdica. A ferramenta possui uma biblioteca com diversos blocos de programação que podem ser usados para desenvolver programas interativos, incluindo animações e jogos (RESNICK et al., 2009).

Figura 22 – Linha do Tempo – Módulo de Lógica de Programação.



Fonte: A autora (2021).

Nas três semanas iniciais do Módulo de Lógica, foram propostas dez atividades para que os alunos pudessem resolvê-las utilizando o *Scratch* (APÊNDICE A).

O *Kahoot* foi utilizado pela professora para criar três *quizzes* (APÊNDICES D, E e F), totalizando 35 questões, com o intuito de avaliar a compreensão das funcionalidades dos blocos do *Scratch*.

Os *quizzes* sobre o *Scratch* foram divididos em três momentos: o primeiro foi aplicado no início do módulo de Lógica, o segundo no meio e o terceiro no final das três semanas em que a professora usou o *Scratch*.

Ainda sobre o uso do *Scratch*, no módulo de Lógica de Programação, as dez atividades, onde os discentes tinham que resolver usando os blocos da programação visual, não foram realizadas ao mesmo tempo, mas sendo solicitadas a cada semana três ou quatro atividades.

A partir da terceira semana do módulo de Lógica de Programação, começou-se a usar o *Visualg* que é um *software* que possibilita criar, editar e executar os algoritmos em Portugol (pseudocódigo em português). Nesse momento do módulo, foram realizadas oito atividades (APÊNDICE B). As atividades 1 e 2 foram pedidas em sala e da 3 a 8 foram as atividades da avaliação do módulo. Foram feitas 23 questões divididas em três *quizzes* (APÊNDICES G, H e I) para avaliar a compreensão sobre Portugol e rastreamento de código feito pelo aluno.

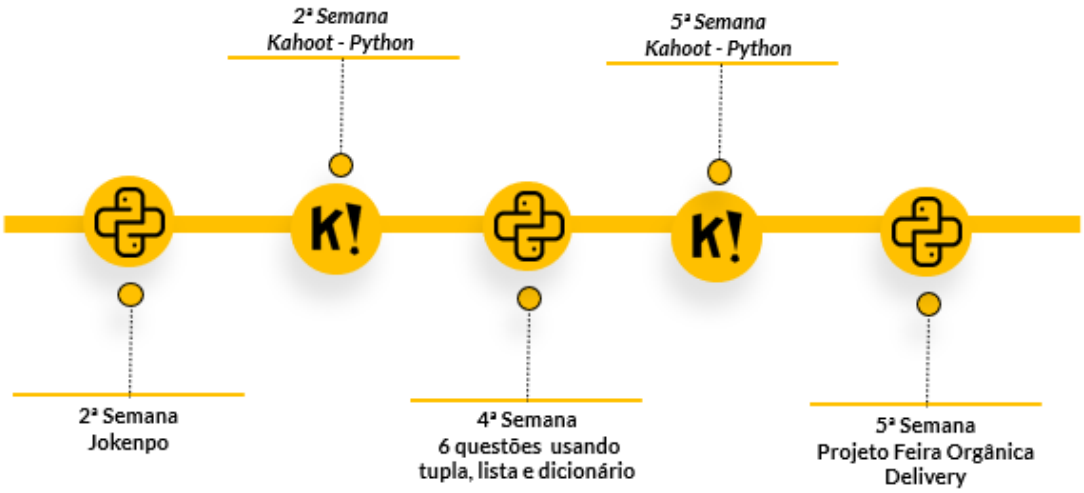
O *quiz* do Portugol foi dividido em três partes: o primeiro possuía onze questões e foi aplicado no início do módulo, o segundo no meio com cinco questões e o terceiro no final com sete questões. As 23 questões não foram realizadas ao mesmo tempo, mas a cada semana foram

aplicadas 1 ou 2 atividades.

5.4 ESTUDO NO MÓDULO DE *PYTHON*

Para o módulo de *Python*, a professora desenvolveu oito atividades (APÊNDICE C). O primeiro exercício foi um desafio para criar um jogo *Jokenpo* sem interface gráfica. A segunda atividade foi um projeto no qual os alunos tinham que fazer um programa em *Python* para pedidos de uma Feira Orgânica *Delivery* (FIGURA 23).

Figura 23 – Linha do Tempo – Módulo de *Python*.



Fonte: A autora (2021).

No módulo de *Python*, nas atividades 3 a 8, os discentes tiveram que criar seis questões que resolvessem um problema real, duas por assunto (tupla, lista e dicionários). Após criarem o enunciado das questões, os alunos também tiveram que resolver as questões usando a linguagem de programação do módulo. O *quiz* de *Python* foi dividido em dois momentos do módulo (APÊNDICES J e K): o primeiro com dez perguntas foi aplicado no início do módulo e o segundo, com dez perguntas, ao final.

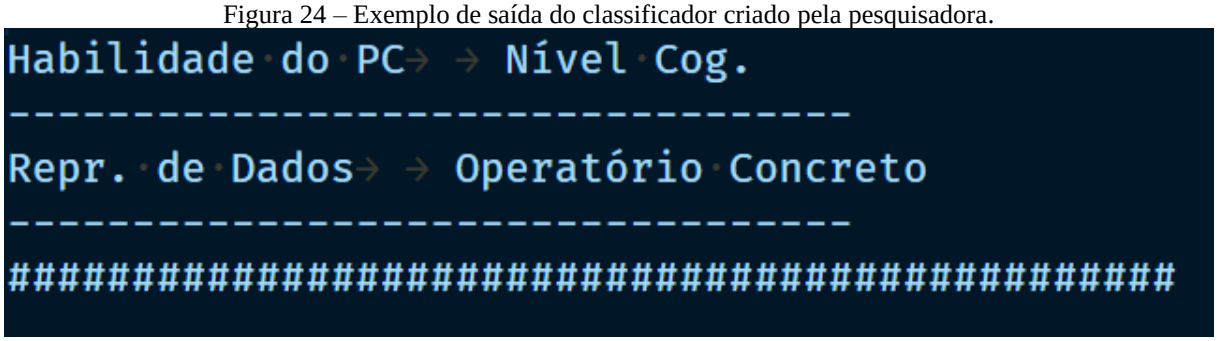
5.5 FERRAMENTAS USADAS NO ESTUDOS

Para o presente estudo, foram usadas algumas ferramentas, entre elas: Dr. *Scratch*, *Kahoot*, *Trello* (aplicativo de gerenciamento de projeto), *Visualg* e *VSCode*. Todas as ferramentas são gratuitas ou possuem versões gratuitas.

Para classificar as habilidades de PC foi usado o Dr. *Scratch* no módulo de Lógica de

Programação. Para a classificação das habilidades de PC e dos Níveis Cognitivos dos alunos através dos seus códigos em Portugol no *Visualg* e em *Python*, a autora criou um classificador que lê os programas dos alunos e, ao final, apresenta as habilidades de PC no código e os Estágios Cognitivos dos discentes (FIGURA 24).

Figura 24 – Exemplo de saída do classificador criado pela pesquisadora.



Fonte: A autora (2021).

O quadro Kanban foi usado no módulo de *Python* para ajudar os alunos a gerenciarem as atividades e fazerem os projetos, além de servir para a professora observar as dificuldades relatadas pelos estudantes.

Adicionalmente, um questionário de autoeficácia foi aplicado para os alunos se autoavaliarem (APÊNDICE L). No final de cada módulo, os discentes responderam ao questionário de autoeficácia. No módulo de Lógica de Programação, os alunos responderam ao questionário de autoeficácia do *Scratch*, *Visualg* (Portugol) e *Python* no fim do módulo.

No questionário de autoeficácia, utilizou-se a escala *Likert*, que é um tipo de resposta psicométrica frequentemente usada em questionários nas áreas de Psicologia, Educação e Marketing. Ao responder a um questionário com base nessa escala, os entrevistados detalham seu nível de concordância com uma afirmação (LIKERT, 1932). A partir das respostas dos alunos, o Valor Médio foi calculado com base na Média Aritmética Ponderada, conforme realizado por Oliveira (2005) por meio da seguinte fórmula:

$$\bar{x}_{assertiva} = \frac{\sum_{i=1}^{5} x_i . f_i}{n}$$

Onde:

$\bar{x}_{assertiva}$ = valor médio das avaliações da assertiva (afirmações);

x_i = valor da resposta assertiva;

fi = frequência da resposta;

n = número de alunos que responderam o questionário.

5.6 CONSIDERAÇÕES SOBRE O CAPÍTULO

Corroborando com os objetivos desta pesquisa, no presente capítulo foi apresentado o estudo com o intuito de aplicar o modelo *Skills-CT* em uma turma de um curso de Desenvolvimento de *Software*, nos módulos de Lógica de Programação e *Python*, para classificar os Estágios Cognitivos dos alunos através dos códigos criados por eles. A partir desta informação, foram propostas atividades para ajudar a desenvolver o Pensamento Computacional em consonância com o Estado Cognitivo no qual o discente se encontra.

Os resultados obtidos serão analisados e discutidos no próximo capítulo, bem como o impacto do Modelo *Skills-CT* no desenvolvimento do PC dos estudantes.

6 RESULTADOS E DISCUSSÃO

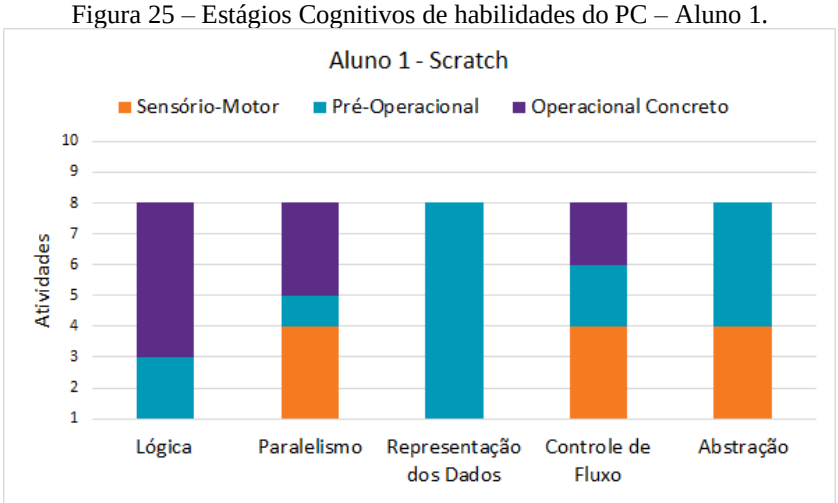
Fonte: *Facebook* Armandinho (2020)[6].

Este capítulo apresenta os resultados das análises dos códigos dos alunos, dos Estágios Cognitivos, dos Pilares do Pensamento Computacional e Questionários.

6.1 RESULTADOS DO *SCRATCH*

A análise foi realizada com base nos códigos dos alunos, nos Estágios Cognitivos e nos Pilares do PC. Após a análise dos códigos dos alunos feitos no Dr. *Scratch*, realizou-se a classificação de seus Estágios Cognitivos. Da Figura 25 até a Figura 35, os Estágios Cognitivos de cada aluno foram apresentados na análise dos dez exercícios (APÊNDICE A). A legenda do gráfico Abstração está relacionada à Abstração e Decomposição de Problemas.

Figura 25 – Estágios Cognitivos de habilidades do PC – Aluno 1.



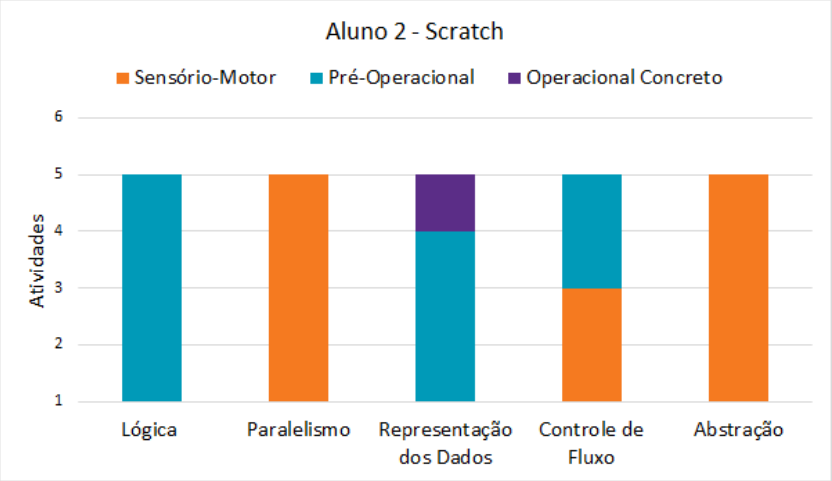
Fonte: A autora (2021).

[6]Disponível em: https://mobile.facebook.com/tirasarmandinho/posts/3820396704672274?_rdc=1&_rdr. Acesso em: 16 jan. 2021.

O Aluno 1 (FIGURA 25) tem um Estágio Cognitivo em desenvolvimento em quase todas as habilidades de Pensamento Computacional, apresentando mais dificuldade em habilidades que requerem o uso de *loop* ou o uso de variáveis em *loop*.

O Aluno 2 (FIGURA 26), apesar de ter experiência com outra disciplina de Introdução à Programação, apresentou o Estágio Cognitivo Sensório-Motor em Paralelismo e Abstração e Decomposição de Problemas. Com isto, foi possível verificar que ele tem dificuldade em dividir os programas em partes menores que possam ser executadas simultânea e recursivamente.

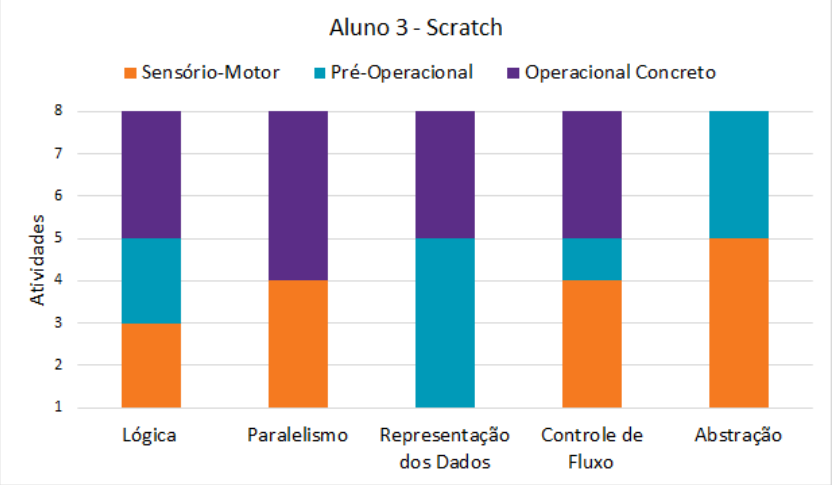
Figura 26 – Estágios Cognitivos de habilidades do PC – Aluno 2.



Fonte: A autora (2021).

A Figura 27 mostra o gráfico do Aluno 3 que, nas habilidades de PC Paralelismo, Controle de Fluxo e Abstração e Decomposição de Problemas, apresenta o Estágio Cognitivo Sensório-Motor, mostrando que o aluno se encontra no nível iniciante em programação.

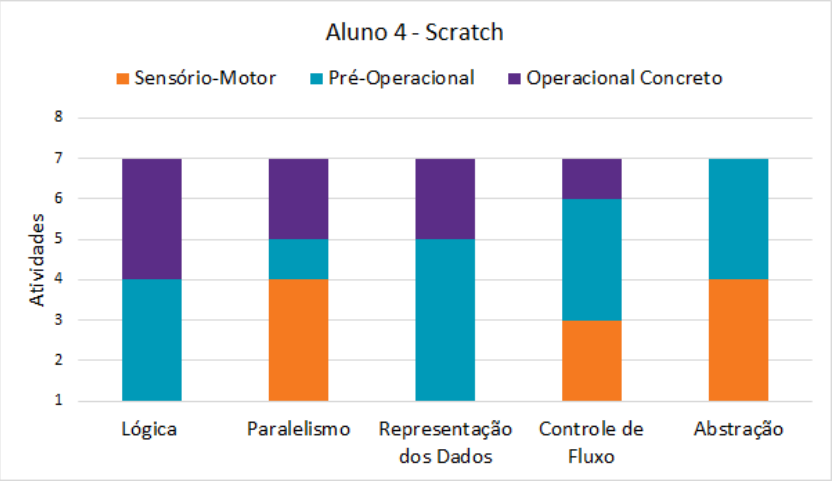
Figura 27 – Estágios Cognitivos de habilidades do PC – Aluno 3.



Fonte: A autora (2021).

O gráfico do Aluno 4 pode ser visto na Figura 28. Ele tem a maioria dos Estágios Cognitivos em desenvolvimento, pois está no Pré-Operacional. As habilidades de Pensamento Computacional, Paralelismo e Abstração e Decomposição do Problema estão com o Estágio Cognitivo Sensório-Motor, mostrando que o Aluno 4 tem dificuldade em aplicar a subdivisão de tarefas e abstração.

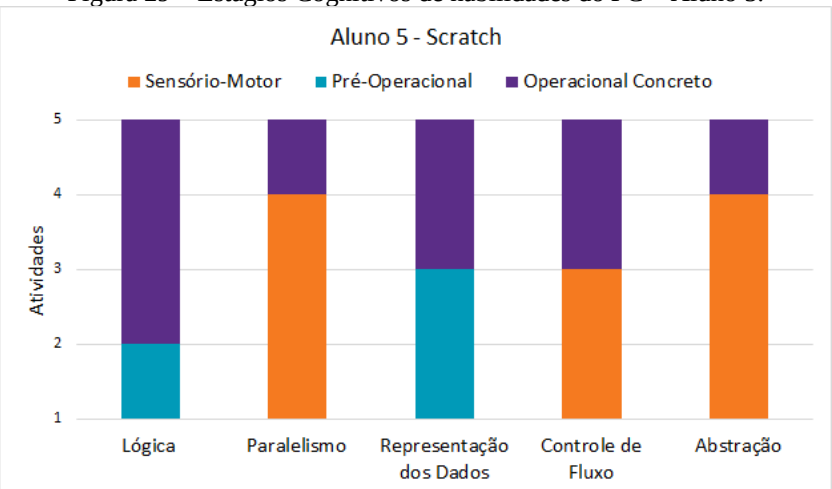
Figura 28 – Estágios Cognitivos de habilidades do PC – Aluno 4.



Fonte: A autora (2021).

O gráfico do Aluno 5 pode ser visualizado na Figura 29. Ele apresenta o nível Operacional Concreto mais presente na habilidade de Lógica e, pelo menos, na metade das resoluções em Representação de Dados e Controle de Fluxo. As habilidades de Paralelismo e Abstração e Decomposição de Problemas estão no início, pois ainda estão no Sensório-Motor.

Figura 29 – Estágios Cognitivos de habilidades do PC – Aluno 5.

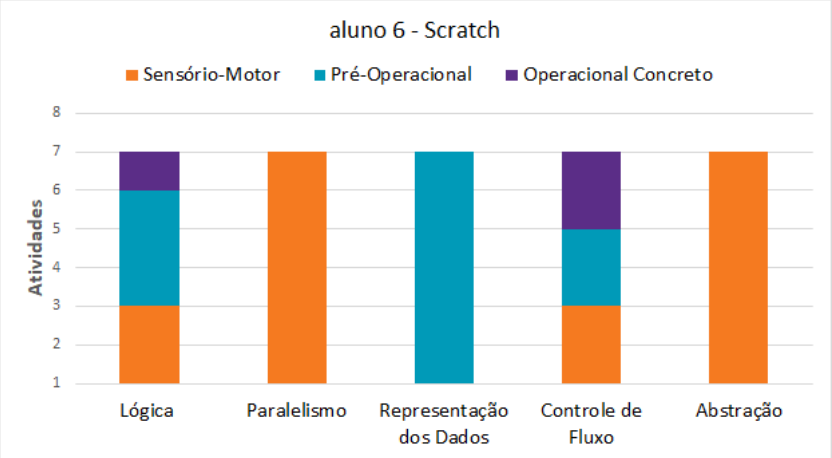


Fonte: A autora (2021).

O Aluno 6 (FIGURA 30) possui o Estágio Cognitivo Sensório-Motor em Paralelismo e

Abstração e Decomposição de Problemas, apresentando dificuldades nessas habilidades. As demais habilidades possuem um pouco dos Estágios Cognitivos, destacando-se Pré-Operacional em Lógica e Controle de Fluxo com os três níveis balanceados, mas o Sensório-Motor e Pré-Operacional possuem mais características de discentes iniciantes.

Figura 30 – Estágios Cognitivos de habilidades do PC – Aluno 6.

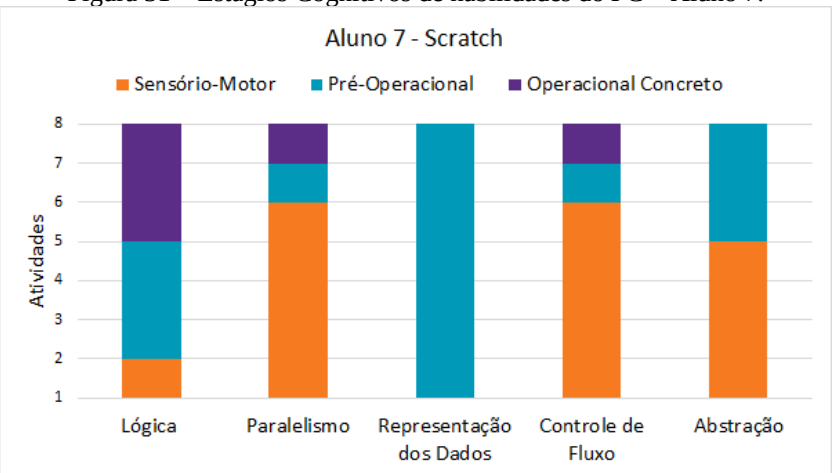


Fonte: A autora (2021).

O Aluno 7 (FIGURA 31) possui o Estágio Cognitivo Sensório-Motor com mais ênfase em Paralelismo, Abstração e Decomposição de Problemas e Fluxo de Dados, relatando dificuldades nessas habilidades.

A Representação dos Dados que se relaciona com a utilização de variáveis apresenta de forma completa o nível Pré-Operacional, indicando que o estudante utiliza e sabe atribuir valores às variáveis no seu código.

Figura 31 – Estágios Cognitivos de habilidades do PC – Aluno 7.

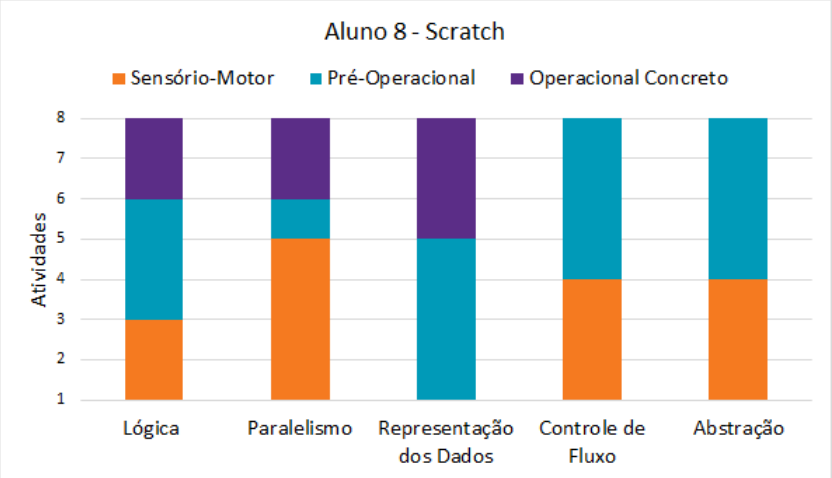


Fonte: A autora (2021).

O Aluno 8 (FIGURA 32) demonstra, em sua maioria, as habilidades de PC em

desenvolvimento e em Estágio Cognitivo Pré-Operacional, o que pode ser uma indicação de que o Aluno 8 é capaz de usar *loops*, variáveis e estrutura de seleção.

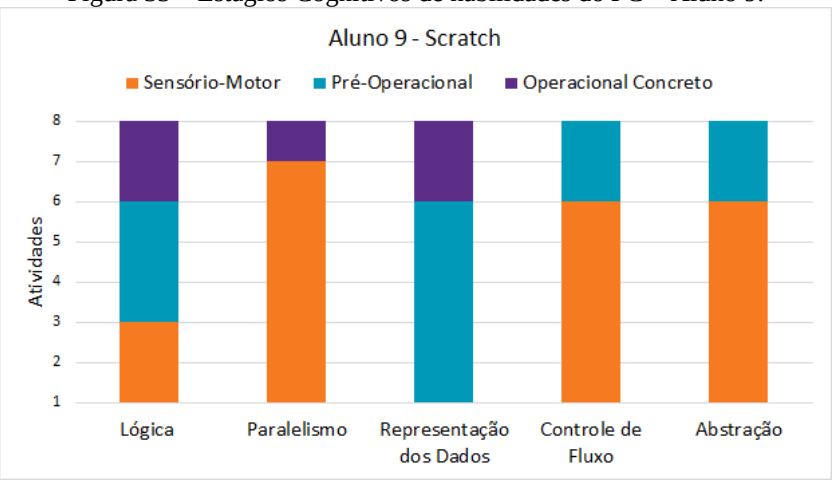
Figura 32 – Estágios Cognitivos de habilidades do PC – Aluno 8.



Fonte: A autora (2021).

O aluno 9 (FIGURA 33) possui o Estágio Cognitivo Sensório-Motor mais acentuado na maioria das habilidades de PC, apresentando dificuldades relacionadas à recursão, abstração e uso e atribuição de valores variáveis.

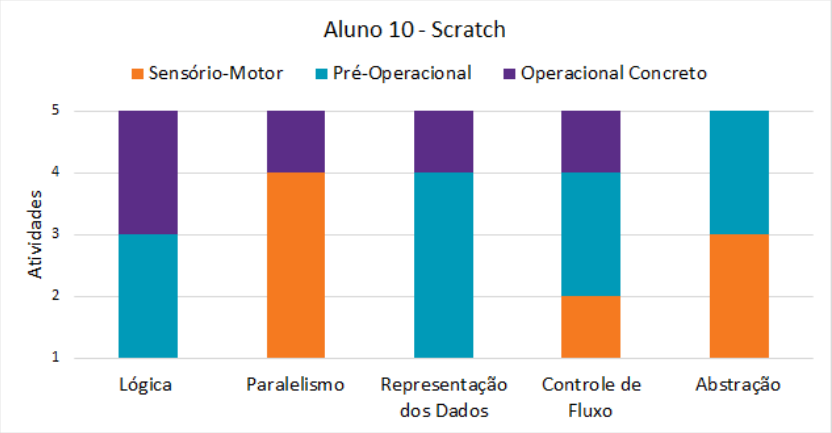
Figura 33 – Estágios Cognitivos de habilidades do PC – Aluno 9.



Fonte: A autora (2021).

O Aluno 10 (FIGURA 34) tem um Estágio Cognitivo Sensório-Motor mais acentuado em Paralelismo e Abstração e Decomposição de Problemas e em outras habilidades que estão em desenvolvimento. Na capacidade de Representação de Dados, o Aluno 10 pode usar e atribuir valores às variáveis.

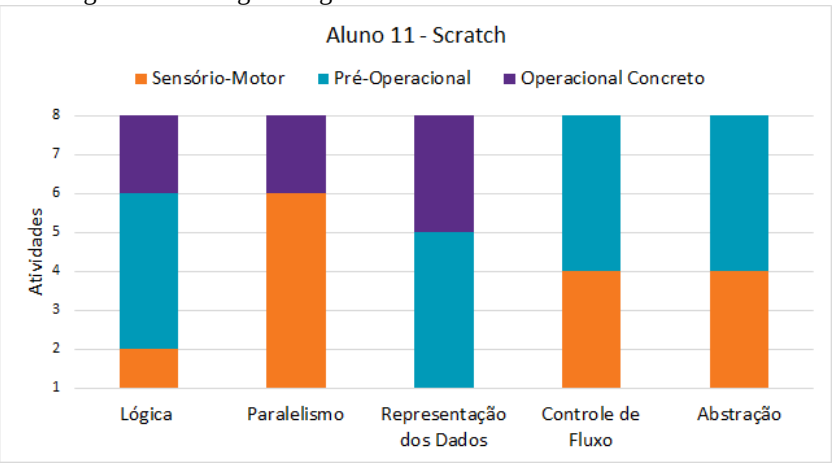
Figura 34 – Estágios Cognitivos de habilidades do PC – Aluno 10.



Fonte: A autora (2021).

O Aluno 11 (FIGURA 35) tem um Estágio Cognitivo Sensório-Motor mais forte no Paralelismo, mostrando que tem dificuldade em aplicar conhecimentos para criar um código que possa ser executado ao mesmo tempo. As habilidades de PC Abstração e Decomposição de Problemas e Controle de Fluxo estão em desenvolvimento, trazendo evidências de que ele pode usar *loop* e variáveis.

Figura 35 – Estágios Cognitivos de habilidades do PC – Aluno 11.



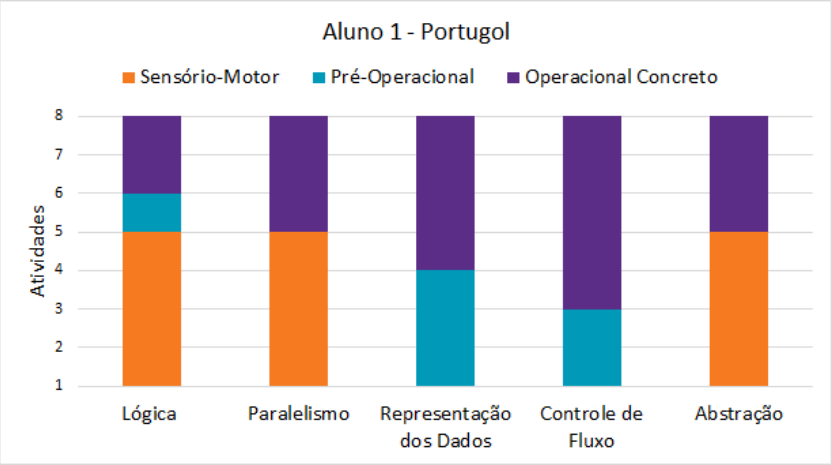
Fonte: A autora (2021).

6.2 RESULTADOS DO PORTUGOL

A análise foi realizada com base nos códigos dos alunos e nos Estágios Cognitivos em cada habilidade do PC. Após a análise dos códigos dos alunos pelo classificador de texto, foram criados gráficos nos quais é possível observar os resultados, demonstrando-os da Figura 36 até a 46. A legenda do gráfico Abstração está relacionada à Abstração e Decomposição de Problemas.

O Aluno 1 (FIGURA 36) tem o Estágio Cognitivo Sensório-Motor presente em três habilidades do PC: Lógica, Paralelismo e Abstração e Decomposição de Problemas. Com isto, pode-se inferir que o estudante tem dificuldade nos três primeiros Pilares do PC, que estão mais relacionados em preparar a resolução de problemas, e em Lógica, relacionada à utilização de operadores lógicos no algoritmo.

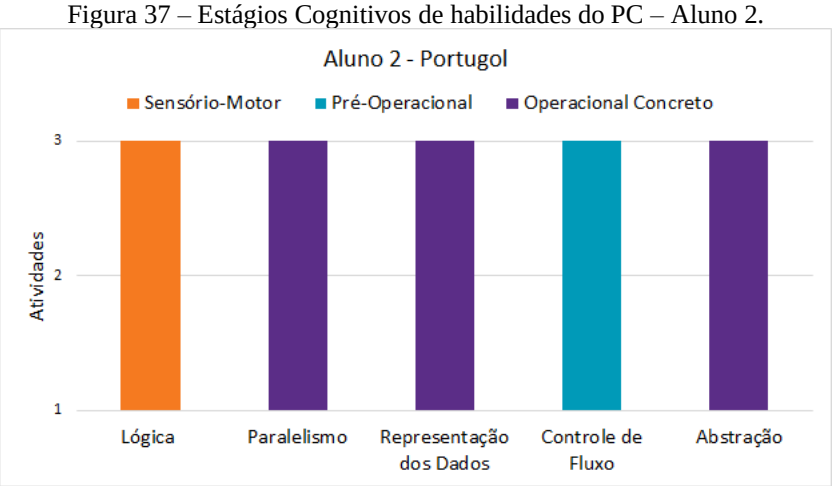
Figura 36 – Estágios Cognitivos de habilidades do PC – Aluno 1.



Fonte: A autora (2021).

O Aluno 2 (FIGURA 37) encontra-se no Estágio Cognitivo Sensório-Motor para a habilidade de Lógica, apresentando que não utilizou o "senão" do condicional ou os operadores lógicos em suas soluções. As outras habilidades mostram que ele tem proficiência em Paralelismo, Representação de Dados e Abstração e está em desenvolvimento em Controle de Fluxo.

Figura 37 – Estágios Cognitivos de habilidades do PC – Aluno 2.

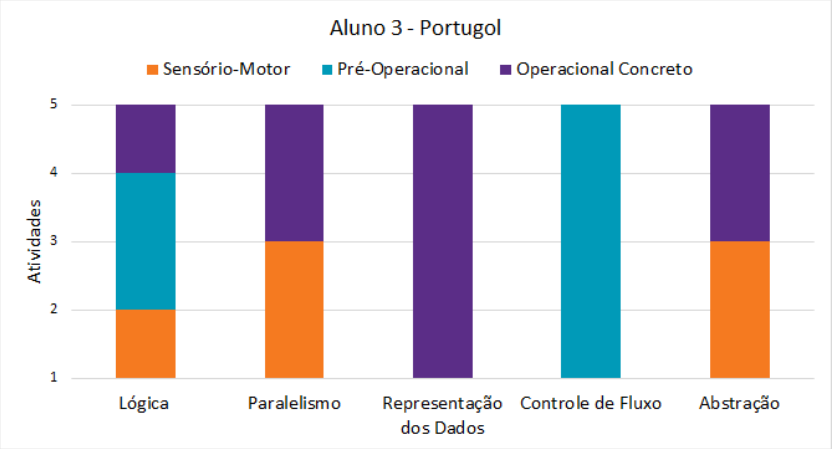


Fonte: A autora (2021).

O Aluno 3 (FIGURA 38) apresenta o Estágio Cognitivo Sensório-Motor na metade das

atividades e nas habilidades de Paralelismo e Abstração. Em Lógica e Controle de Fluxo, ele está em desenvolvimento, apresentando o Estágio Cognitivo Pré-Operacional.

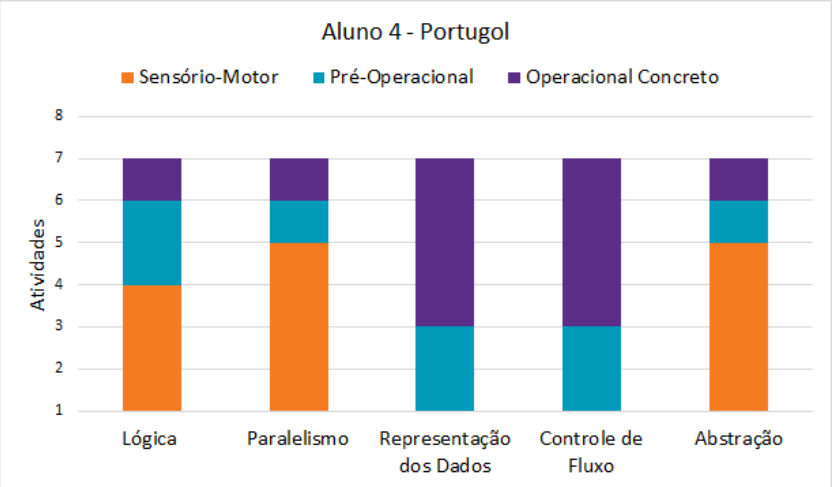
Figura 38 – Estágios Cognitivos de habilidades do PC – Aluno 3.



Fonte: A autora (2021).

O Aluno 4 (FIGURA 39) possui o Estágio Cognitivo Sensório-Motor mais presente nas habilidades de Lógica, Paralelismo e Abstração. Identificou-se que este aluno apresenta dificuldade tanto na utilização de operadores lógicos, como no Pilar de Abstração do PC.

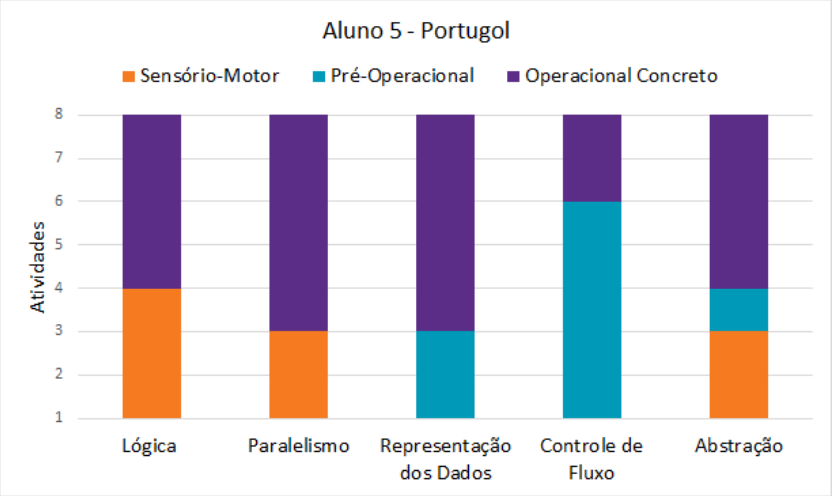
Figura 39 – Estágios Cognitivos de habilidades do PC – Aluno 4.



Fonte: A autora (2021).

O Aluno 5 (FIGURA 40) tem o Estágio Cognitivo Operacional Concreto em destaque para quase todas as habilidades de PC demonstrando que possui proficiência, porém, em Controle de Fluxo, o discente está no Pré-Operacional com nível de habilidade em desenvolvimento.

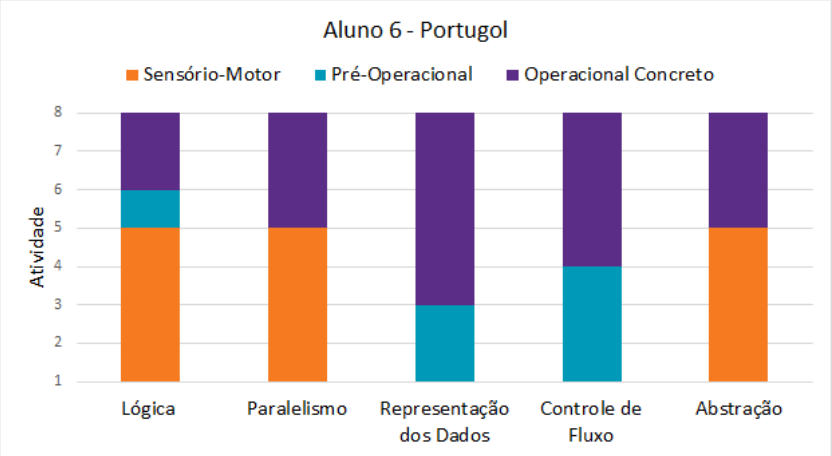
Figura 40 – Estágios Cognitivos de habilidades do PC – Aluno 5.



Fonte: A autora (2021).

O Aluno 6 (FIGURA 41) apresenta o nível Sensório-Motor mais acentuado em Lógica, Paralelismo e Abstração, indicando que o discente é iniciante nessas habilidades. Nas demais habilidades do PC, ele encontra-se no Estágio Cognitivo Operacional Concreto, tendo demonstrado proficiência.

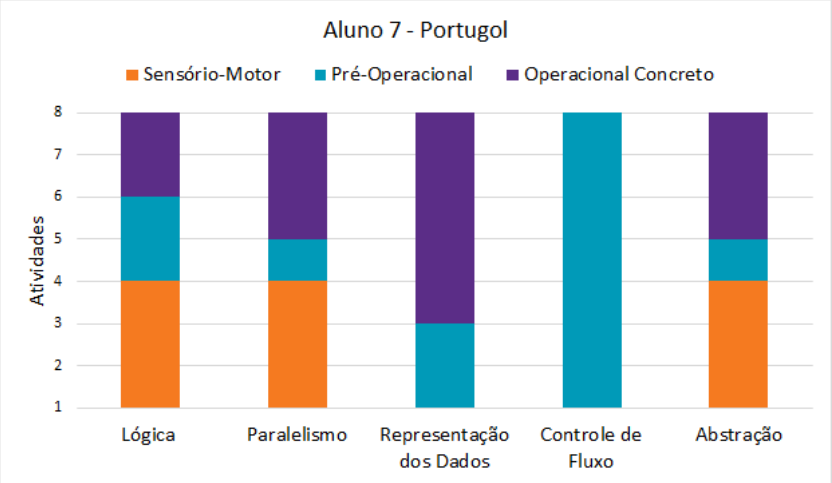
Figura 41 – Estágios Cognitivos de habilidades do PC – Aluno 6.



Fonte: Elaborada pela autora (2021).

O Aluno 7 (FIGURA 42) possui o Estágio Cognitivo Pré-Operacional em Controle de Fluxo em sua totalidade, mostrando que ele está em desenvolvimento e que utiliza *loop* em suas soluções. O Estágio Cognitivo Operacional Concreto encontra-se mais acentuado em Representação de Dados, relacionado em operações com vetores.

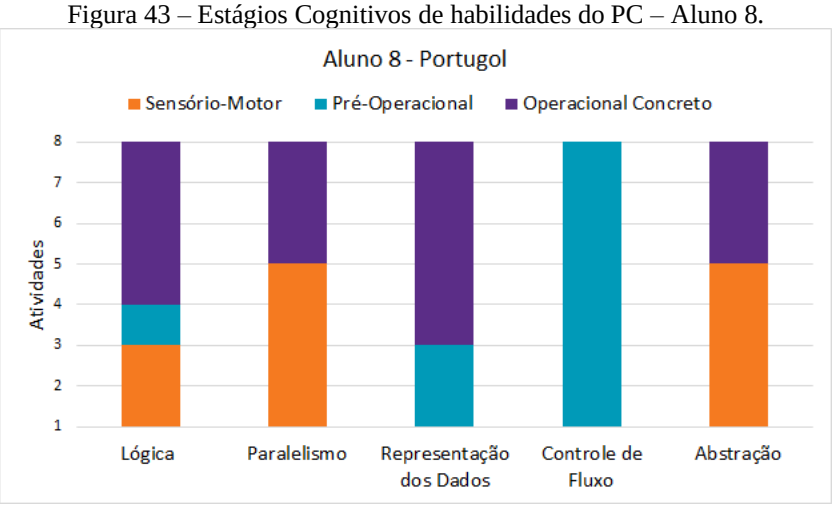
Figura 42 – Estágios Cognitivos de habilidades do PC – Aluno 7.



Fonte: A autora (2021).

O Aluno 8 (FIGURA 43) apresenta o Estágio Cognitivo Sensório-Motor nas habilidades de PC Abstração e Paralelismo, significando que o discente tem dificuldade de criar soluções que isolem uma representação do problema e que leve em consideração um aspecto importante para propor a resolução.

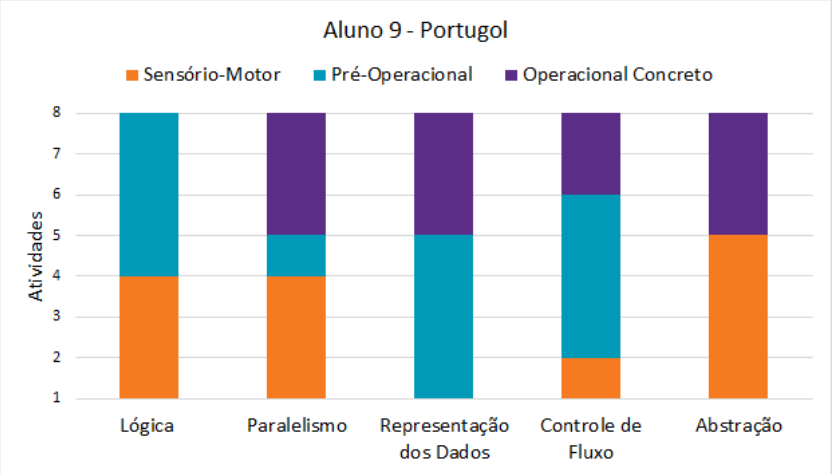
Figura 43 – Estágios Cognitivos de habilidades do PC – Aluno 8.



Fonte: A autora (2021).

O Aluno 9 (FIGURA 44) tem, em sua maioria, o Estágio Cognitivo Sensório-Motor na habilidade de PC de Abstração, enquanto nas outras habilidades os Estágios Cognitivos se equilibram entre Pré-Operacional, demonstrando que estão em desenvolvimento, e Operacional Concreto no nível proficiência.

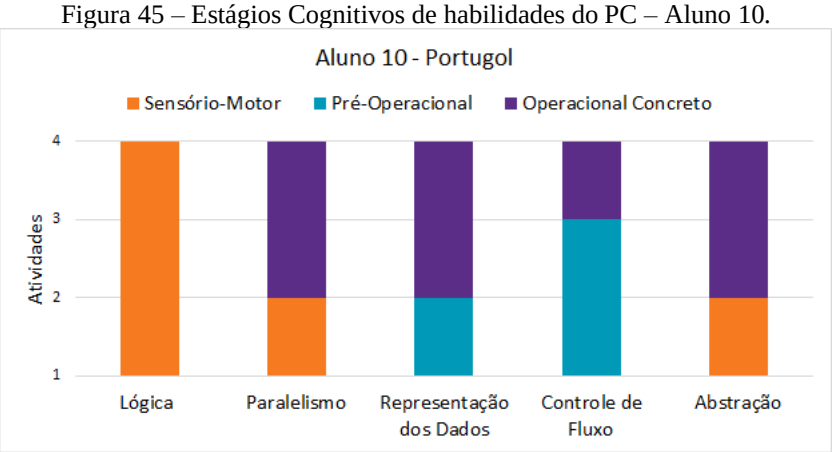
Figura 44 – Estágios Cognitivos de habilidades do PC – Aluno 9.



Fonte: A autora (2021).

O Aluno 10 (FIGURA 45) apresenta, na habilidade de Lógica, o Estágio Cognitivo Sensório-Motor, uma vez que não usou os operadores lógicos em suas soluções. Nas outras habilidades apresentou os Estágios Cognitivos Pré-Operacional e Operacional Concreto, expressando que o discente está em desenvolvimento ou no nível proficiência de suas habilidades de Pensamento Computacional.

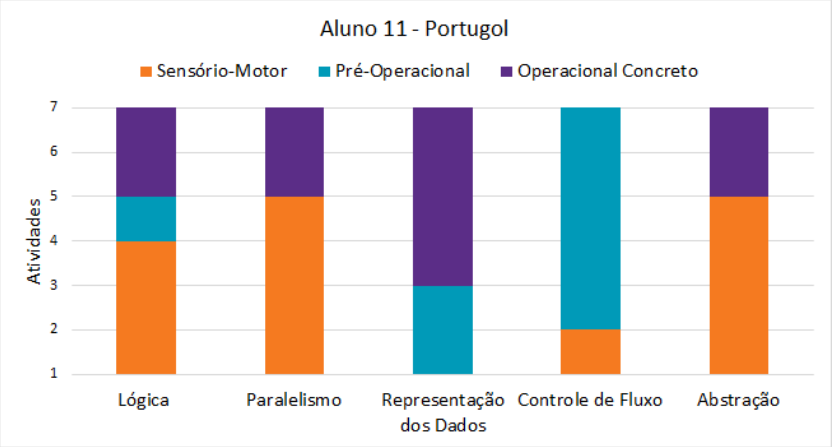
Figura 45 – Estágios Cognitivos de habilidades do PC – Aluno 10.



Fonte: A autora (2021).

O Aluno 11 (FIGURA 46) possui os Estágios Cognitivos Sensório-Motor mais acentuados em Lógica, Paralelismo e Abstração, indicando que se encontra no nível iniciante. Nas habilidades de Controle de Fluxo, está em desenvolvimento, uma vez que foi classificado no Estágio Cognitivo Pré-Operacional.

Figura 46 – Estágios Cognitivos de habilidades do PC – Aluno 11.



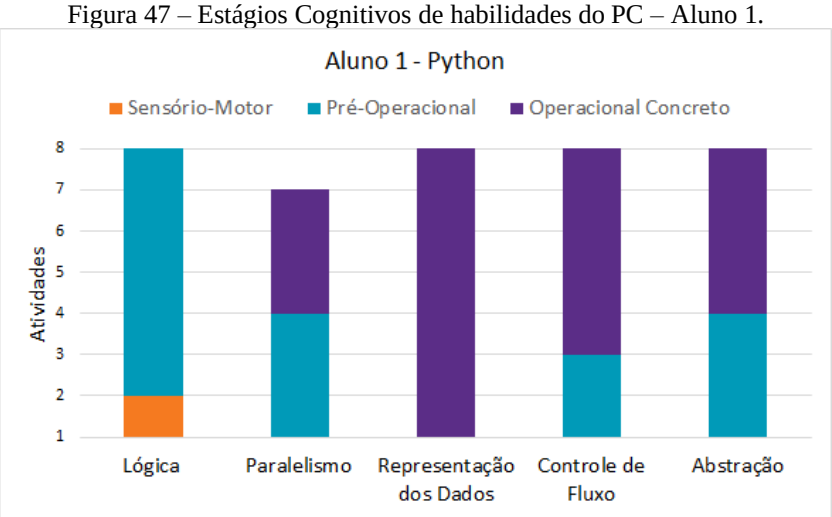
Fonte: A autora (2021).

6.3 RESULTADOS DO *PYTHON*

Aqui a análise também foi realizada com base nos códigos dos alunos e nos Estágios Cognitivos em cada habilidade do PC. Após a análise dos códigos dos alunos pelo classificador de texto, foram criados gráficos com os resultados da Figura 47 até a 57. A legenda do gráfico Abstração está relacionada à Abstração e Decomposição de Problemas.

O Aluno 1 (FIGURA 47) tem um Estágio Cognitivo Operacional Concreto presente em quatro habilidades de Pensamento Computacional, com maior ênfase na Representação de Dados, que possui em sua totalidade o nível Operacional Concreto.

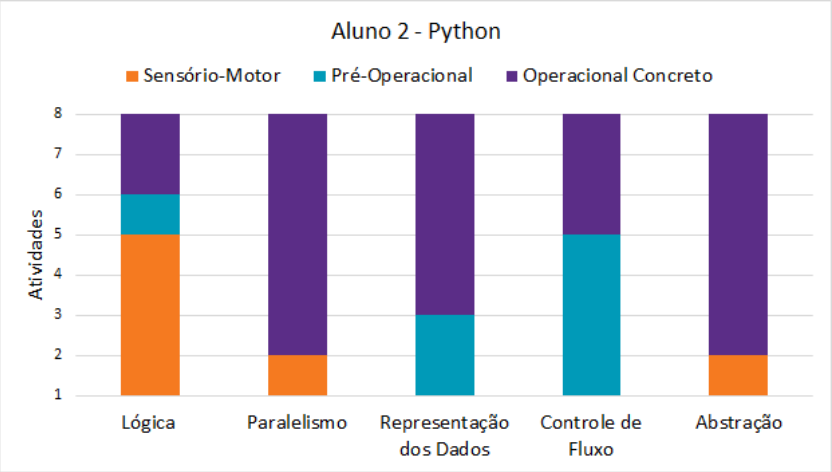
Figura 47 – Estágios Cognitivos de habilidades do PC – Aluno 1.



Fonte: A autora (2021).

O Aluno 2 (FIGURA 48) apresenta um Estágio Cognitivo Operacional Concreto com maior presença em Paralelismo, Abstração e Decomposição de Problemas e Representação de Dados.

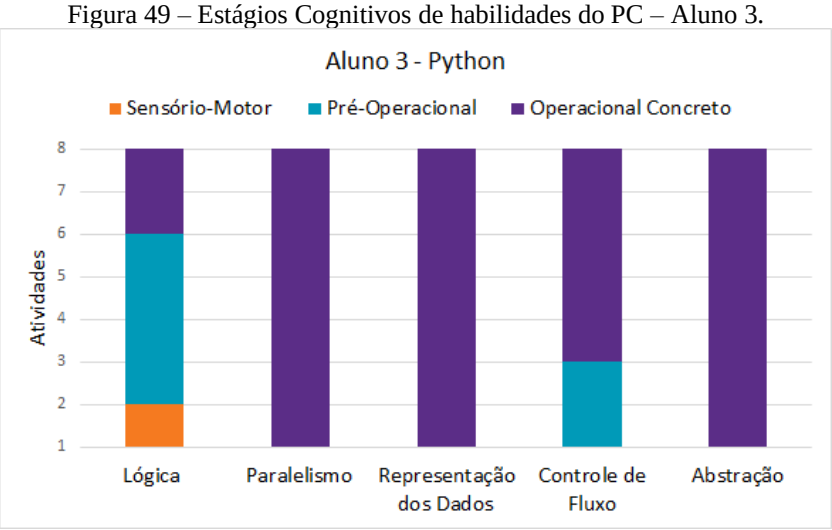
Figura 48 – Estágios Cognitivos de habilidades do PC – Aluno 2.



Fonte: A autora (2021).

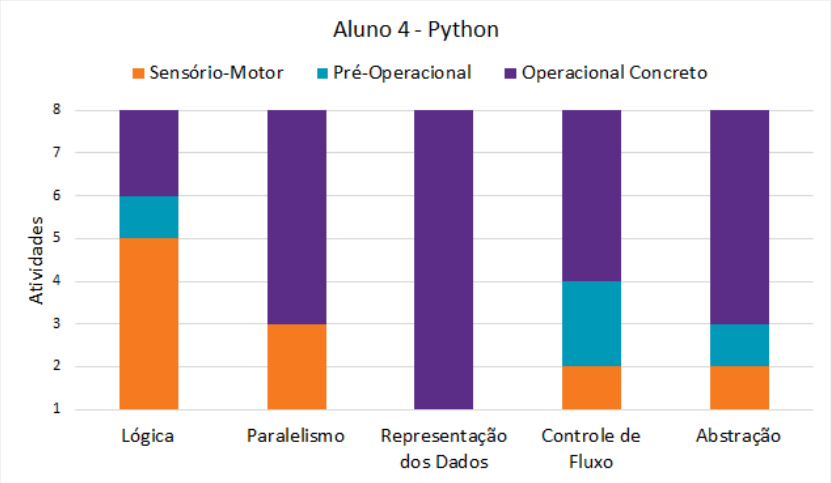
Os Alunos 3 (FIGURA 49) e 4 (FIGURA 50) têm o Estágio Cognitivo Operacional Concreto em Paralelismo, Representação de Dados e Abstração e Decomposição de Problemas, indicando que conseguem usar funções, dividir os problemas em partes menores e trabalhar com listas e/ou *arrays*.

Figura 49 – Estágios Cognitivos de habilidades do PC – Aluno 3.



Fonte: A autora (2021).

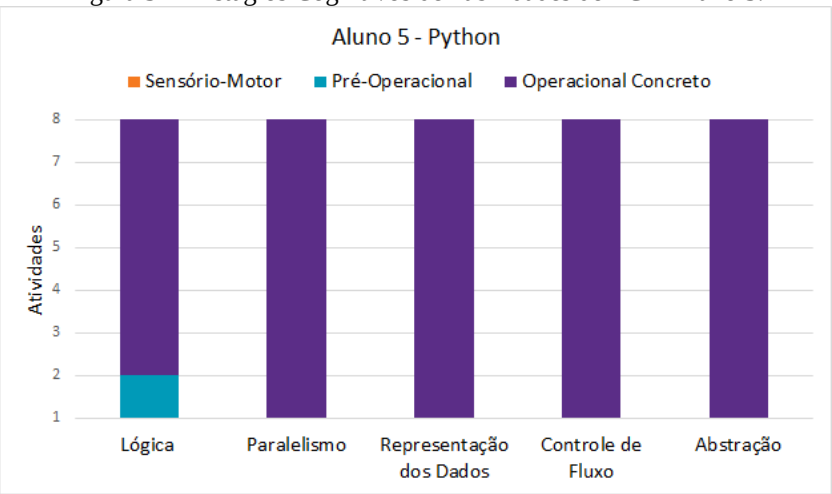
Figura 50 – Estágios Cognitivos de habilidades do PC – Aluno 4.



Fonte: A autora (2021).

O Aluno 5 (FIGURA 51) possui o Estágio Cognitivo Operacional Concreto em todas as habilidades do PC, apontando o uso delas na resolução de problemas algoritmicamente.

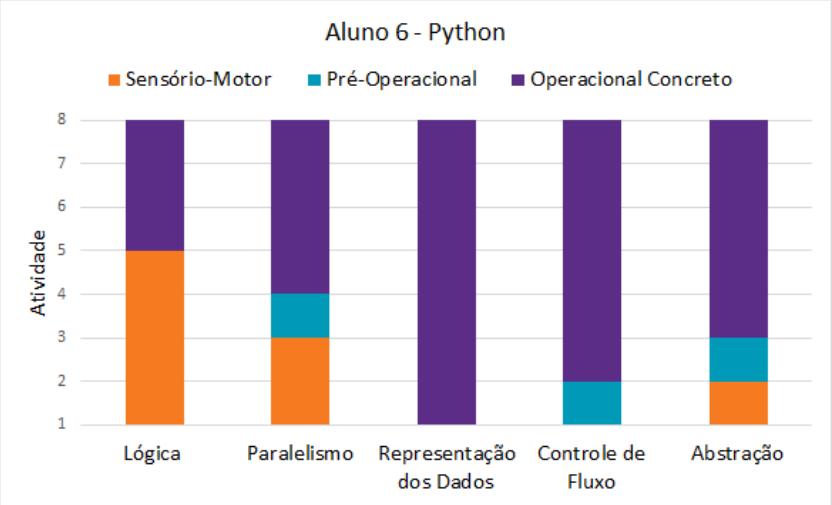
Figura 51 – Estágios Cognitivos de habilidades do PC – Aluno 5.



Fonte: A autora (2021).

O Aluno 6 (FIGURA 52) apresenta o Estágio Cognitivo Operacional Concreto mais acentuado em Controle de Fluxo, Abstração e Decomposição de Problemas, Representação dos Dados e Paralelismo, revelando que, ao longo das resoluções, seu nível de habilidades em PC chegou à proficiência.

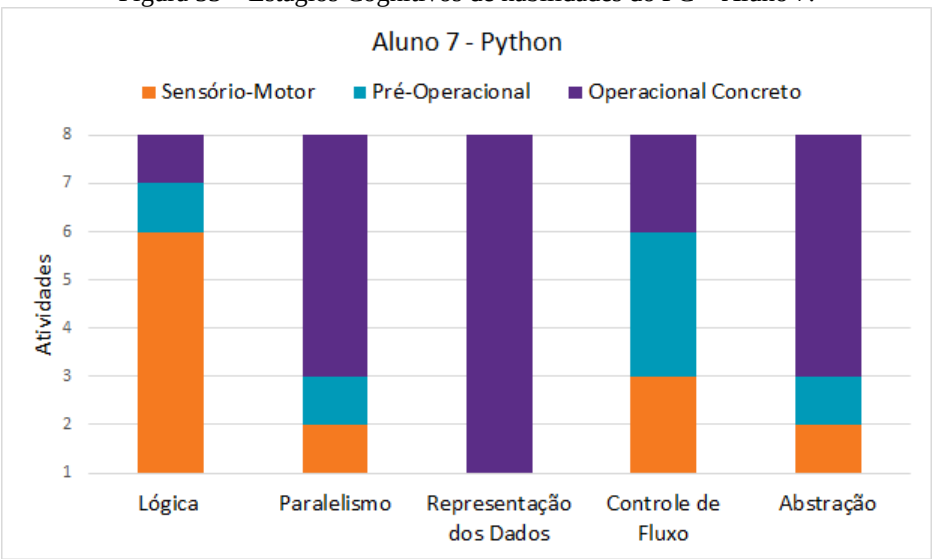
Figura 52 – Estágios Cognitivos de habilidades do PC – Aluno 6.



Fonte: A autora (2021).

O Aluno 7 (FIGURA 53) possui, em sua maioria, os Estágios Cognitivos Sensório-Motor e Pré-Operacional, sinalizando que suas habilidades do PC estão no nível básico ou em desenvolvimento.

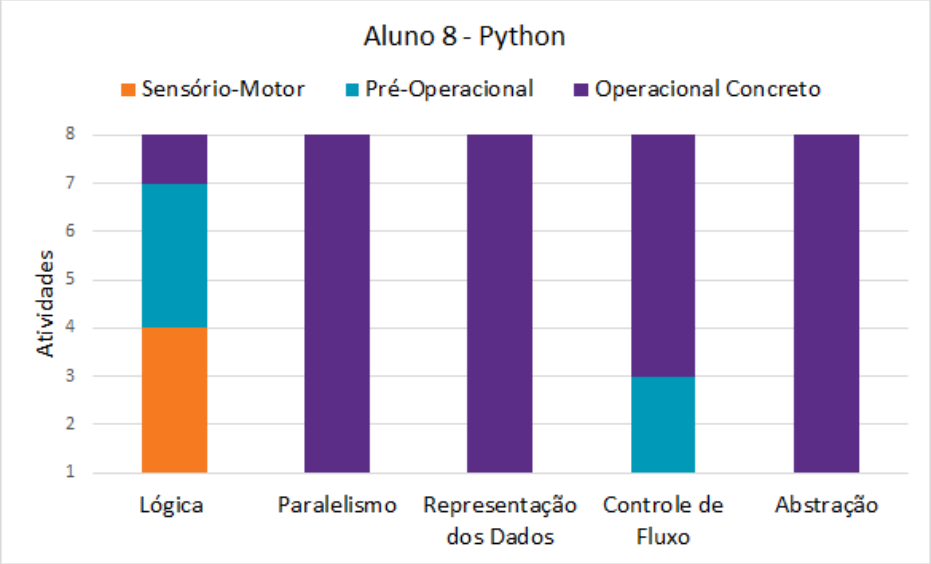
Figura 53 – Estágios Cognitivos de habilidades do PC – Aluno 7.



Fonte: A autora (2021).

O Aluno 8 (FIGURA 54) tem o Estágio Cognitivo Operacional Concreto na maioria das habilidades de PC, indicando proficiência.

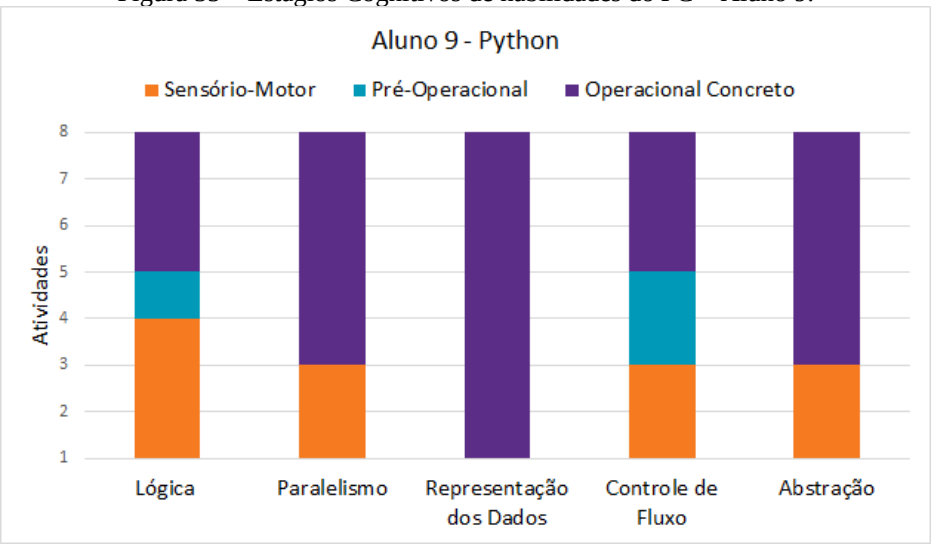
Figura 54 – Estágios Cognitivos de habilidades do PC – Aluno 8.



Fonte: A autora (2021).

O Aluno 9 (FIGURA 55) apresenta no Controle de Fluxo e Lógica nível das habilidades do PC em desenvolvimento.

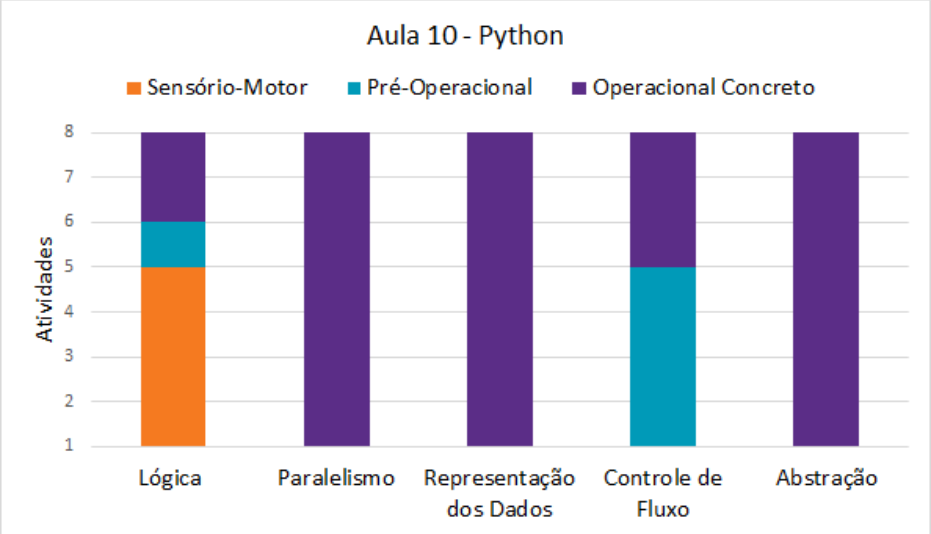
Figura 55 – Estágios Cognitivos de habilidades do PC – Aluno 9.



Fonte: A autora (2021).

O Aluno 10 (FIGURA 56) demonstra o nível Sensório-Motor em Lógica, o que significa nível básico e não utilização de operadores lógicos em suas soluções.

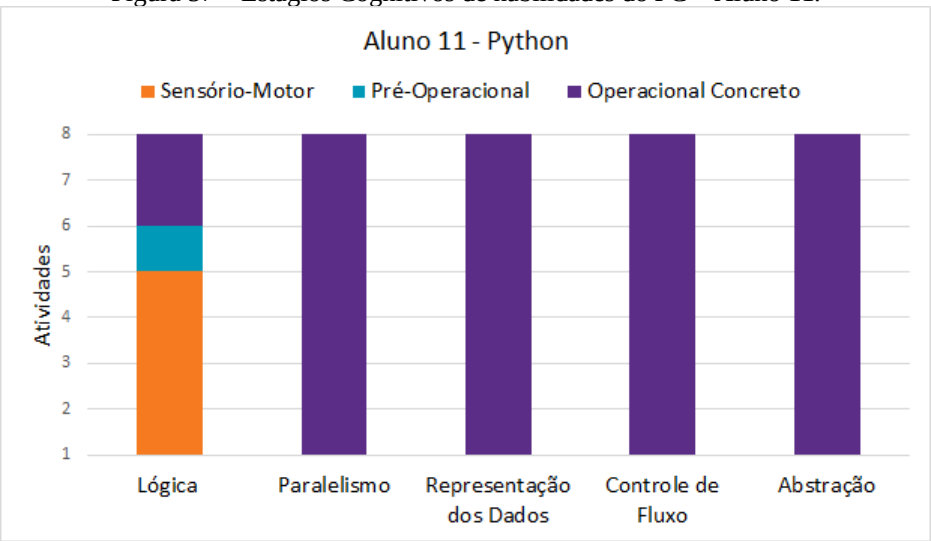
Figura 56 – Estágios Cognitivos de habilidades do PC – Aluno 10.



Fonte: A autora (2021).

O Aluno 11 (FIGURA 57) apresenta nível de habilidades do PC em Proficiência, mas em Lógica ainda está no Sensório-Motor, revelando que não usa operadores lógicos em suas soluções.

Figura 57 – Estágios Cognitivos de habilidades do PC – Aluno 11.



Fonte: A autora (2021).

Sobre o quadro Kanban utilizado no módulo de *Python*, os estudantes foram questionados se ajudou na resolução dos problemas (exercícios e projetos) e algumas respostas foram obtidas:

1) O Trello me ajudou a lembrar onde estavam acontecendo os erros do programa, a interface é bem responsiva, mas confesso que me complicou pois estava aprendendo outras coisas novas no curso ao mesmo tempo, como o *GitHub*,

instalações, Extensões do *Visual Studio Code*, como subir um código, como sincronizar o *Visual Studio Code* com o *GitHub*, o *Git* também e os erros que estive tendo nas instalações do programa por erros na minha máquina.

2) Sim, ajuda a formular uma linha de raciocínio e a mantê-la na hora de programar.

3) Sim, me ajudou muito para acompanhar meu progresso e poder ter o *feedback* de como melhorar.

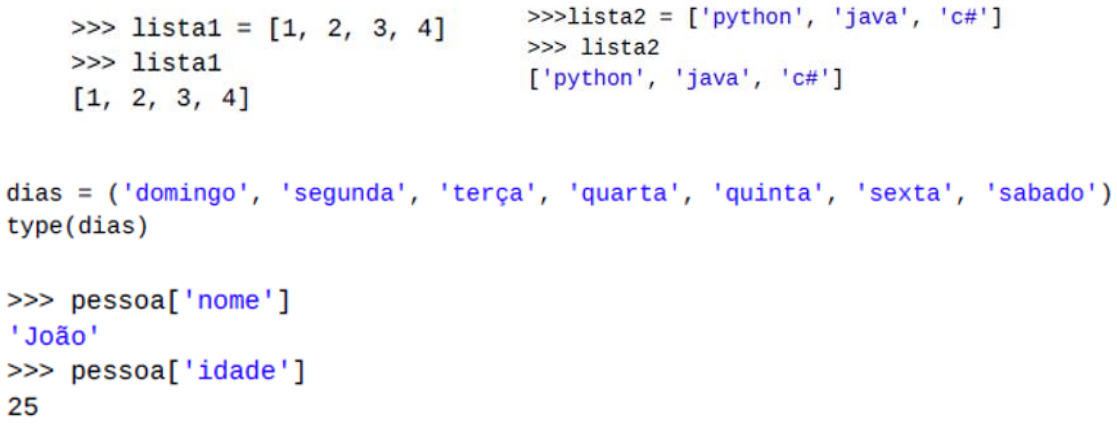
4) Sim! Porque me permitiu ter uma visão geral do processo de desenvolvimento e acompanhar os passos, porém não muito em algoritmos pequenos.

5) Não, porque quando estava desenvolvendo não pensava muito no Trello, e sim como resolver o problema na programação. Eu utilizei o trello como uma forma de documentar o programa.

A utilização do quadro Kanban no planejamento das aulas auxiliou na criação de exercícios e atividades baseados na descrição das dificuldades dos estudantes relatadas por eles.

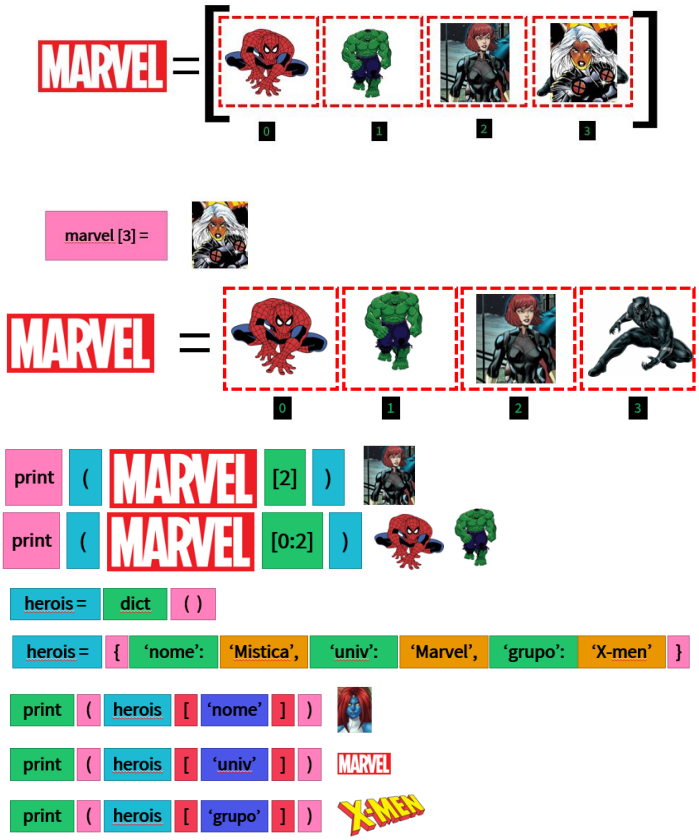
Um exemplo (FIGURA 58) dessa mudança ocorreu na aula de lista, tuplas e dicionários em *Python*. A apresentação da aula foi realizada de forma tradicional com exemplos e exercícios. Como as atividades eram acompanhadas pelo quadro Kanban, os estudantes informaram que não entenderam o conceito e a diferença de cada um desses elementos. Sendo assim, a aula foi reformulada trazendo um assunto de conhecimento dos estudantes, como os personagens da *Marvel* e da *DC* por meio de exemplos visuais e depois aplicando-os em Python (FIGURA 59).

Figura 58 – *Slide* da Aula de Lista, Tupla e Dicionário.



Fonte: A autora (2021).

Figura 59 – *Slide* da Aula Reformulada de Lista, Tupla e Dicionário.



Fonte: A autora (2021).

Depois do conhecimento dessa dificuldade com lista, tuplas e dicionários de *Python*, foi feita a atividade apresentada no Apêndice C, em que os estudantes tiveram que criar as próprias questões e soluções de problemas usando essas estruturas.

No final do módulo de *Python*, a professora fez o seguinte questionamento aos estudantes: "Você acha que usou de criatividade para resolver os problemas (Exercícios e projetos)? Se sim ou se não, justifique". A seguir, são apresentadas algumas respostas:

1) Sim. Eu utilizo métodos de *brainstorm* que a professora recomendou.
2) Sim, amo o modo como a senhora elabora as atividades, deixa tudo mais engajador.
3) Sim. Tive que pensar fora da caixa para resolver alguns problemas lógicos.
4) Sim, tentei adicionar recursos a mais para tornar ainda mais amigável o ambiente para o usuário, com situações criativas.
5) Sim, precisei em alguns exercícios e nos últimos, principalmente, para elaborar questões também.

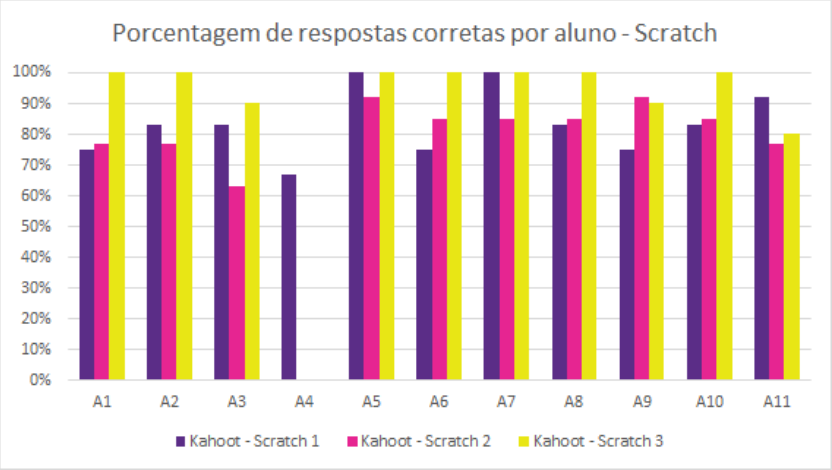
As respostas dos estudantes dão um indício de que aulas que trazem elementos de criatividade, usam assuntos da vivência dos estudantes e propõem atividades que estimulam a

criação de seus próprios problemas podem ajudá-los a usar a criatividade para resolver problemas e estimular a aprendizagem criativa.

6.4 RESULTADO DAS ATIVIDADES COM *KAHOOT*

Em relação ao *quiz* sobre questões teóricas, definição e funcionalidades dos blocos do *Scratch*, das Figuras 60 a 66 tem-se a porcentagem de respostas corretas para as questões. Os três *quizzes* realizados no *Kahoot* relacionados ao Scratch totalizaram 35 questões. A porcentagem de respostas corretas nos *quizzes* 1, 2 e 3 é de 72%, 78% e 87% (FIGURA 60), respectivamente, e das Figuras 67 a 69 há a apresentação das respostas corretas para cada questionário. O *Kahoot* classifica uma questão como difícil quando a porcentagem de respostas corretas é inferior a 35%.

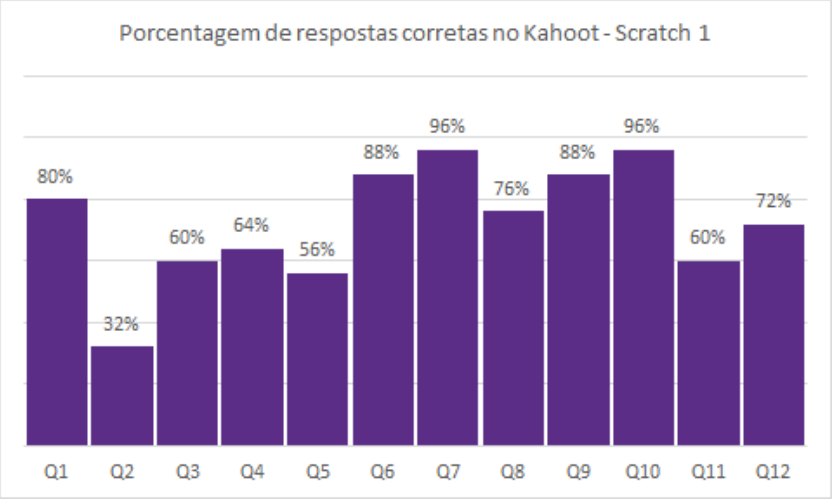
Figura 60 – Porcentagem de respostas corretas por alunos usando *Kahoot* – *Scratch.*



Fonte: A autora (2021).

A questão Q2 do primeiro *quiz* (FIGURA 61) foi classificada como difícil pelo *Kahoot* e tinha como objetivo avaliar se os alunos sabiam “Qual é o propósito dos operadores lógicos?”. Entre as respostas incorretas, aquela com o maior número de respostas é que afirmava que o objetivo dos operadores lógicos é retornar um número ao invés de um valor booleano.

Figura 61 – Porcentagem de respostas corretas no *Kahoot* – *Scratch* 1.

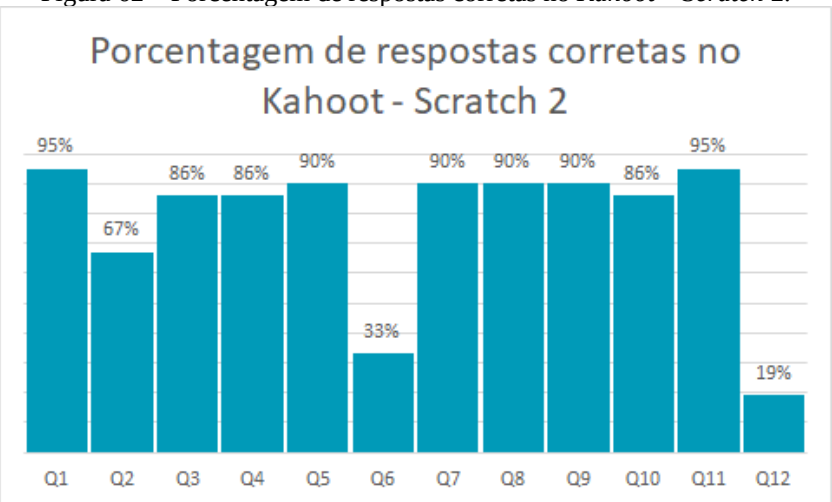


Fonte: A autora (2021).

Na questão Q6 (APÊNDICE E), no segundo *quiz*, os alunos tiveram dificuldade em compreender a funcionalidade do bloco de cenário de mudança, confundindo com o bloco que muda o próximo cenário e não o específico.

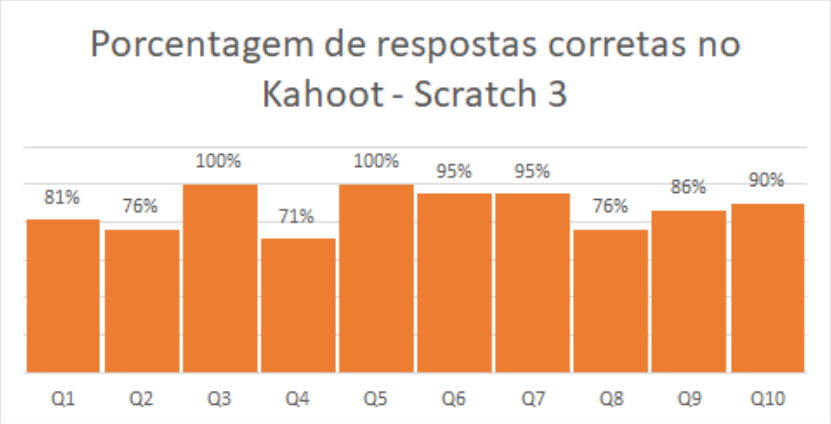
No segundo *quiz* na questão Q12 (APÊNDICE E), os alunos não entenderam a funcionalidade do bloco "repita até" onde contém uma expressão *booleana*, confundido com o bloco de "repetição de N" vezes. Finalmente, no terceiro *quiz*, os alunos não apresentaram mais dificuldades nas funcionalidades dos blocos.

Figura 62 – Porcentagem de respostas corretas no *Kahoot* – *Scratch* 2.



Fonte: A autora (2021).

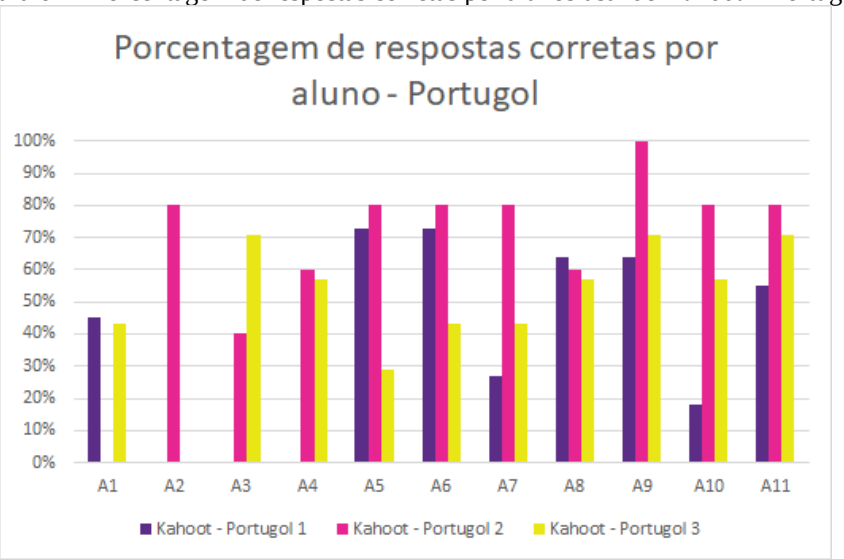
Figura 63 – Porcentagem de respostas corretas no *Kahoot* – *Scratch* 3.



Fonte: A autora (2021).

Sobre o questionário a respeito de questões teóricas, definição e sintaxe do Portugol, as Figuras 66, 67 e 68 mostram a porcentagem de respostas corretas para as questões. Os três *quizzes* realizados no *Kahoot* relacionados ao Portugol totalizaram 23 questões. A porcentagem de respostas corretas nos *quizzes* 1, 2 e 3 é de 44%, 57% e 51%.

Figura 64 – Porcentagem de respostas corretas por alunos usando *Kahoot* – Portugol.



Fonte: A autora (2021).

As questões Q2, Q8 e Q11 do primeiro *quiz* sobre Portugol (FIGURA 66) foram classificadas como difíceis pelo *Kahoot*. A Q9 foi uma questão anulada, pois não tinha resposta correta. A Q2 do *Kahoot* para o Portugol tinha o intuito de avaliar se os alunos sabiam "Os operadores lógicos suportados no Portugol". Entre as respostas incorretas, aquela com o maior número de respostas foi que eles confundiram com os operadores relacionais. A Q8 (FIGURA 65) apresentou um código e o estudante deveria responder se o código compilava ou não. A maioria escolheu a opção verdadeira, mas a resposta certa é falsa, pois falta um "fimse" no

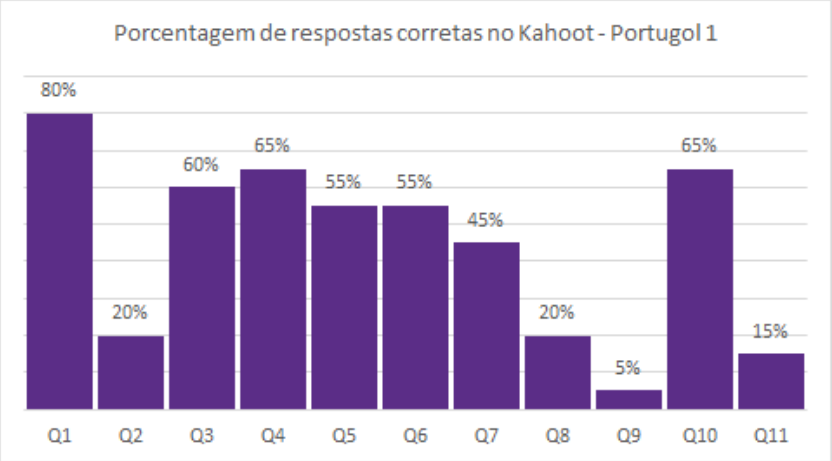
código. Por fim, a Q11 apresentou um algoritmo que pediu a saída do código que calculava os números pares entre 0 e 5. Os estudantes escolheram a opção 2 e 4, porém, a resposta correta é: “0, 2 e 4”. Nesta questão, eles não incluíram o 0 (zero) como número par e dentro do laço de repetição “para”.

Figura 65 – Código da Q8 do Portugol.

```
Var

   numero : inteiro
Inicio

   numero <-100
   para numero de 1 ate 100  faca
      se (numero%10=0)          entao
         escreval ("Múltiplo de 10", numero)

      fimpara

Fimalgoritmo
```

Fonte: A autora (2021).

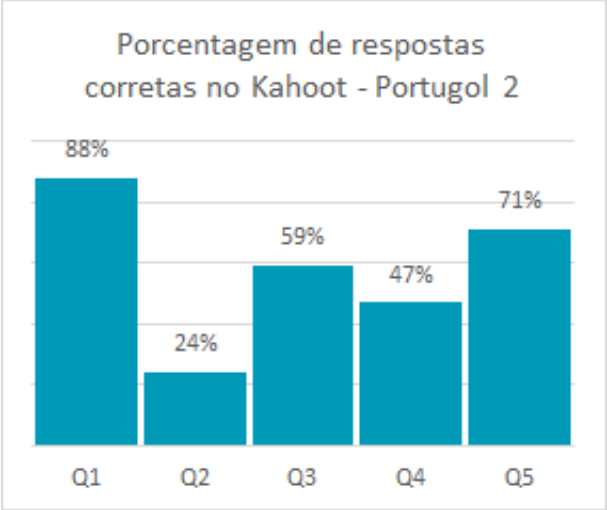
Figura 66 – Porcentagem de respostas corretas no *Kahoot* – Portugol 1.



Fonte: A autora (2021).

O segundo *quiz* na questão Q2 (APÊNDICE H) (FIGURA 67) pediu para o estudante escolher a saída do código que tinha um laço de repetição que somava dois a cada iteração. O laço começava em 0 e terminava em 4. A resposta correta é 10, mas a maioria dos estudantes escolheu a opção 8, tendo dificuldade em entender que são 5 elementos.

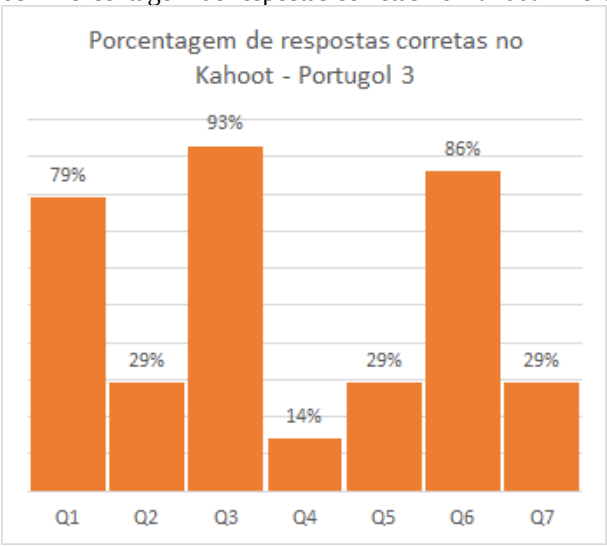
Figura 67 – Porcentagem de respostas corretas no *Kahoot* – Portugol 2.



Fonte: A autora (2021).

No terceiro *quiz* do Portugol (APÊNDICE I) (FIGURA 68), as questões Q2, Q4, Q5 e Q7 foram classificadas como difíceis pelo *Kahoot*. Na Q2 apresentou-se a atribuição de valores a variáveis e os estudantes deveriam escolher a opção correta. A maioria errou a resposta, mostrando dificuldade em fazer atribuição de variável. Na Q4 apresentava-se um vetor com 3 valores e tinha condicionais que, ao final, possuíam o valor final das verificações dos condicionais. Os estudantes erraram a resposta, pois tiveram dificuldade de usar o índice e o valor dos elementos dos vetores. A Q5 é muito similar à questão Q4, mas possuía um *loop* e um condicional. Ao final, tinha um resultado de soma dos valores do vetor. Por fim, a Q7 apresentou um vetor com cinco elementos e, no final, pedia o resultado dos múltiplos de um número no qual seria realizado dentro de um *loop*.

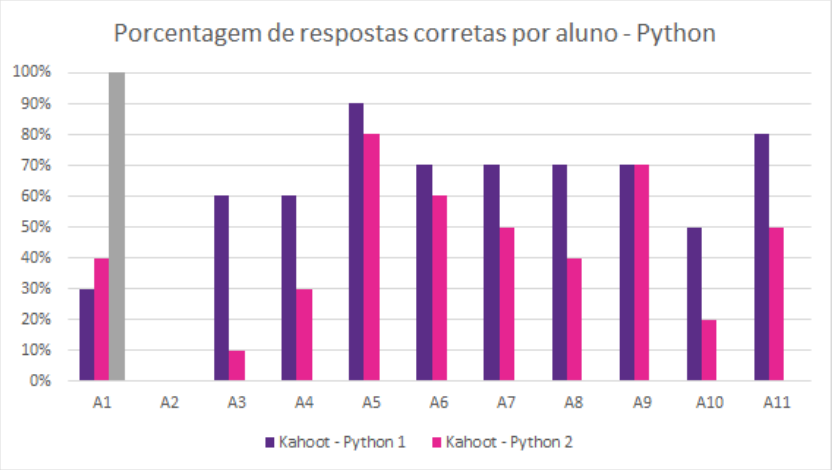
Figura 68 – Porcentagem de respostas corretas no *Kahoot* – Portugol 3.



Fonte: A autora (2021).

Em relação ao *quiz* sobre teoria, definição e sintaxe de *Python*, as Figuras 69 e 70 mostram a porcentagem de respostas corretas para as questões. A porcentagem de respostas corretas nos *quizzes* 1 e 2 é de 44%, 55% e 41%, respectivamente, e a Figura 69 apresenta o percentual de respostas corretas para cada *quiz*.

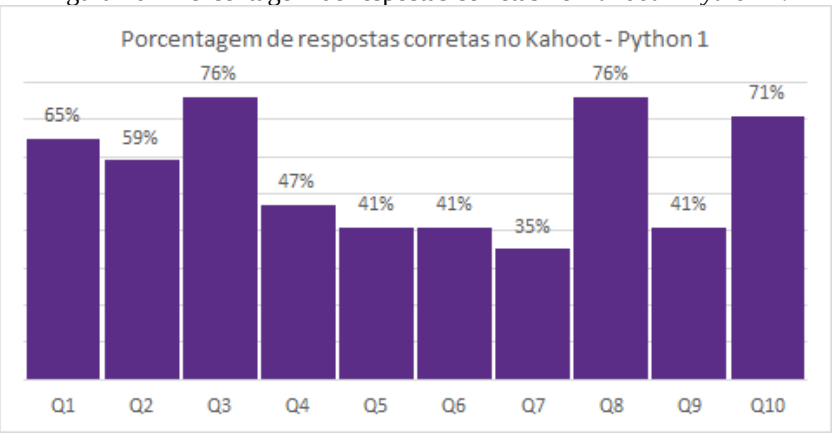
Figura 69 – Porcentagem de respostas corretas por alunos usando *Kahoot – Python*.



Fonte: A autora (2021).

Os dois *quizzes* realizados no *Kahoot* relacionados ao *Python* totalizaram vinte questões. O primeiro *quiz* de *Python*, a Q7 (APÊNDICE J) (FIGURA 70), foi classificado como difícil pelo *Kahoot*. Esta questão tinha o intuito de dizer a posição de uma letra em uma palavra. Para isto, foi usada uma função com dois parâmetros, o nome e a letra. O *loop* em *Python* começou em 0 e os alunos tiveram dificuldade de entender que o 0 era a posição 1 do algoritmo.

Figura 70 – Porcentagem de respostas corretas no *Kahoot – Python* 1.

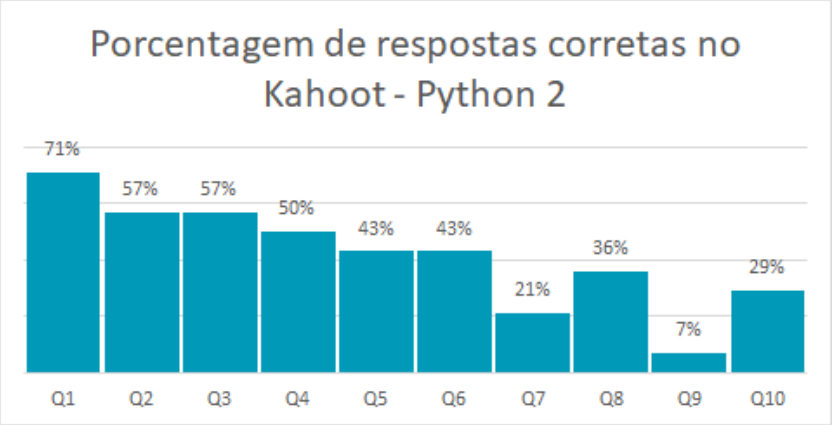


Fonte: A autora (2021).

No segundo *quiz* de Python (APÊNDICE J) (FIGURA 71), as questões classificadas como difíceis pelo *Kahoot* foram as Q7, Q9 e Q10. As três tinham em comum o pedido de

classificar a saída do código correto e a utilização de *loop* e condicionais. Os discentes, neste momento, tiveram dificuldade em fazer operações de variáveis dentro do *loop*.

Figura 71 – Porcentagem de respostas corretas no *Kahoot* – *Python* 2.

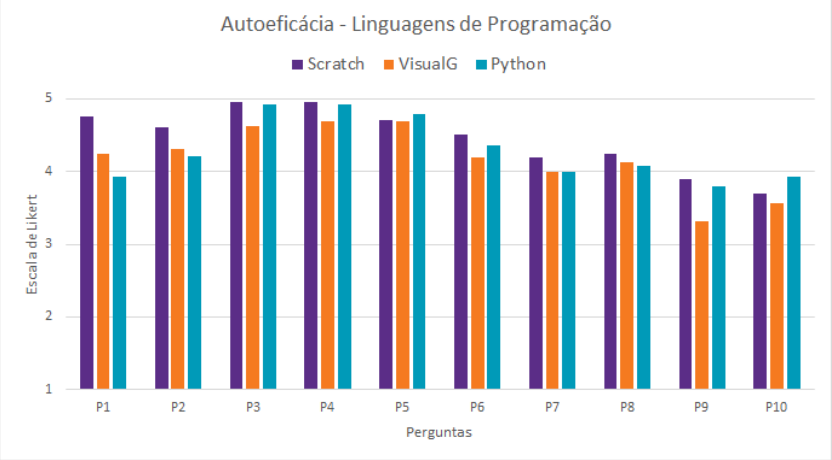


Fonte: A autora (2021).

6.5 RESULTADO DO QUESTIONÁRIO DE AUTOEFICÁCIA DO *SCRATCH*, PORTUGOL E *PYTHON*

O questionário de autoeficácia foi realizado de forma anônima e seu resultado pode ser visualizado na Figura 72. As respostas das perguntas P9 e P10 possuem níveis mais baixos pois são questões que apresentam a avaliação de poder completar um projeto sozinho apenas pelo manual ou por fazer um projeto do zero sem alguém ajudá-lo a começar. Ainda sobre as perguntas P9 e P10, a linguagem nas demais respostas, os alunos permanecem confiantes.

Figura 72 – Resultado do questionário de autoeficácia para todas as linguagens de programação.



Fonte: A autora (2021).

6.6 DISCUSSÃO

Nesta seção, discutiu-se a utilização do Modelo Conceitual *Skills-CT*, usado em um estudo realizado em um módulo de Lógica de Programação e *Python* de um curso de formação profissional em Desenvolvimento de *Software*, que tem como objetivo fazer a relação entre os Pilares do PC e as habilidade do Pensamento Computacional, bem como classificar os Estágios Cognitivos dos estudantes a partir da análise das habilidades do Pensamento Computacional e dos níveis dessas habilidades e fornecer ao professor uma ferramenta para avaliar as habilidades do PC e a aprendizagem do discentes e estimular a aprendizagem criativa.

Sobre a classificação dos Estágios Cognitivos no *Scratch*, a pesquisa mostrou que mesmo alunos com experiência anterior em programação possuem Estágios Cognitivos ainda não desenvolvidos em diversas habilidades de Pensamento Computacional. Isto pode ser uma indicação de que o ensino de habilidades de Pensamento Computacional deve ser priorizado em relação ao ensino focado na sintaxe de uma linguagem de programação.

Em relação aos Pilares do PC, pode-se inferir que os alunos apresentaram o Estágio Cognitivo Sensório-Motor com maior frequência nas habilidades de PC e mais dificuldade nos primeiros Pilares do PC. Os três primeiros Pilares do Pensamento Computacional, que são Decomposição, Reconhecimento de Padrão e Abstração, estão relacionados à construção da solução, à Lógica e ao Controle de Fluxo, que é uma habilidade usada na formalização dessa solução através do algoritmo.

Os resultados mostraram que os alunos tiveram mais dificuldades na criação da solução do que no algoritmo, uma vez que o *Scratch* é uma linguagem de programação visual e fácil de usar. As soluções ótimas para as atividades do *Scratch* podem ser vistas no Quadro 7. Nenhum aluno, mesmo os que tiveram contato com programação antes de entrar no curso, não alcançou o Estágio Cognitivo Operacional Concreto.

Quadro 7 – Solução ótima para as atividades do Scratch.

Conceitos de PC X Estágios Cognitivos				
Lógica	**Paralelismo**	**Representação de Dados**	**Controle de Fluxo**	**Abstração**
Operacional Concreto	Operacional Concreto	Pré-Operacional	Operacional Concreto	Operacional Concreto

Fonte: A autora (2021).

As habilidades do PC que apresentaram os níveis Sensório-Motor e Pré-Operacional

com maior frequência em todas as linguagens foram o Paralelismo e a Abstração da Decomposição do Problema. Embora o estudo não tenha como objetivo comparar os alunos entre si, mas, sim, um aluno com sua evolução, esse foi um fato que se repetiu em quase todas as análises dos discentes. Depreende-se que a razão é porque o estudante do Sensório-Motor tem dificuldade em acompanhar a execução do código e o Pré-Operacional ainda tem que rastrear uma parte do código linha a linha e não por execução, bem como ainda tem dificuldade em entender a recursão.

Sobre os Estágios Cognitivos apresentados nos resultados do uso do Portugol, observou-se que o Pré-Operacional, presente quando o nível da habilidade do PC está em desenvolvimento, foi recorrente nas habilidades de Lógica, Paralelismo e Abstração e Decomposição de Problemas.

Infere-se que os discentes que apresentaram esse Estágio Cognitivo possuem mais dificuldade nos quatro primeiros Pilares do PC. Os três primeiros – Decomposição, Reconhecimento de Padrão e Abstração –estão relacionados à construção da solução e à Lógica, que é uma habilidade usada na formalização desta solução através do algoritmo. Essas habilidades do PC e o Estágio Cognitivo foram melhorados em relação ao *Scratch*, pois as atividades foram direcionadas de acordo com os resultados dos Estágios Cognitivos para amenizar os problemas nessas habilidades. Ademais, em uma linguagem de programação por texto, há menos abstrações para solução do que em uma linguagem visual e, mesmo assim, os Estágios Cognitivos aumentaram com a utilização do Portugol por causa das atividades direcionadas.

No que concerne aos Estágios Cognitivos apresentados no módulo de *Python*, das questões de 3 a 8, as quais os problemas foram elaborados e resolvidos pelos alunos, percebeu-se o Estágio Cognitivo Operário Concreto mais presente. Entende-se que isto é resultado da aprendizagem criativa, uma vez que o discente usou os processos de personalização das informações, de comparação com os assuntos de programação aprendidos a partir de interpretação e pontos de vistas próprios e de geração de novas ideias ao criar suas próprias questões que resolviam um problema do seu contexto social e das suas vivências. Com esses resultados, observa-se que o estudante se transforma em sujeito do seu processo de aprendizado, incorporando e produzindo o conhecimento de maneira personalizada, ativa e criativa.

O Modelo *Skills-CT* possui uma relação entre os Pilares do Pensamento Computacional e *Creative Problem Solving* (CPS) com o objetivo de ajudar o professor a criar cenários de aprendizagem criativos dando ênfase na resolução de problemas. A criação de atividades utilizou a estrutura apresentada na seção 4.2, ajudando a estimular a aprendizagem criativa,

podendo ser observada nas soluções criativas dos estudantes através da criação de *storytelling*, formas de apresentar a resolução usando elementos como sons e animação no *scratch* e na linguagem de programação procedural por meio de mensagens divertidas.

O *Python* é uma linguagem de sintaxe simples, porém, a IDE usada, *VSCode*, foi profissional. O *VSCode* é um editor de código-fonte gratuito desenvolvido pela *Microsoft* e pode ser usado em diversos Sistemas Operacionais além do *Windows*, como *Linux* e *macOS*. Ele inclui suporte à depuração, controle de versionamento *Git*, realce de sintaxe, complementação inteligente de código e refatoração de código.

Os alunos evoluíram nos Estágios Cognitivos apresentando o Operacional Concreto na maioria das Habilidades do PC, com nível de proficiência. Adicionalmente, vê-se que as habilidades do PC foram desenvolvidas pelos estudantes através da análise dos códigos dos alunos, por meio das atividades criadas e a reformulação das aulas, dando ênfase às dificuldades enfrentadas por eles durante a disciplina, sendo esta avaliação realizada durante aulas e atividades e não apenas nas atividades avaliativas valendo nota.

As atividades de *Python* ficaram mais complexas em relação às outras linguagens, porém foram mais direcionadas, baseando-se nas dificuldades dos estudantes, visando melhorar os Estágios Cognitivos e estimular a aprendizagem criativa. Este progresso é consistente com o modelo de ondas sobrepostas em que, conforme os estudantes acumulam habilidades de raciocínio abstrato, eles exibem menos comportamentos característicos de estágios anteriores de desenvolvimento.

O Quadro 8 apresenta a solução ótima das atividades para Portugol e *Python*, que foi alcançada parcialmente pelos alunos, mostrando que eles desenvolveram as habilidades de PC.

Quadro 8 – Solução ótima para as atividades do Portugol e *Python*.

Conceitos de PC X Estágios Cognitivos				
Lógica	**Paralelismo**	**Representação de Dados**	**Controle de Fluxo**	**Abstração**
Operacional Concreto	Operacional Concreto	Operacional Concreto	Operacional Concreto	Operacional Concreto

Fonte: A autora (2021).

Os três questionários feitos no *Kahoot* mostraram que os alunos tiveram dificuldades com questões relacionadas à Lógica e *loop*, o que também se refletiu nos Estágios Cognitivos, pois a maioria dos discentes possuía as habilidades Lógica e Controle de Fluxo, que está

relacionada ao uso do *loop* dos Estágios Cognitivos Sensório-Motor e Pré-Operacional. Isto significa a dificuldade dos alunos na utilização desses elementos para desenvolver seus algoritmos. Observa-se que a Lógica e o Controle de Fluxo estão vinculados ao Pilar denominado Algoritmo no Pensamento Computacional.

Dentre as dificuldades apresentadas pelos alunos, além dos aspectos teóricos, observou-se, através dos resultados, a utilização dos Pilares do PC na resolução de problemas.

Em relação ao questionário de autoeficácia, na maioria das questões os alunos responderam que estão confiantes para realizar as atividades. Assim, denota-se que o *Scratch* tem o maior índice de confiança e que é uma ferramenta fácil e intuitiva de usar. Porém, mesmo o *Visualg* sendo uma ferramenta de fácil utilização, cuja linguagem de programação utilizada, o Portugol, propicia o desenvolvimento de pseudocódigo em português, ele apresentou um índice de confiança menor do que o *Python*, utilizado em uma IDE profissional como o *VSCode*.

Com os dados do gráfico da Figura 65, compreende-se que o Portugol, sendo usado no *Visualg*, possui alguns pontos que explicam a menor confiança no resultado do questionário de autoeficácia. O *Visualg* não apresenta informações detalhadas sobre os erros, mas, muitas vezes, só apresenta o erro sem especificá-lo, possuindo uma sintaxe fortemente tipada, ao contrário do *Python*, que tem uma sintaxe mais simples e, por possuir uma comunidade mais ativa com mais sites/fóruns, mesmo que as informações de erros não sejam feitas para iniciantes, eles têm a possibilidade de consultar a comunidade.

6.7 CONSIDERAÇÕES SOBRE O CAPÍTULO

O presente capítulo apresentou os resultados da utilização do Modelo *Skills-CT* por meio do Guia para o Ensino de Programação, Dr. *Scratch* e do classificador de código. O estudo foi realizado nos módulos de Lógica de Programação e *Python.* Os resultados apontam a relevância da utilização do *Skills-CT* no desenvolvimento das Habilidades do Pensamento Computacional, dos Estágios Cognitivos e resolução criativa dos problemas, indicando que o Modelo influenciou positivamente na aprendizagem de programação dos discentes, o que foi constatado por meio da análise dos códigos criados por eles e pelo resultado do questionário de autoeficácia em cada uma das linguagens usadas: *Scratch*, Portugol e P*ython*.

Através dos resultados, foi possível observar que os estudantes apresentaram progresso nos Estágios Cognitivos passando do Sensório-Motor em *Scratch* para o Operacional Concreto em *Python* na maioria das habilidades do PC. O próximo capítulo apresenta a Conclusão desta tese, abordando contribuições, limitações e trabalhos futuros.

7 CONCLUSÃO E TRABALHOS FUTUROS

Fonte: *Tumblr* Tiras Armandinho (2015)[7].

Este capítulo apresenta as contribuições deste trabalho, as limitações e as possibilidades para realização de futuras pesquisas.

O estudo aqui apresentado foi discorrido a partir da seguinte pergunta: como a relação entre os Estágios Cognitivos Neo-Piagetianos e o Pensamento Computacional pode auxiliar no diagnóstico da aprendizagem do aluno de programação e estimular a aprendizagem criativa? Para tanto, foi proposto um Modelo *Skills-CT* que explora os aspectos das Habilidades do PC e pode contribuir para amenizar a dificuldade dos estudantes em solucionar problemas através da programação e auxiliar a relacionar tais problemas com o Pensamento Computacional.

O Modelo *Skills-CT* visa avaliar as habilidades de Pensamento Computacional e os Estágios Cognitivos dos alunos em cada uma das habilidades de PC para auxiliar o professor em uma avaliação mais profunda dos desenvolvimentos de tais habilidades. Também abrange a autoavaliação do aluno por meio do questionário de autoeficácia no *Scratch*, no Portugol e no *Python*. Adicionalmente, o *Skills-CT* ajuda no desenvolvimento da aprendizagem criativa por meio da criação das atividades que possuem de forma implícita o CPS.

Os resultados indicam que o *feedback* do Dr. *Scratch* exibe apenas as habilidades do PC que foram utilizadas no código, porém, inferir o porquê da ausência e das dificuldades em usá-las foi possível através da avaliação dos Estágios Cognitivos em cada uma das habilidades. Por isso, o Modelo *Skills-CT* e o classificador baseado no Modelo corroboram para que o(a) professor(a) tenha uma avaliação mais precisa e completa sobre cada competência do PC, identificando aquelas que os alunos apresentam mais dificuldades e ainda diagnosticando em qual fase do Pilar do Pensamento Computacional elas ocorrem, propondo atividades que

[7]Disponível em: https://tirasarmandinho.tumblr.com/post/113819416619/tirinha-original. Acesso em: 16 jan. 2021.

estimulem a aprendizagem criativa.

Portanto, a presente pesquisa fornece evidências de que o desenvolvimento de habilidades do PC pode ser avaliado usando a estrutura de desenvolvimento cognitivo Neo-piagetiana no contexto de programadores iniciantes, ao passo que compreender como os discentes desenvolvem habilidades de programação permitirá aos professores acompanharem melhor aqueles que apresentam dificuldades no desenvolvimento dessas habilidades.

7.1 CONTRIBUIÇÕES DA PESQUISA

Considerando a perspectiva do desenvolvimento do Pensamento Computacional através da Programação, Resolução de Problemas e Aprendizagem Criativa, a pesquisa contribuiu para avaliar e propor atividades para auxiliar os estudantes a galgarem os Estágios Cognitivos no desenvolvimento das habilidades do PC. A seguir são listadas as principais contribuições do trabalho desenvolvido:

- Criação e utilização do Modelo *Skills-CT* proposto;
- Definição, desenvolvimento e utilização do classificador de código baseado no Modelo *Skills-CT;*
- Definição e utilização do Guia para o Ensino de Programação (GEP);
- Definição e utilização da Jornada do Estudante (JE);
- Avaliação do Modelo Conceitual *Skills-CT* para o desenvolvimento do Pensamento Computacional, Resolução de Problemas e estímulo da Aprendizagem Criativa, com intuito de auxiliar o processo de ensino-aprendizagem de alunos em programação;
- Estudo experimental realizado com discentes dos módulos de Lógica de Programação, com carga horária de 24h, e *Python*, com carga horária de 20h, em um curso de Desenvolvimento de *Software* que teve a duração de sete meses, possibilitando verificar a influência do modelo proposto no aprendizado dos estudantes, confirmando-se através da detecção dos Estágios Cognitivos;
- Promover a Aprendizagem Criativa por meio do Guia para o Ensino de Programação;
- O modelo conceitual do *Skills-CT* favorece o desenvolvimento do PC e dos Estágios Cognitivos dos discentes durante o processo de aprendizagem de programação, uma vez que apresenta o GEP para propostas de atividades focadas nas dificuldades dos estudantes;

- Agilizar o acompanhamento, por parte do(a) professor(a), do desenvolvimento das habilidades do Pensamento Computacional, bem como os Estágios Cognitivos dessas habilidades;
- Possibilitar a realização de tais atividades nos contextos do ensino remoto ou presencial.

7.2 LIMITAÇÕES

As limitações desta pesquisa serão apresentadas a seguir:

- A pesquisa utilizou o Modelo *Skills-CT*, contudo é necessária uma validação deste Modelo por meio de sua utilização em outros estudos e turmas;
- A aplicação do questionário de autoeficácia foi realizada apenas no final da Jornada do Estudante. A partir disso, observou-se a possibilidade de que a certeza de realização de uma tarefa por parte dos estudantes pode ter mudado durante a transição dos módulos. Como o questionário foi aplicado apenas no final, não há esses dados;
- O questionário de autoeficácia foi respondido de forma anônima, não podendo comparar a evolução dos estudantes em relação à sua convicção sobre a realização das atividades;
- A atividade de elaboração de questões por parte dos alunos foi aplicada apenas no módulo de *Python* e impactou positivamente nos resultados, mas não foi aplicada para as outras linguagens, *Scratch* e Portugol, não permitindo, por conseguinte, verificar seu impacto nos Estágios Cognitivos dos alunos;
- Pesquisa realizada em um pequeno grupo de alunos.

7.3 TRABALHOS FUTUROS

A presente pesquisa alcançou os objetivos traçados inicialmente e apresentou êxito ao usar o Modelo *Skills-CT*, o Guia para o Ensino de Programação e o classificador de códigos em um grupo de aprendizes de programação. Os resultados obtidos conduziram a possíveis trabalhos futuros, entre eles:

- Melhorar o classificador utilizando Aprendizagem de Máquina para fazer inferências mais precisas;

- Validar o Modelo *Skills-CT*;
- Disponibilizar o classificador, por meio de uma aplicação *web*, adicionando uma interface gráfica para envio dos códigos dos estudantes e implementar saídas com gráficos que apresentem as habilidades de PC e Estágios Cognitivos;
- Replicar o estudo em outras turmas de disciplinas de Introdução à Programação e usar a análise dos códigos ao longo das soluções e não apenas no final do projeto ou atividade;
- Aplicar o questionário de autoeficácia no início e no final do curso ou módulo, solicitando aos alunos que se identifiquem no questionário para que seja possível comparar sua avaliação da capacidade de realizar uma tarefa e as habilidades do PC e Estágios Cognitivos;
- Disponibilizar em um site o Guia para o Ensino de Programação e a Jornada do Estudante para que os professores possam usá-los.

Por fim, alguns trabalhos foram publicados a partir desta tese:

- SILVA, T. S. C.; MELO, J. C. B.; TEDESCO, P. C. A. R. Creative Learning in Problem Solving and Development of Computational Thinking. *In:* **International Conference on Computer Supported Education**. Springer, Cham, 2020, p. 199-215.
- SILVA, T. S. C.; MELO, J. C. B.; TEDESCO, P. C. A. R. The Creative Process in the Development of Computational Thinking in Higher Education. **CSEDU**, v. 1, 2020, p. 215-226.
- SILVA, T. S. C.; MELO, J. C. B.; TEDESCO, P. C. A. R. Pensamento Computacional e criatividade: uma análise da percepção dos discentes sobre a prática docente. In: **XXII TISE – Congresso Internacional de Informática Educativa**, Fortaleza, 2017, p. 461-466.

Em vista do exposto, o Modelo *Skills-CT*, o Guia para o Ensino de Programação e o classificador de códigos mostram-se como ferramentas promissoras para auxiliar no ensino de programação sob a perspectiva do desenvolvimento do Pensamento Computacional, da Resolução de Problemas e da Aprendizagem Criativa.

REFERÊNCIAS

ADALBERTO, E. M. L. Movimento Makers e a Aprendizagem Criativa no Ensino da Matemática no fundamental I. In: **Educação Matemática na Contemporaneidade:** desafios e possibilidades, São Paulo, p. 1-8, 2016.

ALENCAR, E. M. L. S.; FLEITH, D. S. **Criatividade:** múltiplas perspectivas. Brasília: Editora UnB, 2009.

AMABILE, T. M. Discovering the unknowable, managing the unmanageable. In: FORD, C. M.; GIOIA, D. A. **Creative Action in Organizations:** Ivory Tower Visions and Real World Voices. Newbury Park, CA: Sage Publications, p. 77-81, 1995.

AMARAL, A. L. S. N. do. **A constituição da aprendizagem criativa no processo de desenvolvimento da subjetividade**. 250 f. Tese (Doutorado em Educação) – Universidade de Brasília, Brasília, 2011.

ANDERSON, D. J. **Kanban:** successful evolutionary change for your technology business. Washington: Blue Hole Press, 2010.

ARAUJO, N.; PRIMO, T. T.; PERNAS, A. M. OntoScratch: ontologias para a avaliação do ensino de Pensamento Computacional através do Scratch. In: **Simpósio Brasileiro de Informática na Educação**, 31, 2020, on-line. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, p. 1823-1832, 2020.

ASKAR, P.; DAVENPORT, D. An investigation of factors related to self-efficacy for Java Programming among engineering students. **Turkish Online Journal of Educational Technology**, v. 8, n. 1, 2009.

ATEEQ, M. et al. C++ or Python? Which One to Begin with: A Learner's Perspective. In: **2014 International Conference on Teaching and Learning in Computing and Engineering – IEEE**, p. 64-69, 2014.

BANDURA, A. Toward a psychology of human agency. **Perspectives on psychological science**, v. 1, n. 2, p. 164-180, 2006.

BARR, V.; STEPHENSON, C. Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community?. **ACM Inroads**, v. 2, n. 1, p. 48-54, 2011.

BBC. **Introduction to computational thinking**. BBC, UK, 2017. Disponível em:
BENNEDSEN, J.; CASPERSEN, M. E. Teaching object-oriented programming-Towards teaching a systematic programming process. In: **Eighth Workshop on Pedagogies and Tools for the Teaching and Learning of Object Oriented Concepts**, 2004.

BLIKSTEIN, P. **O pensamento computacional e a reinvenção do computador na educação.** Blikstein Consultoria, 2008. Disponível em: http://www.blikstein.com/paulo/documents/online/ol_ pensamento_computacional.html. Acesso em: 16 jan. 2021.

BOOM, J. Commentary on: Piaget's stages: the unfinished symphony of cognitive development. **New Ideas in Psychology**, v. 22, n. 3, p. 239-247, 2004.

BOOM, K-D. et al. Relationship between computational thinking and a measure of intelligence as a general problem-solving ability. In: **Proceedings of the 23rd Annual ACM conference on innovation and technology in computer science education**, p. 206-211, 2018.

CABRAL, G. **A esperança audaz:** a pedagogia de Paulo Freire. Ultimato, 2015. Acesso em: https://www.ultimato.com.br/revista/artigos/353/a-esperanca-audaz-a-pedagogia-de-paulo-freire. Acesso em: 20 set. 2021.

CACHERO, C. et al. Impact of programming exposure on the development of computational thinking capabilities: An empirical study. **IEEE Access**, v. 8, p. 72316-72325, 2020.

CARLBORG, N. et al. The Scope of Autonomy Model: Development of Teaching Materials for Computational Thinking in Primary School. In: **Proceedings of the Conference on Creativity and Making in Education**, p. 37-44, 2018.

CASTRO, L. M. C. et al. Analyzing students' computational thinking practices in a first-year engineering course. **IEEE Access**, v. 9, p. 33041-33050, 2021.

CHUECHOTE, S. et al. A Neo-Piagetian Analysis of Algorithmic Thinking Development through the "Sorted" Digital Game. **Contemporary Educational Technology**, v. 12, n. 1, p. 1-15, 2020.

CSIKSZENTMIHALYI, M. Society, culture, and person: a systems view of creativity. **The Systems Model of Creativity**, p. 47-61, 1988.

CULLIGAN, N.; CASEY, K. **Exploring the Coding Behaviour of Successful Students in Programming by Employing Neo-Piagetian Theory**. Research Gate, 2021. Disponível em: https://www.researchgate.net/publication/350546233_Exploring_the_Coding_Behaviour_of_Successful_Students_in_Programming_by_Employing_Neo-Piagetian_Theory. Acesso em: 16 jan. 2021.

EPSTEIN, R. Generativity theory and creativity. In: RUNCO, M. A.; ALBERT, R. S. (Eds.), **Theories of creativity**. Sage Publications, Inc., 2005.

FELDMAN, D. Piaget's stages: the unfinished symphony of cognitive development. **New Ideas in Psychology**, v. 22, n. 3, p. 175-231, 2004.

FERRANCE, E. **Action research**. Providence, RI: LAB, Northeast and Island Regional Education Laboratory at Brown University, 2000.

FREIRE, P. **Pedagogia da Esperança:** reencontro com a Pedagogia do Oprimido. Rio de Janeiro: Paz e Terra, 1992.

GAL, L. et al. Suggesting a log-based creativity measurement for online programming learning environment. In: **Fourth ACM Conference on Learning@ Scale**, p. 273-277, 2017.

GIRAFFA, L.; MULLER, L.; MORAES, M. C. Ensinado Programação apoiada por um ambiente virtual e exercícios associados a cotidiano dos alunos: compartilhando alternativas e lições aprendidas. In: **Anais dos Workshops do Congresso Brasileiro de Informática na Educação**, p. 1330-1339, 2015.

GOMES, A. et al. A teacher's view about introductory programming teaching and learning Portuguese and Macanese perspectives. **IEEE Frontiers in Education Conference (FIE) Proceedings**, p. 1-8, 2017.

GOMES, A.; MENDES, A. A teacher's view about introductory programming teaching and learning: Difficulties, strategies and motivations. **IEEE Frontiers in Education Conference (FIE) Proceedings**, p. 1-8, 2014.

GUILFORD, J. P. **The nature of human intelligence**. New York: McGraw-Hill, 1967. https://www.bbc.com/education/guides/zp92mp3/revision/1. Acesso em: 10 mai. 2021.

GUN, M. **Técnicas de Criatividade.** Keep Learning, 2017. Disponível em: https://blog.keeplearning.school/. Acesso em: 12 mai. 2021.

IEEE Computer Society. **Computer Science Curricula 2013:** Curriculum Guidelines for Undergraduate Degree Programs in Computer Science. New York, NY, USA, 2013. Disponível em: https://www.acm.org/binaries/content/assets/education/cs2013_web_final.pdf. Acesso em: 17 jan. 2021.

IZU, C.; ALEXANDER, B. Using unstructured practice plus reflection to develop programming/problem-solving fluency. In: **Proceedings of the 20th Australasian Computing Education Conference (ACE '18)**. Association for Computing Machinery, New York, NY, USA, p. 25-34, 2018.

JIN, W.; JOHNSON, C. L.; DEKHANE, S. A Guided Inquiry Approach for Detecting and Developing Problem-solving Strategies for Novice Programming Students. In: **Proceedings of the 2020 ACM Southeast Conference (ACM SE '20).** Association for Computing Machinery, New York, NY, USA, p. 211-217, 2020.

KAFAI, Y.; PROCTOR, C.; LUI, D. From Theory Bias to Theory Dialogue: Embracing Cognitive, Situated, and Critical Framings of Computational Thinking in K-12 CS Education. **ACM Inroads**, v. 11, n. 1, p. 44-53, 2020.

KNIGHT, C. C.; SUTTON, R. E. Neo-Piagetian theory and research: Enhancing pedagogical practice for educators of adults. **London Review of Education**, v. 2, n. 1, p. 47-60, 2004.

KOULOURI, T.; LAURIA, S.; MACREDIE, R. D. Teaching introductory programming: a quantitative evaluation of different approaches. **ACM Transactions on Computing Education (TOCE)**, v. 14, n. 4, p. 1-28, 2014.

KOVALKOV, A.; SEGAL, A.; GAL, K. Inferring Creativity in Visual Programming Environments. In: **Proceedings of the Seventh ACM Conference on Learning@ Scale**, p. 269-272, 2020.

KUTSCHA, T. **Towards a Practical Application of the Neo-Piagetian Theory for Novice**

Programmers. Master's thesis, Radboud University Nijmegen, 2017. Disponível em: https://www.ru.nl/publish/pages/769526/timkutschamasterthesis.pdf. Acesso em: 16 jan. 2021.

LIKERT, R. A technique for the measurement of attitudes. **Archives of Psychology**, v. 22, 1932.

LIN, P. H.; CHEN, S. Y. Design and evaluation of a deep learning recommendation based augmented reality system for teaching programming and computational thinking. **IEEE Access**, v. 8, p. 45689-45699, 2020.

LISTER, R. Concrete and Other Neo-Piagetian Forms of Reasoning in the Novice Programmer. In: **Proceedings of the Thirteenth Australasian Computing Education Conference (ACE '11).** Australian Computer Society, Inc., AUS, p. 9-18, 2011.

LISTER, R. On the cognitive development of the novice programmer: and the development of a computing education researcher. In: **Proceedings of the 9th Computer Science Education Research Conference (CSERC '20).** Association for Computing Machinery, New York, NY, USA, Article 2, p. 1-15, 2020.

MACKINNON, D. W. The nature and nurture of creative talent. **American Psychologist**, v. 17, n. 7, p. 484-495, 1962.

MALIK, S. I.; MATHEW, R.; HAMMOOD, M. M. PROBSOL: A web-based application to develop problem-solving skills in introductory programming. In: **Smart Technologies and Innovation for a Sustainable Future**, p. 295-302, 2019.

MARTINEZ, A. M.; TACCA, M. C. V. R. **A complexidade da aprendizagem**. Campinas: Editora Alínea, 2009.

MEDEIROS, R. P. **Hello, world**: uma análise sobre dificuldades no ensino e na aprendizagem de introdução à programação nas universidades. 192 f. Tese (Doutorado) – Pós-Graduação em Ciência da Computação, Universidade Federal de Pernambuco, Recife, 2019.

MEDEIROS, R. P.; RAMALHO, G. L.; FALCÃO, T. P. A Systematic Literature Review on Teaching and Learning Introductory Programming in Higher Education. **IEEE Transactions on Education**, v. 62, n. 2, p. 77-90, 2019.

MORENO-LEÓN, J.; ROBLES, G.; ROMÁN-GONZÁLEZ, M. Dr. Scratch: análisis automático de proyectos Scratch para evaluar y fomentar el Pensamiento Computacional. **Revista de Educación a Distancia (RED)**, n. 46, 2015.

NOBRE, I. A. M.; MENEZES, C. S. Suporte à Cooperação em um Ambiente de Aprendizagem para Programação (SAmbA). In: **XII Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação – SBIE)**, p. 337-347, 2002.

OLIVEIRA, C. M. et al. Utilização de Desafios para o Desenvolvimento do Pensamento Computacional no Ensino Superior: um relato de experiência. In: **Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação – SBIE)**, p. 2005, 2019.

OLIVEIRA, L. H. D. **Exemplo de cálculo de Ranking Médio para Likert**. Notas de Aula. Metodologia Científica e Técnicas de Pesquisa em Administração. Mestrado em Adm. e Desenvolvimento Organizacional. PPGA CNEC/FACECA: Varginha, 2005.

OSBORN, A. **Your Creative Power**. London and New York: Read Books, 2008.

PELLEGRINO, J. W.; HILTON, M. L. (Ed.). National Research Council. **Education for life and work:** developing transferable knowledge and skills in the 21st century. Washington, DC: The National Academies Press, 2012.

PESSOA, F. I. R. et al. T-mind: um Aplicativo Gamificado para Estímulo ao Desenvolvimento de Habilidades do Pensamento Computacional. In: **Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação – SBIE)**, p. 645, 2017.

PETERANETZ, M. S. et al. Computational Creativity Exercises: An Avenue for Promoting Learning in Computer Science. **IEEE Transactions on Education**, v. 60, n. 4, p. 305-313, 2017.

PITEIRA, M.; HADDAD, S. Innovate in your program computer class: an approach based on a serious game. In: **Proceedings of the 2011 Workshop on Open Source and Design of Communication**, p. 49-54, 2011.

RESNICK, M. et al. Scratch: programming for all. **Commun. ACM**, v. 52, n. 11, p. 60-67, 2009.

RUBIO, M. A. Automatic Categorization of Introductory Programming Students. In: **International Joint Conference:** 12th International Conference on Computational Intelligence in Security for Information Systems (CISIS 2019) and 10th International Conference on EUropean Transnational Education (ICEUTE 2019). Springer, Cham, p. 302-311, 2019.

SANTANA, S. J.; OLIVEIRA, W. Desenvolvendo o pensamento computacional no ensino fundamental com Arduino e Scratch. In: **Encontro Nacional de Computação dos Institutos Federais (ENCOMPIF)**, 5, 2018, Natal. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2018.

SANTIAGO, A. D.; KRONBAUER, A. Um Modelo Lúdico para o Ensino de Conceitos de Programação de Computadores. In: **Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação – SBIE)**, v. 27, n. 1, p. 420-428, 2016.

SANTOS, R. P. dos; COSTA, H. A. X. Análise de Metodologias e Ambientes de Ensino para Algoritmos, Estruturas de Dados e Programação aos iniciantes em Computação e Informática. **INFOCOMP**, v. 05, n. 01, p. 41-50, 2006.

SCAICO, P. D.; DANTAS, V. F.; FARIAS, A. B. Atividades Virtuais de Apoio Aplicadas ao Ensino Introdutório de Programação. In: **XXXIX Latin American Computing Conference, Puerto azul/Naiguata/Venezuela, XXI Iberoamerican Congress on Higher Education in Computing (CIESC-2013)**, 2013.

SHELL, D. F. et al. Improving students' learning and achievement in CS classrooms through computational creativity exercises that integrate computational and creative thinking. In: **Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education**, p. 543-548, 2017.

SIEGLER, R. **Emerging Minds:** The Process of Change in Children's Thinking. New York: Oxford University Press, 1996.

SILVA, T. S. C.; MELO, J. C. B.; TEDESCO, P. C. A. R. Creative Learning in Problem Solving and Development of Computational Thinking. In: **International Conference on Computer Supported Education**, Springer, Cham, p. 199-215, 2020.

STERNBERG, R. J. **Wisdom, inteligence, and creativity synthesized.** Cambridge: Cambridge University Press, 2003.

TEAGUE, D. **Neo-Piagetian Theory and the novice programmer.** PhD by Publication, Queensland University of Technology, 2015.

TROIANO, G. M. et al. Is my game OK Dr. Scratch? Exploring programming and computational thinking development via metrics in student-designed serious games for STEM. In: **Proceedings of the 18th ACM International Conference on Interaction Design and Children (IDC '19).** Association for Computing Machinery, New York, NY, USA, p. 208-219, 2019.

TSCHIMMEL, K. C. **Sapiens e Demens no pensamento criativo do design**. 574 f. Tese (Doutorado) – Curso de Design, Departamento de Comunicação e Arte, Universidade de Aveiro, Portugal, 2010.

TYRÉN, M. et al. Considerations and technical pitfalls for teaching computational thinking with BBC micro: Bit. In: **Proceedings of the Conference on Creativity and Making in Education (FabLearn Europe '18).** Association for Computing Machinery, New York, NY, USA, p. 81-86, 2018.

VEERASAMY, A. K. et al. Relationship between perceived problem-solving skills and academic performance of novice learners in introductory programming courses**. Journal of Computer Assisted Learning**, v. 35, n. 2, p. 246-255, 2019.

VOURLETSIS, I.; POLITIS, P. Effects of a Computational Thinking Experimental Course on Students' Perceptions of Their Problem-Solving Skills. In: **Proceedings of the 2020 9th International Conference on Educational and Information Technology**. 2020. p. 14-20.

WAGH, A.; GRAVEL, B.; TUCKER-RAYMOND, E. The role of computational thinking practices in making: How beginning youth makers encounter & appropriate ct practices in making. In: **Proceedings of the 7th Annual Conference on Creativity and Fabrication in Education (FabLearn '17).** Association for Computing Machinery, New York, NY, USA, Article 9, p. 1-8, 2017.

WATSON, C.; LI, F. W. B. Failure rates in introductory programming revisited. In: **Proceedings of the 2014 conference on Innovation & technology in computer science education (ITiCSE '14).** Association for Computing Machinery, New York, NY, USA, p.

39-44, 2014.

WING, J. M. Computational thinking. **Commun. ACM**, v. 49, n. 3, 2006.

YOUNG, J. G. What is creativity? **The Journal of Creative Behavior**, v. 19, n. 2, p. 77-87, 1985.

ZAVADIL, P.; SILVA, R. P. Modelo teórico do pensamento e processo criativo em indivíduos e em grupos de design. **Design & Tecnologia**, v. 6, n. 12, p. 01-20, 2016.

APÊNDICE A – ATIVIDADES DO *SCRATCH*

At1. Cremildo foi fazer um saque no caixa eletrônico do Banco, mas, para fazer o saque, precisa:

- Ter dinheiro no banco;
- O valor do saque de ser menor ou igual ao que tem na conta;
- Debitar (subtrair) o valor do saque do que já possui na conta;
- Faça no *Scratch* um algoritmo que simule um caixa eletrônico.

At2. Cremildo quer calcular o seu salário já que fez muitas horas extras. Elabore um algoritmo no *Scratch* que leia o número de horas trabalhadas de Cremildo e calcule o salário sabendo que ele ganha R$ 30,50 por hora. Quando o número de horas exceder a 40, calcule o excesso de pagamento armazenando-o na variável. Caso contrário, zerar tal variável. A hora excedente de trabalho vale o dobro da hora de trabalho. No final do processamento, mostre na tela o valor do salário total (com o *money* extra).

At3. José é pescador e precisa da sua ajuda para controlar o rendimento diário de seu trabalho. Toda vez que ele traz um peso de peixes maior que o estabelecido pelo regulamento de pesca do estado de São Paulo (50 quilos), deve pagar uma multa de R$ 4,00 por quilo excedente. José precisa que você faça um algoritmo no *Scratch* que leia a variável Peso (peso de peixes) e verifique se há excesso. Se houver, gravar na variável Excesso (Excesso) e na variável Multa o valor da multa que José deverá pagar. Caso contrário, mostrar tais variáveis com o conteúdo ZERO.

At4. Na cidade de *Springfield* vai acontecer uma eleição para prefeito. Os candidatos são:

- Homer Simpson
- Nad Flenders
- Selma Bovier
- dna Krabappel

Você deve simular uma urna para essa eleição com esses candidatos e ela deve contabilizar os votos através do clique nas fotos dos candidato(a). Ao final, o seu programa deve exibir os votos de cada candidato, quem venceu a eleição, a foto do(a) vencedor(a) e quantidade de votos. A cidade possui 20 pessoas votantes.

At5. A prefeita de *Gotham* deseja fazer uma pesquisa entre seus habitantes. Faça um algoritmo para coletar dados sobre o salário e número de filhos de cada habitante e, após as leituras, escrever:
a) Média de salário da população;
b) Média do número de filhos;
c) Maior salário dos habitantes;
d) Percentual de pessoas com salário menor que R$ 900.

At6. Uma rainha requisitou os serviços de um monge e disse-lhe que pagaria qualquer preço. O monge, necessitando de alimentos, indagou à rainha sobre o pagamento, se poderia ser feito com grãos de trigo dispostos em um tabuleiro de xadrez, de tal forma que o primeiro quadro deveria conter apenas um grão e, nos quadros subsequentes, o dobro do quadro anterior. A rainha achou o trabalho barato e pediu que o serviço fosse executado, sem se dar conta de que seria impossível efetuar o pagamento. Faça um programa no *Scratch* para calcular o número

de grãos que o monge esperava receber (GASPAR, 2020).

At7. Faça um programa no *Scratch* que permita a leitura das notas de uma turma de 10 alunos. Calcular a média da turma e contar quantos alunos obtiveram nota acima desta média calculada. O programa deve exibir a média da turma e o resultado da contagem.

At8. Faça um programa no *Scratch* para ler o código e o preço de 10 produtos. Calcular e exibir na tela: o maior preço lido e a média aritmética dos preços dos produtos.

At9. O custo de um carro novo ao consumidor é a soma do custo de fábrica com a porcentagem do distribuidor e dos impostos (aplicados ao custo de fábrica). Supondo que o percentual do distribuidor seja de 20% e os impostos de 30%, faça um programa para ler o custo de fábrica de um carro, calcular e escrever o custo final ao consumidor.

At10. Ler a hora de início e a hora de fim de um jogo de Xadrez (considere apenas horas inteiras, sem os minutos) e calcule a duração do jogo em horas, sabendo-se que o tempo máximo de duração do jogo é de 24 horas e que o jogo pode iniciar em um dia e terminar no dia seguinte (CARVALHO, 2007).

REFERÊNCIAS

CARVALHO, F. P. de. **Exercícios e respostas de lógica de programação – algoritmos**. FACCAT – Faculdades Integradas de Taquara, 2007. Disponível em: https://fit.faccat.br/~fpereira/apostilas/exerc_resp_alg_mar2007.pdf. Acesso em: 15 nov. 2020.

GASPAR, W. **Exercício 20:** Quantos grãos de trigo a rainha pagou ao monge? Wagner Gaspar, 2020. Disponível em: https://wagnergaspar.com/exercicio-20-quantos-graos-de-trigo-a-rainha-pagou-ao-monge/. Acesso em: 15 nov. 2020.

APÊNDICE B – ATIVIDADES DO *VISUALG*

At1. Fazer uma calculadora que recebe dois números A e B e faça os seguintes cálculos:

- Soma (A+B), Subtração (A-B) Subtração (B-A);
- Multiplicação (A*B);
- Divisão (A/B), Divisão(B/A);
- Quadrado de A, Quadrado de B;
- Divisores inteiro de A, Divisores inteiros de B;
- Fatorial de A, Fatorial de B.

At2. Faça um programa que leia uma quantidade indeterminada de números positivos e conte quantos deles estão nos seguintes intervalos: [0-25], [26-50], [51-75] e [76-100]. A entrada de dados deverá terminar quando for lido um número negativo (AVILA, 2019).

At3. Em uma competição de ginástica, cada atleta recebe votos de sete jurados. A melhor e a pior nota são eliminadas. A sua nota fica sendo a média dos votos restantes. Você deve fazer um programa que receba o nome do ginasta e as notas dos sete jurados alcançadas pelo atleta em sua apresentação e depois informe a sua média, conforme a descrição acima informada (retirar o melhor e o pior salto e depois calcular a média com as notas restantes). As notas não são informadas ordenadas. Um exemplo de saída do programa deve ser conforme o exemplo abaixo (PYTHON BRASIL, 2016):

```
Atleta: Aparecido Parente
Nota: 9.9
Nota: 7.5
Nota: 9.5
Nota: 8.5
Nota: 9.0
Nota: 8.5
Nota: 9.7

Resultado final:
Atleta: Aparecido Parente
Melhor nota: 9.9
Pior nota: 7.5
Média: 9,04
```

At4. Faça um programa que leia um número indeterminado de valores, correspondentes a notas, encerrando a entrada de dados quando for informado um valor igual a -1. Após esta entrada de dados (PYTHON BRASIL, 2016):

a) Mostre a quantidade de valores que foram lidos;
b) Exiba todos os valores na ordem em que foram informados;
c) Exiba todos os valores na ordem inversa à que foram informados;
d) Calcule e mostre a soma dos valores;
e) Calcule e mostre a média dos valores;
f) Calcule e mostre a quantidade de valores acima da média calculada;
g) Calcule e mostre a quantidade de valores abaixo de sete.

At5. Foi feita uma pesquisa de canal de TV em várias casas da cidade de João Câmara. Para cada casa visitada, foi preenchida uma ficha contendo o número do canal (3, 5, 8, 11 ou 13) e o número de pessoas que estavam assistindo ao canal naquela casa. Faça um algoritmo que (PYTHON BRASIL, 2016):
a) Leia a informação da quantidade de casas participantes da pesquisa;
b) Receba as informações coletadas em cada uma das casas participantes da pesquisa: a informação do canal que estava sendo assistido e a quantidade de pessoas que estava assistindo aquele canal na casa;
c) Contabilize quantas pessoas estavam assistindo a cada canal (leve em conta a quantidade de pessoas que assistiam cada canal em cada casa).

At6. Faça um programa que solicite o nome do usuário e imprima-o na vertical de forma que o nome fique em formato de escada. Olhe a imagem abaixo (PYTHON BRASIL, 2016):

```
F
FU
FUL
FULA
FULAN
FULANO
```

At7. Faça uma função que retorne o reverso de um número inteiro informado. Por exemplo: 127 -> 72.

At8. Uma empresa de pesquisas precisa tabular os resultados da seguinte enquete feita a jogadores: “Qual o melhor jogo?”. As possíveis respostas são: 1- Valorant; 2- Among Us; 3- Free Fire; 4- Fall Guys; 5-Pixel Ripped 1995; 6-Celeste. Você foi contratado para desenvolver um programa que leia o resultado da enquete e informe, ao final, o resultado. O programa deverá ler os valores até ser informado o valor -1, que encerra a entrada dos dados. Não deverão ser aceitos valores além dos válidos para o programa (1 a 6 e -1). Os valores referentes a cada uma das opções devem ser armazenados num vetor. Após os dados terem sido completamente informados, o programa deverá calcular o percentual de cada um dos jogos e informar o vencedor da enquete. O formato da saída foi dado pela empresa e pode ser visto na imagem abaixo:

```
Jogos                Votos    %
-------------------  -----   ---
Valorant              1500   17%
Among Us              3500   40%
Free Fire             3000   34%
Fall Guys              500    5%
Pixel Ripped 1995      150    2%
Celeste                150    2%
-------------------  -----
Total                 8800

O jogo mais votado foi  Among Us, com 3500 votos, correspondendo a 40% dos votos.
```

REFERÊNCIAS

AVILA, S. **Faça um programa que leia uma [...]. A entrada de dados deverá terminar quando for lido um número negativo.** Instituto de Computação, Universidade Estadual de Campinas, 2019. Disponível em: https://www.ic.unicamp.br/~sandra/pdf/class/2019-1/mc102/2019-04-12-MC102KLMN-Aula13.pdf. Acesso em: 15 nov. 2020.

PYTHON Brasil. **Estrutura de Repetição.** Phyton Brasil, 2016. Acesso em: https://wiki.python.org.br/EstruturaDeRepeticao. Acesso em: 15 nov. 2020.

APÊNDICE C – ATIVIDADES DE *PYTHON*

At1. Criar um jogo Jokenpo em *Python*.

At2. Você deve fazer um programa em *Python* em que o usuário possa fazer seus pedidos para entregar na sua casa. O programa deve ter a funcionalidades:

- Itens com preços;
- Os itens devem ser agrupados por tipos de produtos (verificar as listas nas páginas seguintes);
- A cada pedido realizado deve aparecer o valor atualizado (um carrinho de compras);
- No final da comprar deve ser pedido: nome, endereço e forma de pagamento;
- As formas de pagamento: cartão de crédito ou débito ou dinheiro (Não precisa adicionar os dados do cartão);
- O programa deve exportar um arquivo com nome do usuário, endereço da entrega, todos os itens do pedido com os preços e a quantidade, o total a ser pago e a forma de pagamento.

At3, At4, At5, At6, At7, At8. Você deve criar as 6 perguntas (2 de lista, 2 de tupla e 2 de dicionário). Estas questões precisam resolver um problema.

Aluno 1 (At3, At4, At5, At6, At7, At8)

1. Crie uma lista telefônica com números de emergência em formato de tupla e permita que o usuário busque esses números.

2. Faça uma tupla com a capital dos estados do NE e que o usuário possa pesquisar através de uma lista de UF.

3. Faça uma lista de compras para casa que, no final, informe o valor total da compra, o item mais caro, o mais barato e todos os itens da lista com seu respectivo preço.

4. Usando uma lista, faça um programa com 4 receitas. Ao selecionar uma das receitas em uma lista, o código mostrará os ingredientes.

5. Usando dicionário, faça um código com dicionário onde, entrando com o CPF, retorne o nome e telefone do indivíduo.

6. Usando os dados a seguir, faça um código com dicionário que, ao pesquisar pelo CPF, mostre o nome, telefone, CPF, data de atendimento e tipo de atendimento do cliente em uma clínica médica.

Aluno 2 (At3, At4, At5, At6, At7, At8)

1. Crie dois vetores A e B, preencha cada um com 5 valores e, por último, calcule: C = A - B. Utilize tupla.

2. Dados determinados times, mostre os 5 primeiros em ordem alfabética e os quatro últimos. Utilize tupla.

3. Leia 10 valores e mostre a posição do maior e do menor no vetor. Use lista.

4. Leia um vetor de 10 posições e verifique se o número digitado já foi armazenado; se sim, leia novamente. No final, seu programa deve mostrar todos os números. Use lista.

5. Usando dicionário com os meses do ano e seus respectivos dias:
- Mostre os meses com 31 dias;
- Mostre os meses com 30 dias.

6. Usando dicionário, converta as seguintes temperaturas de C° para F°:
- Paris = 5°
- Recife = 26°
- Hong Kong = 16°
- Nova Iorque = 9°
- Barcelona = 15°
- Tóquio = 11°
- Egito = 14°
- Roma = 13°

Aluno 3 (At3, At4, At5, At6, At7, At8)

1. Usando tuplas, faça uma pesquisa com 10 pessoas em que elas terão que informar seu time preferido de Pernambuco. Depois exiba um *ranking* com o top 3 dos favoritos da galera. Exiba também o número de votos de cada time. Em caso de empate, refaça a pesquisa até desempatar.

2. Separe os 12 signos em 4 tuplas:
- Signos de Fogo – Áries, Leão e Sagitário;
- Signos de Terra – Touro, Virgem e Capricórnio;
- Signos de Ar – Gêmeos, Libra e Aquário;
- Signos de Água – Câncer, Escorpião e Peixes.

Depois peça para o usuário digitar um signo e, em seguida, informe de qual elemento é esse signo.

3. Use listas e faça um jogo de par ou ímpar, dividindo os números de 0 a 100 em 2 listas: ímpares e pares, onde o jogador digita um número e é exibido se o número digitado é ímpar ou par.

4. O técnico de um time quer saber a média de idade, o jogador mais velho e mais novo do seu clube, além de uma tabela onde mostra a idade de todos os 15 atletas. Faça um algoritmo usando 2 listas, uma para o nome e outra para a idade, que ajude o técnico com isso.

5. A professora quer saber quem é o mais alto, o mais baixo e a média de alunos de uma turma de 10 alunos. Faça um algoritmo que a ajude com esses dados, usando dicionário, onde as chaves são os alunos e os valores as alturas.

6. Utilizando o dicionário:
- {'Brasil': 5, 'Alemanha': 4, 'Itália': 4, 'França': 2, 'Argentina': 'Espanha': 1, 'Inglaterra': 1, 'Uruguai': 1}

Onde contém as seleções campeãs mundiais e a quantidade de vezes que ganharam a Copa do

Mundo. Faça um algoritmo que peça uma seleção e seja exibida a quantidade de vezes que ela foi campeã. Ao final, pergunte se o usuário quer verificar outra seleção; se sim, pergunte a seleção novamente e exiba o novo resultado.

Aluno 4 (At3, At4, At5, At6, At7, At8)

1. Crie tuplas para adicionar NOME, GÊNERO e DATA DE NASCIMENTO, depois imprima a lista conforme o exemplo:

- NOME: João -- GÊNERO: Cis Hétero -- DATA DE NASCIMENTO: 20-07-1989

2. Crie uma tupla com dados de Atores/Atrizes, incluindo NOME, SOBRENOME, ANO DE NASCIMENTO, FILMES e os ANOS DE ESTREIA DOS FILMES.

3. Faça um programa que simule uma lista de compras avulsa, com menu para inserir, deletar itens e encerrá-la. A cada escolha feita, imprima a lista para que o usuário leia e impeça que itens repetidos sejam adicionados.

4. Para tornar candidato(a) ao cargo de Presidente da República, o indivíduo deverá ter idade mínima de 35 anos:

- Para o cargo de Governador do Estado, o indivíduo deverá ter idade mínima de 30 anos;
- Para o cargo de Deputado Federal ou Prefeito, o indivíduo deverá ter idade mínima de 21 anos;
- Para o cargo de Vereador, o indivíduo deverá ter idade mínima de 18 anos.

Faça uma lista que armazena os nomes dos candidatos e para quais cargos poderão se eleger de acordo com suas idades. Não se qualificando para nenhum cargo, informar que o indivíduo não possui a idade mínima para se eleger a nenhum cargo e, por fim, mostrar a lista de candidatos a cada cargo.

5. Crie um dicionário adicionando NOME, IDADE e CIDADE de um indivíduo, imprima a IDADE e a CIDADE antes de serem atualizados. Atualize a IDADE e a CIDADE do indivíduo e imprima ambos os dados depois da atualização. Por fim, digite o comando para deletar a CIDADE.

6. Crie um dicionário com a entrada de USUÁRIO e SENHA de contas, imprima a quantidade de contas cadastradas no dicionário. Depois, pergunte qual conta se deseja acessar e disponibilize uma forma de encerrar o programa.

Aluno 5 (At3, At4, At5, At6, At7, At8)

1. A faculdade Ideias Remotas está aperfeiçoando seu ambiente virtual para permitir que seus alunos alterem a data de pagamento das mensalidades sem muita burocracia. Implemente um algoritmo que tenha definido um grupo de datas para pagamento e permita que o aluno faça a escolha de uma delas. Se a data escolhida estiver na predefinição, exiba uma mensagem de sucesso, a antiga data e a nova. Se não, exiba uma mensagem de erro e a data atual. A data de pagamento inicial é a menor. Para prevenir a alteração no grupo de datas, devem estar contidas em uma tupla.

2. Virgínia está estudando investimentos e, para entender o ritmo da economia global, resolveu "escovar moedas" a fim de comparar a valorização dos principais países. Implemente um programa para auxiliar Virgínia a converter valores em Real para as seguintes moedas: Dólar,

Euro, Libra, Peso e Franco. Quando um valor em Real for inserido, todos os valores convertidos das demais moedas devem ser atualizados. Os valores bases das moedas devem ser armazenados previamente em uma tupla para posterior consulta durante as conversões.

3. Will está planejando seguir uma série de dietas e estas sofrem alterações ao longo de um certo tempo, devido a ajustes. Elabore um algoritmo que permita a Will cadastrar e remover elementos contidos em uma lista que fazem parte da dieta atual dele. O algoritmo deve exibir os elementos atualizados.

4. Ana possui uma pequena biblioteca em casa, mas está com dificuldades para encontrar livros específicos. Devido a isso, ela pensou em catalogar todos os livros digitalmente. Desenvolva um algoritmo para ajudar Ana a catalogar os livros de sua estante utilizando listas compostas. O programa deve permitir a consulta por prateleiras (ex.: prateleira 1, prateleira 2, etc.) e exibir as obras das mesmas em ordem alfabética.

5. A empresa *Solweb* foi contratada para desenvolver um serviço *web* para autenticação de usuários. O serviço deve permitir o cadastro e *login* de N usuários. Implemente um algoritmo que registre os usuários com ID (número inteiro), *nick* e senha e permita o *login* usando o *nick* e a senha. Caso o usuário esteja registrado e o *login* for bem-sucedido, exibir uma mensagem de sucesso com o ID; caso contrário, exibir alerta de erro. Os dados dos usuários devem ser armazenados em dicionários.

6. Roberto tem o hábito de criar uma senha aleatória para todo tipo de conta que cria. Devido ao alto número de contas que Roberto possui, ele está tendo dificuldade para administrá-las. Crie um algoritmo que registre o ID (número inteiro), endereço do *site*, usuário e senha aleatória de Roberto em dicionários individuais e os exiba. Sempre que uma nova senha for inserida, a lista exibida deve ser atualizada.

Aluno 6 (At3, At4, At5, At6, At7, At8)

1. Maria tem uma tupla com consoantes ('b','d','f','l','m') e uma com vogais ('a','e','i','o','u') e quer formar as sílabas com essas letras.

2. Ester vai começar a fazer academia e suas aulas serão 3 vezes na semana: domingo, terça e quinta. Use tuplas para mostrar os dias da semana que terão aulas imprimindo uma mensagem de lembrete.

3. Joaquim coleciona filmes no computador, mas quer gravar alguns em CD. Precisa organizar sua lista por ordem alfabética e gravar os cinco primeiros.

4. João precisa atualizar a lista de empregados da sua firma. Lista antiga (“Maria”, “Joao”, “Pedro”, “Ana”, “Jose”, “Miguel”, “Ester”, “Leticia”, “Marcos”, “Matheus”). Para isso, ele precisa tirar quatro nomes, alternadamente, e depois incluir os novos nomes (Bruce, Clark, Diana, Peter, Henri).

5. Use dicionário para resolver o problema:

- Ana tem uma lanchonete e quer fazer um menu digital onde as pessoas possam ver o que querem escolher por nome ou pelo preço.

6. Lurdes tem um brechó e quer um programa que armazene todas as suas peças com seus

respectivos preços e que ela possa excluir e incluir itens depois. Use um dicionário e um menu com opções de incluir e excluir.

Aluno 7 (At3, At4, At5, At6, At7, At8)

1. Um bibliotecário precisa indexar a posição de alguns livros novos que chegaram no sistema da biblioteca. Após finalizar a indexação, ele precisa localizar a posição dos livros Dom Casmurro e Vidas Secas para informar a um determinado usuário. Segue a lista de livros com suas respectivas estantes:

- Dom Casmurro – estante 1 / O cortiço – estante 2 / Vidas secas – estante 3 / O alienista – estante 4 / Macunaíma – estante 5.

Use tupla para resolver o problema.

2. Um novo funcionário de supermercado quer saber em qual posição se encontra o produto Omo 400g no sistema. No sistema existem os seguintes produtos:

- Detergente IPE 500ML
- Detergente em Pó *Advanced* 2,5kg
- Secante para lava-louças *Finish* 250ml
- Omo 400g

O funcionário precisa encontrar a posição do produto para ajudar o cliente em sua compra. Use tupla para resolver o problema.

3. Um aluno deseja saber suas médias antes da postagem de seu professor para planejar com antecedência as suas férias. O aluno deseja criar um sistema que receba os nomes das disciplinas, suas duas notas e apresente em um boletim os nomes das disciplinas e suas respectivas médias. Use lista.

4. Uma nutricionista deseja criar um quadro para seu cliente que contemple as refeições que ele deve fazer durante um determinado dia e a quantidade de calorias necessárias de maneira que, durante a manhã, seja ingerido até 100 calorias, no almoço 400 calorias e 200 calorias no jantar. Segue a lista das refeições, alimentos e suas respectivas calorias:

- Café da manhã: Suco de pêssego natural – 77 e Maçã verde – 79
- Almoço: Suco de abacaxi natural – 100 e Salada de Alface, Tomate e Pepino – 43
- Jantar: Omelete – 170 e Café com açúcar – 62

Use lista.

5. O dono de uma locadora está organizando no sistema a lista de lançamentos da seguinte maneira:

- Nome do filme / gênero / valor de locação por 24h

Ao final da organização, o dono deseja exibir uma placa de maneira que os clientes consigam visualizar esses três dados e queiram efetuar a locação. Organize os filmes no sistema dessa forma e apresente em forma de lista. Segue a lista de filmes:

- Mar em Fúria – Ação – R$12,00/24h
- Duelo de Titãs – Drama – R$10,00/24h
- A Corrente do Bem – Romance – R$13,00/24h
- Homens de Honra – Drama – R$12,00
- A Nova Onda do Imperador – Comédia – R$12,00

Use dicionário.

6. Uma funcionária do DETRAN está organizando no sistema os alunos e as categorias que estão se formando. No dia das provas ela precisa apresentar os alunos por categoria para que os avaliadores se preparem para a aplicação. Segue a lista de alunos:

- Maria – B / Joana – A / Tereza – B / João – A e B / Pedro – C / José – A / Paulo – B /Bruna – B /Beatriz – A e B.

Use dicionário.

Aluno 8 (At3, At4, At5, At6, At7, At8)

1. Dona Margarida foi ao Hortifruti e, ao chegar lá, se deparou com a tabela dos produtos com os seguintes preços:

- Tomate = R$ 1,98 kg
- Cebola = R$ 1,48 kg
- Maçã = R$ 8,49 kg
- Banana = R$ 1,99 kg
- Alface = R$ 1,59 unidade
- Couve = R$ 4,85 unidade

Utilizando uma única tupla com elementos heterogêneos, mostre uma tabela com código, nome do produto e preços por Kg ou unidade (dependendo do produto). Após exibir a tabela, pergunte qual produto irá levar (utilizando o código do produto) e quantos quilos ou unidades desse produto escolhido, até continuar == n. Quando continuar da repetição, mostre no final o total gasto na compra. Dica: usar o índice ou contador pode ajudar no código do produto.

2. Mathias está ansioso para saber qual vai ser o *game* do ano no evento *The Game Awards*, considerado o "Oscar do videogame". Os indicados são: *The Last of Us Part 2*, *Animal Crossing*: *New Horizon*, *Final Fantasy 7 Remake*, *Ghost of Tsushima* e *Hades*. Para se preparar para o evento, decidiu jogar todos os jogos indicados e depois dará uma nota para cada jogo. Utilizando tupla faça uma tabela com os jogos a serem jogados e peça a nota de cada jogo. Ao final, mostre o jogo com a maior nota e a menor nota. Tanto a maior nota quanto a menor devem ter o nome do jogo relacionado.

3. Um grupo de alunos da área de saúde decidiu criar um mini *quiz* para conscientizar as pessoas a melhorarem seus hábitos e sua saúde. Eles não sabem quantas pessoas irão participar da pesquisa. As perguntas são as seguintes: "Está se alimentando com alimentos saudáveis?", "Está bebendo água suficiente?", "Está dormindo o suficiente?", "Vai ao médico anualmente?" e "Pratica exercícios físicos regularmente?". Utilize as perguntas dentro de uma lista. Se a resposta for sim para duas perguntas ou menos, exiba a seguinte mensagem: "Você está deixando a desejar bastante, cuide mais de sua saúde!". Se a resposta for sim para três perguntas, exiba a seguinte mensagem: "Está ficando preocupante, mas ainda há tempo para melhorar seus hábitos!". Se a resposta for sim para quatro perguntas, exiba a seguinte mensagem: "Está indo bem, mas pode melhorar!" Se a resposta for sim para cinco perguntas, exiba a seguinte mensagem: "Parabéns por se esforçar em manter uma boa saúde!" Quando finalizar as 5 perguntas da lista, perguntar se deseja fazer outra pesquisa; se a resposta for "não", exiba a quantidade de pessoas pesquisadas e saia do *loop*.

4. Josemar trabalha numa revendedora de carros usados, tem 4 carros na empresa e, a cada carro vendido, ele ganha 0,5 % (0,005 em decimal) de comissão:
Os carros são:

1) Classic ----- (R$17.000)
2) Onix ----- (R$45.000)

3) Palio ----- (R$15.000)
4) Logan ----- (R$40.000)

Utilizando lista composta, guarde o nome e o preço dos carros. Para vender o carro, mostre uma tabela com os carros e os preços. A pessoa escolhe o carro pelo código que vai de 1 a 4, do Classic ao Logan, respectivamente. A cada carro vendido, deve adicionar esse carro numa nova lista chamada carros vendidos. Quando o carro for vendido, a mensagem do carro vendido deve mudar. Por exemplo: Se o carro 1 for vendido a tabela ficará:
'Vendido'
2) Onix ----- (R$45.000)
3) Palio ----- (R$15.000
4) Logan ----- (R$40.000)
A venda acaba quando os carros acabarem ou quando a pessoa não quiser continuar. No final, mostre: quantidade de carros restantes (tamanho da lista de carros – tamanho da lista de carros vendidos), quantidade de carros vendidos e quanto Josemar lucrou no final em reais. Dica: para utilizar o código, coloque junto do nome do carro, por exemplo – o código aqui é 1: '1-Classic'.

5. Angelino mora em Recife e está planejando viajar para os estados que fazem fronteira com Pernambuco. Para isso decidiu fazer uma tabela com esses estados e capitais. Faça um dicionário para cada estado e insira dentro de uma lista. Cada dicionário deve incluir: Estado e Capital. Imprima na tela os Estados que fazem fronteira com Pernambuco e suas respectivas capitais utilizando lista e dicionário.
Alagoas – Maceió
Bahia – Salvador
Ceará – Fortaleza
Paraíba – João Pessoa
Piauí – Teresina

6. Alguns amigos saíram para jantar e ficou definido que seria feito um sorteio para quem iria pagar a conta. Para isso, faça um dicionário com os nomes dos amigos e faça um sorteio dentre estes para ver quem vai pagar a conta.

Aluno 9 (At3, At4, At5, At6, At7, At8)

1. Seu Carlos está abrindo uma lanchonete com apenas atendimento para delivery. Levando em conta que não haverá loja física, o atendimento será *on-line*. Ajude seu Carlos a montar o cardápio da lanchonete.

- Utilizando tuplas, monte todo o cardápio com comidas e bebidas. Por enquanto, não precisa do preço;
- Deve mostrar quantos itens existem no cardápio;
- O programa deve mostrar as comidas e bebidas separadas.

2. É comum no jogo de dominó ter uma rodada antes de iniciar o jogo, na qual todos os integrantes escolhem apenas uma pedra e quem tirar o maior valor inicia o jogo. Outra situação é quando o jogo fecha e inicia a contagem de pontos, aquele que tiver a menor pontuação vence. Simule um jogo de dominó com quatro jogadores onde devem entrar na tupla os valores para iniciar e outra para a contagem de pontos. No final deve mostrar a maior e menor peça, os valores ordenados.

3. Com o passar do tempo os signos ganharam bastante popularidade, vindo de encontro também temos o Zodíaco Chinês. Levando em conta que não é muito visto no nosso cotidiano,

faça um programa que ajude as pessoas a descobrirem seu signo do Zodíaco Chinês.
Considerando que ele é composto por animais com ciclo de 12 anos, uma maneira simples de identificar é com o ano do nascimento. Use tupla para resolver o problema.
Sendo:
0 – Macaco
1– Galo
2 – Cão
3 – Porco
4 – Rato
5 – Boi
6 – Tigre
7 – Coelho
8 – Dragão
9 – Serpente
10 – Cavalo
11 – Carneiro

4. Joana deseja fazer uma festa. Para isto, ela quer fazer uma lista com todos os seus amigos como convidados, mas Joana tem muitos amigos e se perdeu quando colocou os nomes e já não sabe qual nome adicionou. Faça um programa que ajude Joana a listar todos os seus amigos. Caso já tenha um nome na lista, mostrar uma mensagem para avisar que o nome já existe. No final, mostrar a lista completa e quantos convidados Joana listou. Use tupla para resolver o problema.

5. Em algumas faculdades, é normal os alunos trabalharem para a instituição para ganhar descontos na mensalidade. É comum esses estudantes trabalharem divulgando a instituição e procurando pessoas para efetuarem matrículas e participarem de vestibulares. Normalmente, o cadastro inicial é feito com caneta e papel, já que é mais fácil, rápido e mais seguro quando se trabalha na rua. Levando isto em consideração, faça um programa que simule a entrada desses dados: nome, idade e sexo. O programa deve rodar até o usuário não querer mais.

- É necessária a validação caso o sexo seja digitado errado;
- Mostrar quantas pessoas cadastradas, a média das idades e quem está acima dessa média.

6. Na rede SESI, alunos com pai ou mãe funcionários da indústria possuem o direito de receber descontos na mensalidade. Digamos que o desconto seja de 10%, faça um programa que simule a matrícula de um aluno, onde:

- Deve inserir nome e ano de nascimento;
- Perguntar ao usuário se tem ou não pais funcionários da indústria;
- Se não, mostrar nome, ano de nascimento, idade e quando iria terminar o ensino médio;
- Se sim, continuar as perguntas tais como:
  - Número da matrícula de funcionário da indústria;
  - Salário;
  - Calcular o desconto de 10% na mensalidade com valor de R$ 450;
  - Mostrar nome, ano de nascimento, idade e quando vai terminar o ensino médio;
  - Mostrar o valor do desconto e quanto vai ficar a mensalidade.

Aluno 10 (At3, At4, At5, At6, At7, At8)

1. Caio quer retirar todas as vogais das palavras, mas não sabe como. Crie um algoritmo (usando

tuplas) que faça este trabalho para Caio.

- Banana => BNN
- Acerola => CRL

2. Larissa quer saber a quantidade de vogais em uma palavra. Crie um algoritmo (usando tuplas) que faça este trabalho para Larissa.

3. Maria é professora e quer escrever as duas notas de X alunos da sala de aula e depois saber a média do aluno que ela escolher. Crie um algoritmo (usando lista) que ajude Maria a executar essa tarefa.

4. Ruana adicionou sem querer o mesmo item 2 vezes na lista de compras e agora quer remover e adicionar "Biscoito" no lugar. Crie um algoritmo (usando lista) que ajude Ruana com o seu problema.

- Feijão
- Arroz
- Macarrão
- Leite
- Ovos

5. João quer organizar sua lista de compras em ordem alfabética, mas não sabe como. Crie um algoritmo (usando dicionário) para ajudar João com sua lista de compras, sabendo que a ordem da lista atual é: {'Feijão': 5.50: 'Arroz': 4.75: 'Macarrão': 3.10: 'Leite': 6.45: 'Ovos': 9.20}. No final, exiba o valor total de compras.

6. Pedro quer mostrar para o seu chefe a lista dos setores que mais gostou na empresa, na ordem do mais alto para o mais baixo, mas não sabe como. Crie um algoritmo (usando dicionário) para ajudá-lo com sua lista, sabendo que a ordem da lista atual é: {'Administrativo': 10980.00, 'Financeiro': 4860.00, 'Recursos Humanos': 7090.00, 'Comercial': 6500.00, 'Operacional': 15500.00}.

Aluno 11 (At3, At4, At5, At6, At7, At8)

1. Uma família formada por pai (Antonio), mãe (Maria), filho (Caio) e filha (Bia) está com um pequeno problema para decidir quem, durante a semana, irá ficar responsável por lavar a louça e encher as garrafas de água da geladeira, então faça um programa usando tupla para sortear quem vai ficar com cada função, lembrando que a mesma pessoa não pode ficar com as duas funções e a filha (Bia) não pode lavar a louça porque só tem 9 anos.

2. Cinco amigos decidem se reunir para assistir um filme na casa de um deles entre o horário das 15h até 22h, mas cada um deles quer assistir um filme diferente que são:

- O Poço
- Sementes Podres
- Bacurau
- Corra
- Um Lugar Silencioso

Faça um sorteio utilizando tupla para decidir o filme, porém se eles decidirem assistir ao filme depois das 18h horas, só poderão assistir os filmes de terror que são Corra ou Um Lugar Silencioso.

3. Tiago, nessa *Black Friday*, resolveu comprar um fone de ouvido e está pesquisando o preço em várias lojas. Faça um programa usando lista onde Tiago vai digitar o nome da loja e o valor do fone de ouvido. No final exiba qual loja tem o menor e o maior preço.

4. Todo mês Mateus economiza um certo valor indefinido do seu salário. Ele deseja saber qual mês do ano economizou o maior e o menor valor e quanto foi, então faça um programa utilizando lista para resolver esse problema.

5. Uma biblioteca deseja fazer um sistema para alugar livro onde recebe o nome da pessoa que vai alugar, a quantidade de dias e o nome do livro. Por fim, exiba todas as informações. Obs: Utilize dicionário para guardar as informações.

6. Faça um sistema de cadastro de entrada dos pacientes no hospital utilizando o dicionário e guarde as informações do nome e se o estado do paciente é Leve, Grave ou Crítico. Se o paciente estiver em estado leve, receberá a pulseira verde; se grave, a amarela; e se crítico, vermelha. Por fim, exiba todos os pacientes cadastrados e também exiba os paciente de maneira separada pela sua pulseira.

APÊNDICE D – *SCRATCH* – PRIMEIRO *QUIZ*

Q1. Para que servem os operadores aritméticos?

▲ São utilizados para obter resultados numéricos. ✓

♦ São utilizados para obter resultados booleanos. ✗

● São utilizados para obter resultados do tipo *string*. ✗

■ São utilizados para obter resultados de imagens. ✗

Q2. Qual o objetivo dos operadores lógicos?

▲ Servem para combinar resultados de expressões, retornando *true* ou *false*. ✓

♦ São utilizados para comparar *string* de caracteres e números. ✗

● Servem para combinar resultados de expressões, retornando um número. ✗

■ São os utilizados para obter resultados numéricos. ✗

Q3. O que é sequência lógica?

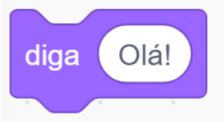
▲ São passos executados até atingir um objetivo ou solução de um problema. ✓

♦ São passos executados infinitamente para solucionar um problema. ✗

● São passos ambíguos até atingir um objetivo ou solução de um problema. ✗

■ São os utilizados para obter resultados numéricos. ✗

Q4. Qual a função do bloco abaixo?



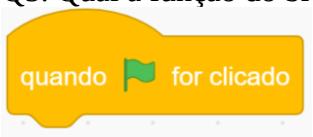
▲ Mostra algo específico em um balão de conversa. ✓

♦ Mostra o escrito, em um balão de pensar, pelo tempo configurado. ✗

● Mostra o especificado, em um balão de conversa, pelo tempo específico. ✗

■ Mostra o especificado em um balão de pensamento. ✗

Q5. Qual a função do bloco abaixo?



▲ Roda o *script* abaixo quando a bandeira verde for clicada. ✓

♦ Roda o *script* abaixo quando o ator especificado for clicado no palco. ✗

● Executa o *script* abaixo quando a tecla especificada for pressionada. ✗

■ Aguarda o tempo especificado e passa para o bloco seguinte. ✗

Q6. Qual a função do bloco abaixo?

▲ Repete a execução do bloco interno até que a condição esteja satisfeita. ✓

♦ Executará o *script* se a condição colocada estiver satisfeita. ✗

● Espera até que a condição seja verdadeira para executar o bloco seguinte. ✗

■ Para a execução do *script* em que estiver inserido. ✗

Q7. Qual a função do bloco abaixo?

▲ Executa o "se" caso a condição dada for satisfeita; em caso contrário, "senão". ✓

♦ Espera até que a condição seja verdadeira para executar o bloco seguinte. ✗

● Espera até que a condição seja verdadeira para executar o bloco seguinte. ✗

■ Para a execução do *script* em que estiver inserido. ✗

Q8. Qual a função do bloco abaixo?

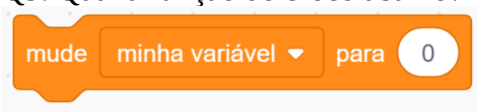
▲ Devolve o resto da divisão do 1º pelo 2º número. ✓

♦ Espera até que a condição seja verdadeira para executar o bloco seguinte. ✗

● Devolve o resto da divisão do 2º pelo 1º número. ✗

■ Para a execução do *script* em que estiver inserido.✗

Q9. Qual a função do bloco abaixo?



▲ Muda o valor da variável indicada para o valor especificado. ✓

♦ Espera até que a condição seja verdadeira para executar o bloco seguinte. ✗

● Mostra, no Palco, o monitor da variável indicada. ✗

■ Retira, do Palco, o monitor da variável indicada. ✗

Q10. Qual a função do bloco abaixo?

▲ Verifica se pelo menos uma das condições está satisfeita. ✓

♦ Espera até que a condição seja verdadeira para executar o bloco seguinte. ✗

● Verifica se as 2 condições dadas estão satisfeitas. ✗

■ Retira, do Palco, o monitor da variável indicada. ✗

Q11. Qual a função do bloco abaixo?

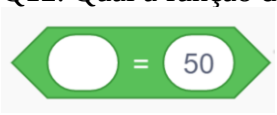
▲ Verifica se pelo menos uma das condições está satisfeita. ✗

♦ Espera até que a condição seja verdadeira para executar o bloco seguinte. ✗

● Verifica se as 2 condições dadas estão satisfeitas. ✓

■ Concatena 2 elementos no formato frase. ✗

Q12. Qual a função do bloco abaixo?



▲ Verifica se pelo menos uma das condições está satisfeita. ✗

♦ Verifica se o 1º nº é igual ao 2º. ✓

● Verifica se as 2 condições dadas estão satisfeitas. ✗

■ Verifica se o 2º nº é igual ao 1º. ✗

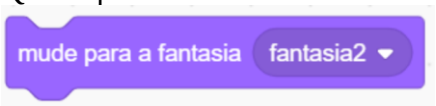
APÊNDICE E – *SCRATCH* – SEGUNDO *QUIZ*

Q1. Para que servem os operadores aritméticos?
▲ São utilizados para obter resultados numéricos. ✓
♦ São utilizados para obter resultados booleanos. ✗
● São utilizados para obter resultados do tipo *string*. ✗
■ São utilizados para obter resultados de imagens. ✗

Q2. Qual o objetivo dos operadores lógicos?
▲ Servem para combinar resultados de expressões, retornando *true* ou *false*. ✓
♦ São utilizados para comparar *string* de caracteres e números. ✗
● Servem para combinar resultados de expressões, retornando um número. ✗
■ São utilizados para obter resultados numéricos. ✗

Q3. O que é sequência lógica?
▲ São passos executados até atingir um objetivo ou solução de um problema. ✓
♦ São passos executados infinitamente para solucionar um problema. ✗
● São passos ambíguos até atingir um objetivo ou solução de um problema. ✗
■ São utilizados para obter resultados numéricos. ✗

Q4. O que faz o bloco abaixo?

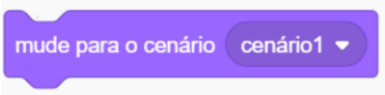


▲ Muda a aparência do ator assumindo o traje especificado. ✓
♦ Muda para o traje seguinte da lista de trajes. ✗
● Mostra o valor do traje em uso. ✗
■ Mostra o especificado, em um balão de conversa, pelo tempo. ✗

Q5. O que faz o bloco abaixo?

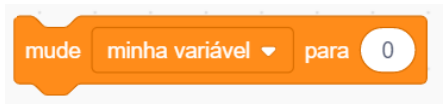
▲ Faz o ator aparecer no palco. ✗
♦ Muda para o traje seguinte da lista de trajes. ✗
● Faz o ator desaparecer. ✓
■ Mostra o especificado, em um balão de conversa, pelo tempo. ✗

Q6. O que faz o bloco abaixo?



▲ Muda para o cenário seguinte. ✗
♦ Muda para o traje seguinte da lista de trajes. ✗
● Muda o *background* cujo nome for especificado. ✓
■ Mostra o especificado, em um balão de conversa, pelo tempo. ✗

Q7. Escolha a opção que “Executa os blocos envolvidos um determinado número de vezes”:

▲ 

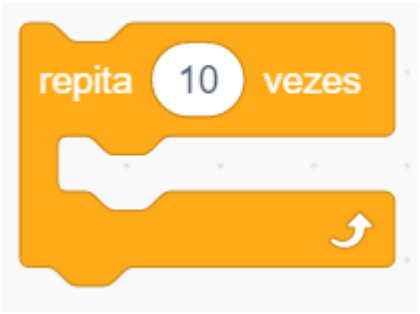
✗

♦

✗

●

✗

■ 

✓

Q8. Escolha a opção que “Se a condição for VERDADEIRA, executa os blocos envolvidos”:

▲

✗

♦

✗

●

✓

■

✗

Q9. Tem o valor verdadeiro se ambas as condições forem verdadeiras:

▲

✓

♦

✗

●

✗

■

✗

Q10. Tem o valor verdadeiro se alguma das condições for verdadeira:

♦ True ✓
▲ False ✗

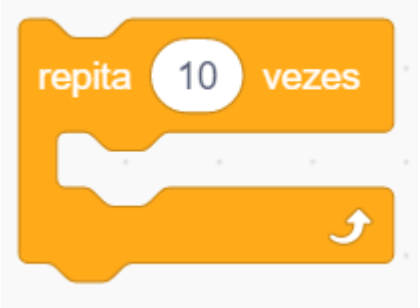
Q11. Concatena (combina, junta) cadeias de caracteres:

♦ True ✓
▲ False ✗

Q12. Executa os blocos envolvidos um determinado número de vezes:

♦ True ✗
▲ False ✓

Q13. Repete os blocos que se seguem até que a condição seja VERDADEIRA:



♦ True ✗
▲ False ✓

APÊNDICE F – *SCRATCH* – TERCEIRO *QUIZ*

Q1. Qual a função do bloco “Mais Blocos” no *scratch*?

▲ Permite criar uma rotina ou um comando não contido nas categorias. ✓

♦ Permite utilizar os resultados booleanos para criar extensão. ✗

● Permite criar uma rotina ou um comando no comando “se”. ✗

■ São utilizados para obter resultados de imagens. ✗

Q2. A imagem abaixo é da opção “mais blocos”. Qual entrada que ela permite?

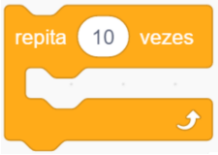
▲ Número ou texto. ✓

♦ Imagem e texto.✗

● Imagem. ✗

■ Número e imagem. ✗

Q3. Executa os blocos envolvidos um determinado número de vezes:



♦ True ✓

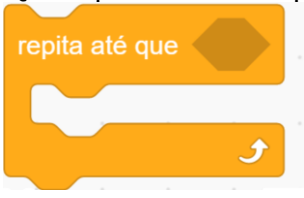
▲ False ✗

Q4. Executa repetidamente e para sempre os comandos no seu interior:

♦ True ✗

▲ False ✓

Q5. Repete os blocos que se seguem até que a condição seja verdadeira:



♦ True ✓

▲ False ✗

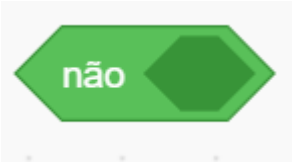
Q6. Tem o valor verdadeiro se alguma das condições for verdadeira:

♦ True ✓
▲ False ✗

Q7. Tem o valor verdadeiro se ambas as condições forem verdadeiras:

♦ True ✓
▲ False ✗

Q8. Tem o valor verdadeiro se a condição for falsa e tem o valor falso se a condição for verdadeira:

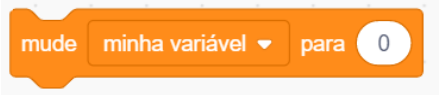


♦ True ✓
▲ False ✗

Q9. Concatena (combina, junta) cadeias de caracteres:

♦ True ✗
▲ False ✓

Q10. Executa um *script* quando o atributo selecionado ultrapassar um valor especificado:



♦ True ✗
▲ False ✓

APÊNDICE G – PORTUGOL – PRIMEIRO *QUIZ*

Q1. Quais os tipos de dados suportados pelo *Visualg*?

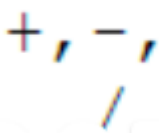
▲ Inteiro, real, caractere e lógico. ✓

♦ Double, *string*, *float* e *int*. ✗

● *Object*, inteiro, real e lógico. ✗

■ Interface, *class*, lógico e inteiro. ✗

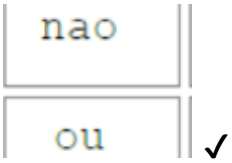
Q2. Qual imagem a seguir apresenta alguns operadores lógicos suportados no *Visualg*?

▲ 

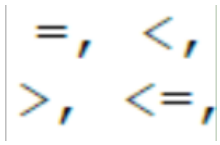
✗

♦

✗

● 

✓

■ 

✗

Q3. Marque a alternativa que apresenta de forma correta a declaração de uma variável no *Visualg*.

▲ <lista-de-variáveis> : <tipo-de-dado> ✓

♦ <lista-de-variáveis> :- <tipo-de-dado> ✗

● <lista-de-variáveis> = <tipo-de-dado> ✗

■ <lista-de-variáveis> <- <tipo-de-dado> ✗

Q4. Marque a alternativa que apresenta de forma correta a atribuição de um valor de uma variável no *Visualg*.

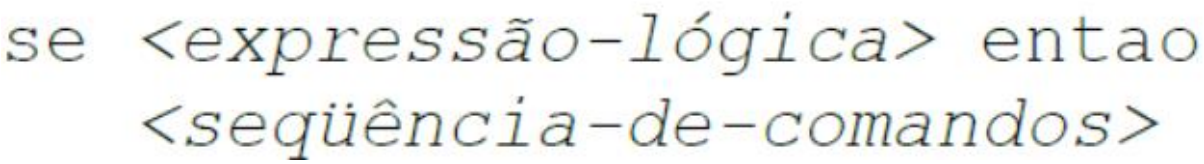
▲ a <- 3 ✓

♦ a = 3 ✗

● a== 3 ✗

■ a :<-3 ✗

Q5. A seguir é apresentado o comando desvio condicional no *Visualg*. Analise a imagem e assinale a alternativa correta.



▲ A sintaxe está incorreta, pois está faltando o senão ao final. ✗

♦ A sintaxe está incorreta, pois está faltando um fimse ao final. ✓

● A sintaxe está correta e não vai dar erro. ✗

■ A sintaxe está incorreta, pois está faltando as { }. ✗

Q6. Qual é a saída do código abaixo?

```
Var

   x: inteiro

Inicio

   x <- 0
   repita
      x <- x + 1
      escreva (x:3)
      se x = 10 entao
         interrompa
      fimse
   ate falso


Fimalgoritmo
```

▲ 1 2 3 4 5 6 7 8 9 10 ✓

♦ 0 1 2 3 4 5 6 7 8 9 ✗

● 1 2 3 4 5 6 7 8 9 ✗

■ 0 1 2 3 4 5 6 7 8 9 10 ✗

Q7. Qual a saída do código da imagem?

```
Var

   j: inteiro

Inicio

   para j de 10 ate 1 passo -1 faca
      escreva (j:3)
   fimpara

Fimalgoritmo
```

▲ 10 9 8 7 6 5 4 3 2 1 ✓

♦ 9 8 7 6 5 4 3 2 1 0 ✗

● 9 8 7 6 5 4 3 2 1 ✗

■ 10 9 8 7 6 5 4 3 2 1 0 ✗

Q8. O código a seguir vai compilar.

```
Var

   numero : inteiro
Inicio

   numero <-100
   para numero de 1 ate 100  faca
      se (numero%10=0)         entao
         escreval ("Múltiplo de 10", numero)

      fimpara

Fimalgoritmo
```

♦ True ✗

▲ False ✓

Q9. (anulada) Questão incompleta.

Q10. Qual a saída deste código assumindo que o valor de n=10?

```
Var

n,j: inteiro

Inicio

j<-0
escreval("Escreva um número")
leia (n)
para j de 0 ate n  faca
escreva (j:3)
fimpara
Fimalgoritmo
```

▲ 0 1 2 3 4 5 6 7 8 9 10 ✓

♦ 1 2 3 4 5 6 7 8 9 ✗

● 1 2 3 4 5 6 7 8 9 10 ✗

■ 0 1 2 3 4 5 6 7 8 9 ✗

Q11.Qual é a saída do código abaixo?

```
Var

   numero,j : inteiro

Inicio

   numero <-5
   para j de 0 ate numero  faca
      se (j%2=0)          entao
         escreval (j:4, " é par")
      fimse

   fimpara
Fimalgoritmo
```

▲ 0 é par 2 é par 4 é par ✓

♦ O código não vai compilar, pois não foi atribuído um valor para j ✗

● 2 é par 4 é par ✗

■ O código não vai compilar, pois o para no *Visualg* não pode começar com 0 ✗

APÊNDICE H – PORTUGOL – SEGUNDO *QUIZ*

Q1. Qual a saída do código abaixo assumindo que o valor das posições: [1] = 5, [2] = 15, [3]=1

```
Algoritmo "semnome"

Var
   SOMA, i : inteiro
   MEDIA : real
   VET : vetor [1..3] de inteiro

Inicio

   para i de 1 até 3 faca
      escreva( "Entre com o valor do índice ", i, " : ")
      leia( VET[i] )
      SOMA <- SOMA + VET[i]
   fimpara
   MEDIA <- SOMA/3
   escreval( "Média: ", MEDIA )
Fimalgoritmo
```

▲ 7 ✓

♦ 9 ✗

● 8 ✗

■ 3 ✗

Q2. Supondo que o elemento nas posições 0 até 4 é 2. Qual a saída do algoritmo?

```
Var
   // Seção de Declarações das variáveis
   numeros : vetor [0..4] de inteiro
   cont, soma : inteiro

funcao somatorio :inteiro
inicio
   para cont de 0 ate 4 faca
      escreval("Entre com um número:")
      leia(numeros[cont])
      soma <- soma + numeros[cont])
   fimpara
   retorne soma
fimfuncao

Inicio
   escreval("A soma dos números é", somatorio)
   escreval()
Fimalgoritmo
```

▲ 10 ✓

♦ 8 ✗

● 2 ✗

■ 12 ✗

Q3. Esse algoritmo possui três “para”. De cima para baixo qual a função do primeiro para?

```
Var
nomes : vetor [0..4] de caractere
cont : inteiro

Inicio
para cont de 0 ate 4 faca
   escreval("Entre com um nome:")
   leia(nomes[cont])

fimpara
para cont de 0 ate 4 faca
   escreval(nomes[cont])
fimpara
escreval()


para cont de 4 ate 0 passo - 1 faca
   escreval(nomes[cont])
fimpara
Fimalgoritmo
```

▲ Guardar no vetor nomes o que for digitado pelo usuário. ✓

♦ Contar os nomes digitados pelo usuário. ✗

● Guardar no vetor cont o que for digitado pelo usuário. ✗

■ Guardar no índice nomes o que for digitado pelo usuário. ✗

Q4. Esse algoritmo possui três “para”. De cima para baixo qual a função do segundo para?

```
Var
nomes : vetor [0..4] de caractere
cont : inteiro

Inicio
para cont de 0 ate 4 faca
   escreval("Entre com um nome:")
   leia(nomes[cont])

fimpara
para cont de 0 ate 4 faca
   escreval(nomes[cont])
fimpara
escreval()


para cont de 4 ate 0 passo - 1 faca
   escreval(nomes[cont])
fimpara
Fimalgoritmo
```

▲ Escrever na tela todos os nomes digitados que estão no vetor nomes. ✓

♦ Escrever na tela todos os nomes digitados que estão no índice nomes. ✗

● Escrever na tela todos os nomes digitados que estão no vetor cont. ✗

■ Guardar no índice nomes o que for digitado pelo usuário. ✗

Q5. Esse algoritmo possui três “para”. De cima para baixo qual a função do terceiro para?

```
Var
nomes : vetor [0..4] de caractere
cont : inteiro

Inicio
para cont de 0 ate 4 faca
   escreval("Entre com um nome:")
   leia(nomes[cont])

fimpara
para cont de 0 ate 4 faca
   escreval(nomes[cont])
fimpara
escreval()


para cont de 4 ate 0 passo - 1 faca
   escreval(nomes[cont])
fimpara
Fimalgoritmo
```

▲ Escrever na tela todos os nomes digitados em ordem inversa. ✓

♦ Contar os nomes digitados pelo usuário. ✗

● Escrever na tela todos os nomes digitados na ordem que foram digitados. ✗

■ Atribuir ao vetor nomes o que foi digitado pelo usuário. ✗

APÊNDICE I – PORTUGOL – TERCEIRO *QUIZ*

Q1. A seguir são apresentadas algumas variáveis. Qual o valor de r e s, respectivamente?

```
r<-2

s<-4

r <-s
```

▲ 4 e 4 ✓

♦ 2 e 2 ✗

● 2 e 4 ✗

■ 4 e 2 ✗

Q2. A seguir são apresentadas algumas variáveis. Qual o valor de a, b e c, respectivamente?

```
a<-5

b<-3

c <-7

a<-c

b<-a

c<-b
```

▲ 7, 7 e 7 ✓

♦ 7, 3 e 3 ✗

● 5, 3 e 7 ✗

■ 7, 5, 3 ✗

Q3. Verifique o código abaixo e assinale o valor de s ao final do algoritmo.

```
Var
   p,q,s: inteiro

Inicio
   p<-1
   q<-2
   s<-4
   se (p<q) entao

      se (q>4) entao
         s<-5
      senao
         s<-6
      fimse
   fimse
   escreval(s)

Fimalgoritmo
```

▲ 6 ✓

♦ 4 ✗

● 5 ✗

■ 2 ✗

Q4. Qual a saída do algoritmo abaixo?

```
Var
   vetor_: vetor  [0..2] de inteiro
   aux:inteiro
Inicio
   vetor_[0]<- 2
   vetor_[1]<- 3
   vetor_[2]<- 1
   se (ve
      tor_[0] > vetor_[1])      entao
      aux <- vetor_[0]
      vetor_[0] <- vetor_[1]
      vetor_[1] <- aux
   fimse
   se (vetor_[1] > vetor_[2])      entao
      aux <- vetor_[1]
      vetor_[1] <- vetor_[2]
      vetor_[2] <- aux
   fimse
   se (vetor_[0] > vetor_[1])      entao
      aux <- vetor_[0]
      vetor_[0] <- vetor_[1]
      vetor_[1] <- aux
   fimse
   escreval(vetor_[0], vetor_[1], vetor_[2])
Fimalgoritmo
```

▲ 1 2 3 ✓

♦ 3 2 1 ✗

● 2 3 1 ✗

■ 3 1 2 ✗

Q5. Qual o valor de *result* no final do algoritmo?

```
Var
   vetorI: vetor  [1..5] de inteiro
   i,result:inteiro
Inicio
   result <-0
   vetorI[1]<- 1
   vetorI[2]<- 3
   vetorI[3]<- 2
   vetorI [4]<- 5
   vetorI [5]<- 4

   para   i de 1 ate 5 faca

      se (i+1 < 5) entao
         se (vetorI[i] < vetorI[i+1])entao
            result<-result+ vetorI[i]
         fimse
      fimse

   fimpara
   escreval(result)
Fimalgoritmo
```

▲ 3 ✓

♦ 4 ✗

● 5 ✗

■ 2 ✗

Q6. Qual o valor de result no final do algoritmo?

```
Var
   vetorI: vetor  [1..3] de inteiro
   i,result:inteiro
Inicio
   result <-0
   vetorI[1]<- 4
   vetorI[2]<- 7
   vetorI[3]<- 3
   result<-vetorI[1]
   result<- result+ vetorI[2]
   result<- result+ vetorI[3]

   escreval(result)
Fimalgoritmo
```

▲ 14 ✓

♦ 4 ✗

● 10 ✗

■ 11 ✗

Q7. Observe o código abaixo e responda. Qual o valor de vetorP[aux] quando o índice for 1 e 3 respectivamente?

```
   // Seção de Declarações das variáveis
   vetorP: vetor [1..5] de inteiro
   n,aux,numero:inteiro


Inicio
   // Seção de Comandos, procedimento, funções, operadores, etc..
   aux<-1
   numero<-60

   para n  de 1 ate numero faca
      numero<-numero+1
      se (n%3=0) entao
         se (aux <=5) entao
            vetorP[aux]<-n
            escreval("Múltiplos de 3: ",aux,vetorP[aux])
            aux<-aux+1
         fimse
      fimse
   fimpara
Fimalgoritmo
```

▲ 3 e 9 ✓

♦ 6 e 9 ✗

● 3 e 6 ✗

■ 9 e 12 ✗

APÊNDICE J – *PYTHON* – PRIMEIRO *QUIZ*

Q1. Qual característica abaixo está relacionada à Estrutura de Dados Tupla?

▲ Tupla é uma Lista imutável. ✓

♦ Tupla não tem index. ✗

● Tupla é uma Lista mutável. ✗

■ Tupla possui chave/valor. ✗

Q2. A imagem a seguir é de?

```
#coding: utf-8

nums = [1, 2, 3, 4, 5]
```

▲ Uma lista. ✓

♦ Uma tupla composta. ✗

● Uma tupla. ✗

■ Um dicionário. ✗

Q3. Uma lista é representada como uma sequência de objetos separados por vírgula e dentro de colchetes:

♦ True ✓

▲ False ✗

Q4. Qual a saída do código a seguir?

```
numeros = [12, 14, 15, 16, 17, 18, 19]
print(numeros[2])
print(numeros[7-6])
print(numeros[-1])
print(len(numeros))
```

▲ 15 14 19 7 ✓

♦ 14 14 19 12 ✗

● 15 19 19 7 ✗

■ 15 14 19 6 ✗

Q5. Qual a saída do código da imagem?

```
numeros = [1, 2, 3, 4, 5, 6, 7, 8]
print(numeros[::-1])
```

▲ [8, 7, 6, 5, 4, 3, 2, 1] ✓

♦ [8] ✗

● [1, 2, 3, 4, 5, 6, 7, 8] ✗

■ [1] ✗

Q6. Qual a saída do código abaixo?

```
estados = ['PE']+['RN']
print(len(estados))
estados += ['PI']
print(estados)
```

▲ 2['PE', 'RN', 'PI'] ✓

♦ 3 ['PE', 'RN', 'PI'] ✗

● 3 ['PE', 'RN'] ✗

■ 2 ['PE', 'RN'] ✗

Q7. Qual a saída do código abaixo?

```
def posicoes(frase, caractere):
    lista = []
    for i in range(0, len(frase)):
        if frase[i] == caractere:
            lista.append(i)
    return lista
print(posicoes('reviver', 'e'))
```

▲ [1, 5] ✓

♦ [2, 6] ✗

● [1, 4] ✗

■ [2,5] ✗

Q8. Observe o código abaixo e assinale a alternativa correta.

```
tupla = ('1', '2', '4', '4')
tupla[2] = 3
print(tupla)
```

▲ O código não compila, pois não é possível mudar um valor em uma tupla. ✓

♦ Todos os elementos de uma tupla precisam ser homogêneos. ✗

● O código não compila, pois o valor a ser adicionado é de outro tipo. ✗

■ O código compila sem erro. ✗

Q9. Observe o código abaixo e assinale a alternativa correta.

```
listaOrdenada = []
lista = ['2', 1, '4', 3, '6', 5]
for l in range(0, len(lista)):
    listaOrdenada.append(int(lista[l]))
print(sorted(listaOrdenada))
```

▲ O código não compila pois são tipos de dados diferentes (*string* e int). ✗

♦ O código funciona e ao final exibe os seguintes valores: [1, 2, 3, 4, 5, 6]. ✓

● O código não compila, pois em *python* só há listas do mesmo tipo. ✗

■ O código funciona e ao final exibe: [2, 1, 4, 3, 6, 5] ✗

Q10. Qual a saída do código abaixo?

```
dicionario = {'Joao': 20, 'Maria': 22, 'Leticia': 25}
dicionario.pop('Joao')
print(dicionario)
```

▲ {'Maria': 22, 'Leticia': 25} ✓

♦ {'Joao': 20} ✗

● {'Joao': 20, 'Maria': 22, 'Leticia': 25} ✗

■ {'Joao'} ✗

APÊNDICE K – *PYTHON* – SEGUNDO *QUIZ*

Q1. Qual a saída do código (MENA-CHALCO, 2013)?

```
aluno = "Joao Carlo"
disciplina = "PI"
if aluno=="Joao Carlos" or disciplina=="PI":
    print "Conceito A"
else:
    print "Aluno nao cadastrado"
```

▲ Conceito A ✓

♦ O código não compila ✗

● Aluno não Cadastrado ✗

■ O *if* está incorreto ✗

Q2. Qual a saída do código abaixo (MENA-CHALCO,2013)?

```
x=8
y=5
z=13
if x>=1 and x<=31:
    if y>0 and y<13:
        if x+y!=z:
            print "A data de hoje eh 8/5/13"
        else:
            print "A data de hoje nao eh 8/5/13"
```

▲ A data de hoje nao é 8/5/136. ✓

♦ Código não compila. ✗

● A data de hoje eh 8/5/13 6. ✗

■ Erro no if ✗

Q3. Simule a execução da função abaixo e indique (algebricamente) a sua saída (MENA-CHALCO, 2013).

```
def funcaoEnigma1(a,b,p,q):
    a = float(a)
    b = float(b)
    p = int(p)
    q = int(q)
    i = 0
    soma = 0
    while i<=(q-p):
        soma = (a/b)**(i)
        i = i+1
    return soma
```

▲ $(\frac{a}{b})^{q-p}$ ✓

♦ $(\frac{a}{b})^{p-q}$ ✗

● $(\frac{b}{a})^{q-p}$ ✗

■ $(\frac{b}{a})^{p-q}$ ✗

Q4. Qual a saída do código abaixo? Considere o valor de n = 2 (MENA-CHALCO, 2013).

```
def funcaoEnigma3(n):
    soma = 0
    for i in range(1,n+1):
        soma = soma+i
    j=1
    while j<=n:
        soma = soma-j
        j = j+1
    return soma
```

▲0 ✓

♦ 2 ✗

● 1 ✗

■ 3 ✗

Q5. Qual a saída do código abaixo? Considere valores de n > 0 (MENA-CHALCO, 2013).

```
def funcaoEnigma4(n):
    mult = 1.0
    i = 0
    while i<=n-2:
        mult = mult*((i+1.0)/(i+2.0))
        i = i+1
    return mult*n
```

▲ 1.0 ✓

♦ 0.0 ✗

● 2.0 ✗

■ 3.0 ✗

Q6. Qual a saída do código? Considere como parâmetro de entrada a *string* ‘abracadabra’ (MENA-CHALCO, 2013).

```
def funcao1(frase):
    str1 = ""
    str2 = ""
    k = len(frase)-1
    while k>=0:
        str1 = str1 + frase[k]
        str2 = frase[k] + str2
        k = k-1
    print str1
    print str2
```

▲ arbadacarba abracadabra ✓

♦ abracadabra abracadabra ✗

● abracadabra arbadacarba ✗

■ arbadacarba arbadacarba ✗

Q7. Qual a saída do código abaixo? Considere como parâmetro de entrada as *strings* ‘UFABC’ e ‘123’ (MENA-CHALCO, 2013).

```
def funcao2(palavra):
    str = ''
    k = 0
    while k<len(palavra):
        str = str + palavra[k]
        k = k+1
    while k>0:
        k = k-1
        str = str + palavra[k]
    print str
```

▲ UFABCCBAFU 123321 ✓

♦ UFABC 123 ✗

● 123321 UFABCCBAFU ✗

■ 123 UFABC ✗

Q8. Qual a saída do código abaixo? Considere como parâmetro de entrada a *string* ‘um dois tres’ (MENA-CHALCO, 2013).

```
def funcao3(frase):
    contador = 0
    k = 0
    while k<len(frase)/2:
        if frase[k]==" ":
            contador = contador+1
    print contador
```

▲ 0 infinitamente ✓

♦ ‘um dois tres’ infinitamente ✗

● 0 ✗

■ ‘um doi tres’ ✗

Q9. Qual a saída do código abaixo? Considere como parâmetro de entrada a *string* ‘54321’ (MENA-CHALCO, 2013).

```
def funcao4(frase):
    nova_frase = ''
    k = 0
    while k<len(frase):
        nova_frase = frase[k] + nova_frase + frase[k]
        k = k+1
    print nova_frase
```

▲ 1234554321 ✓

♦ 1234512345 ✗

● 5432112345 ✗

■ 5432154321 ✗

Q10. Qual a saída do código abaixo? Considere como parâmetro de entrada a lista [1,2,4,16,32,64,-128] (MENA-CHALCO, 2013).

```
def funcao1(lista):
    temp1 = lista[0]
    temp2 = lista[len(lista)-1]
    for elemento in lista:
        if temp1>elemento:
            temp1 = elemento
        if temp2<elemento:
            temp2 = elemento
    print str(temp1) + ' ' + str(temp2)
```

▲ -128 64 ✓

♦ 32 64 ✗

● 64 -128 ✗

■ 64 32 ✗

REFERÊNCIAS

MENA-CHALCO, J. P. **Caderno de Exercícios Processamento da Informação – Edição Python.** Universidade Federal do ABC, 2013. Disponível em: http://professor.ufabc.edu.br/~jesus.mena/. Acesso em: 15 nov. 2020.

APÊNDICE L – QUESTÕES DE AUTOEFICÁCIA

1. Eu posso entender a estrutura dos blocos do *Scratch*. 1. Eu posso escrever códigos em *Visualg*/*Python* sintaticamente corretos.
2. Eu posso escrever código logicamente corretos no *Scratch/Visualg/Python*.
3. Eu posso escrever um programa no *Scratch/Visualg/Python* que exibe uma mensagem de saudação.
4. Eu posso escrever um programa no *Scratch/Visualg/Python* que calcula a média de três números.
5. Eu posso escrever um programa no *Scratch/Visualg/Python* que calcula a média de qualquer número.
6. Eu posso escrever um programa no *Scratch/Visualg/Python* para resolver um problema que me é familiar.
7. Eu posso depurar (corrigir todos os erros) de um programa que eu havia desenvolvido e fazê-lo funcionar.
8. Eu posso concluir um projeto de programação se alguém me mostrasse como resolver o problema primeiro.
9. Eu posso concluir um projeto de programação se tivesse apenas o manual do *Scratch/Visualg/ Python* para obter ajuda.
10. Eu posso concluir um projeto de programação no *Scratch* assim que outra pessoa me ajudasse a começar.

Fonte: Adaptado de Askar e Davenport (2009).