UNIVERSIDADE ESTADUAL DE PONTA GROSSA
PRÓ-REITORIA DE PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO ENSINO DE CIÊNCIAS E EDUCAÇÃO MATEMÁTICA

EMERSON BLUM CORRÊA

O DESENVOLVIMENTO DO PENSAMENTO COMPUTACIONAL E ALGÉBRICO NA FORMAÇÃO INICIAL DE PROFESSORES DE MATEMÁTICA: UM ESTUDO DE CASO COM *SCRATCH*

PONTA GROSSA
2020

EMERSON BLUM CORRÊA

O DESENVOLVIMENTO DO PENSAMENTO COMPUTACIONAL E ALGÉBRICO NA FORMAÇÃO INICIAL DE PROFESSORES DE MATEMÁTICA: UM ESTUDO DE CASO COM *SCRATCH*

Dissertação apresentada como requisito para a obtenção do título de Mestre em Ensino de Ciências e Educação Matemática no Programa de Pós-Graduação em Ensino de Ciências e Educação Matemática na Universidade Estadual de Ponta Grossa.

Orientadora:
Prof.ª Dra. Luciane Grossi

PONTA GROSSA
2020

FICHA CATALOGRÁFICA

Corrêa, Emerson Blum

C824 O desenvolvimento do pensamento computacional e algébrico na formação inicial de professores de matemática: um estudo de caso com Scratch / Emerson Blum Corrêa. Ponta Grossa, 2020.

235 f.

Dissertação (Mestrado em Ensino de Ciências e Educação Matemática - Área de Concentração: Espaços Formais e Não Formais no Ensino de Ciências), Universidade Estadual de Ponta Grossa.

Orientadora: Profa. Dra. Luciane Grossi.

1. Formação de professores. 2. Educação matemática. 3. Pensamento computacional. 4. Pensamento algébrico. 5. Tecnologias educacionais. I. Grossi, Luciane. II. Universidade Estadual de Ponta Grossa. Espaços Formais e Não Formais no Ensino de Ciências. III.T.

CDD: 510.7

Ficha catalográfica elaborada por Maria Luzia Fernandes Bertholino dos Santos- CRB9/986

TERMO

TERMO DE APROVAÇÃO

EMERSON BLUM CORRÊA

" O DESENVOLVIMENTO DO PENSAMENTO COMPUTACIONAL E ALGÉBRICO NA FORMAÇÃO INICIAL DE PROFESSORES DE MATEMÁTICA: UM ESTUDO DE CASO COM SCRATCH."

Dissertação aprovada como requisito parcial para obtenção do grau de Mestre no Programa de Pós Graduação em Ensino de Ciências e Educação Matemática, Setor de Ciências Exatas e Naturais da Universidade Estadual de Ponta Grossa, pela seguinte banca examinadora:

Ponta Grossa 14 de Setembro de 2020.

Membros da Banca:

Dra. Luciane Grossi - (UEPG) – Presidente

Dr. Thiago Schumacher Barcelos - (IFSP)

Dra. Célia Finck Brandt - (UEPG)

Documento assinado eletronicamente por **Adriana Aparecida Telles**, **Secretário(a)**, em 08/09/2020, às 17:50, conforme art. 1º, III, "b", da Lei 11.419/2006.

Documento assinado eletronicamente por **Thiago Schumacher Barcelos**, **Usuário Externo**, em 21/09/2020, às 16:45, conforme art. 1º, III, "b", da Lei 11.419/2006.

Documento assinado eletronicamente por **Luciane Grossi**, **Coordenador(a) do Programa de Pós-Graduação em Ensino de Ciências e Educação Matemática**, em 24/09/2020, às 13:10, conforme art. 1º, III, "b", da Lei 11.419/2006.

Documento assinado eletronicamente por **Celia Finck Brandt**, **Professor(a)**, em 24/09/2020, às 14:58, conforme art. 1º, III, "b", da Lei 11.419/2006.

A autenticidade do documento pode ser conferida no site https://sei.uepg.br/autenticidade informando o código verificador **0289254** e o código CRC **85D378F9**.

Dedico esta dissertação à minha mãe Maria Paula Blum que, apesar de todos os desafios e dificuldades, não poupou esforços para me ajudar a concretizar essa pesquisa.

AGRADECIMENTOS

Agradeço à minha orientadora Prof.ª Dra. Luciane Grossi por sua disponibilidade, ensinamentos, incentivo e apoio extraordinários que foram fundamentais para a realização e conclusão deste estudo. A agradeço também pelas conversas, críticas e reflexões que contribuíram para amadurecer essa pesquisa, e para além disso, para me amadurecer como investigador, professor e pessoa. Eternamente grato por todo o apoio.

À todos os professores do Programa de Pós-Graduação em Ensino de Ciências e Educação Matemática (PPGECEM), da Universidade Estadual de Ponta Grossa, pelos ensinamentos e pelos esforços que possibilitaram a existências dessa pós-graduação.

À todos os membros do Grupo de Pesquisa de Tecnologias Educacionais em Matemática (GTEM), aos docentes e discentes do PPGECEM e ao Gustavo Menim Cruz por suas contribuições nas diversas etapas dessa pesquisa.

À banca examinadora desta pesquisa, Prof.ª Dra. Célia Finck Brandt e ao Prof. Dr. Thiago Schumacher Barcelos, pelas considerações que me auxiliaram a nortear, enriquecer e concluir este trabalho.

À Universidade Estadual de Ponta Grossa por possibilitar que eu concluísse tanto minha formação inicial em licenciatura em Matemática, quanto minha formação nesta pós-graduação.

Aos meus familiares e amigos por me ajudarem financeiramente, psicologicamente e emocionalmente em todos os momentos que precisei.

RESUMO

Atualmente há uma significativa interdependência entre espaços físicos e digitais que tende a se ampliar ao longo do tempo. Essa relação tem alterado a forma com que vivemos, pensamos e agimos dentro de nossa sociedade e, a escola não está alheia a essas mudanças. É necessário discutir como integrar as tecnologias digitais ao cotidiano escolar de forma a preparar os estudantes para viver nessa sociedade altamente dependente do ciberespaço. Para que essa integração ocorra é necessário atentar-se à formação de docentes. Em particular, voltamos nossa atenção aos profissionais em formação no curso de Licenciatura em Matemática. Diante desse contexto, este estudo buscou responder a seguinte pergunta: Quais aspectos do Pensamento Computacional e Pensamento Algébrico são evidenciados por licenciandos em Matemática na realização de atividades com o Scratch? Para isso realizamos um estudo de caso exploratório de abordagem mista. Os sujeitos dessa pesquisa foram 14 acadêmicos dos quartos anos, integral e noturno, do curso de Licenciatura em Matemática de uma instituição de Ensino Superior do estado do Paraná. Os dados empíricos foram coletados no decorrer de uma sequência didática elaborada pelos autores e validada pelo Grupo de Pesquisa de Tecnologias Educacionais em Matemática (GTEM). Os instrumentos de coleta de dados utilizados foram: questionário pré-implementação, algoritmos de construção de um quadrado e de um triângulo em linguagem materna e no Scratch. Os dados qualitativos foram analisados segundo a Análise de Conteúdo de Bardin, enquanto os quantitativos foram analisados por meio de porcentagens simples geradas a partir das frequências das categorias. Com base em Radford, Brennan e Resnick delineamos dois aspectos gerais nos algoritmos: algébricos e computacionais. Os aspectos algébricos subdividem-se em objetificação, simbolização e generalização, enquanto os computacionais subdividem-se em estrutura e depuração. Esses aspectos foram analisados a partir de indicadores produzidos no decorrer desta pesquisa. Dentre os resultados obtidos destacamos que: uma significativa parcela dos sujeitos afirmou desconhecer o Scratch, bem como conceitos fundamentais do Pensamento Computacional e Algébrico. Os algoritmos com estruturas mais complexas foram produzidos por discentes que tinham experiência prévia com programação ou demonstravam interesse por esse tema. Já a atividade de programação proposta nessa pesquisa nos possibilitou identificar três tipos de dificuldade dos sujeitos em relação a programação e refletir sobre suas causas. A saber as dificuldades se deram em relação a: compreensão de objetos matemáticos, simbolização e elaboração de algoritmos; observamos que em alguns casos a interface e estrutura do Scratch auxiliou os estudantes na elaboração do algoritmo, visto que estes sujeitos não conseguiram elaborar o algoritmo em linguagem materna. Por fim, destacamos que nesse estudo mostramos que é possível abordar esses temas nos cursos de licenciatura em Matemática sem realizar projetos de programação longos e complexos, uma vez que mesmo projetos curtos, como o desenho de um quadrado, trabalham com uma variedade de conceitos e práticas desses pensamentos.

Palavras-chave: Formação de professores; Educação Matemática; Pensamento Computacional; Pensamento Algébrico; Tecnologias Educacionais.

ABSTRACT

Nowadays, there is a significant interdependence between physical and digital spaces that tends to expand over time. This relationship has changed the way we live, think and act within our society, the school is not immune to these changes. It is necessary to discuss how to integrate digital technologies into the school routine, to prepare students to live in this society highly dependent on the cyberspace. For this integration to occur, it is necessary to pay attention to teacher training. In particular, we turn our attention to the initial training of Mathematics teachers. Given this context, this study sought to answer the following question: What aspects of Computational Thinking and Algebraic Thinking Mathematics teachers in initial training manifested by realizing activities with Scratch? For this we conducted an exploratory case study with a mixed approach. The subjects of this research were 14 academics of the fourth years, full and night periods, in the initial training of Mathematics teachers of a Higher Education institution in the state of Paraná. Empirical data were collected during a didactic sequence developed by the authors and validated by the Research Group on Educational Technologies in Mathematics (GTEM). The data collection instruments used were pre-implementation questionnaire, algorithms for constructing a square and a triangle in native language and in Scratch. Qualitative data were analyzed according to Bardin's Content Analysis, while quantitative data were analyzed using simple percentages generated from the frequencies of the categories. Based on Radford, Brennan and Resnick we outline two general aspects of the algorithms: algebraic and computational. Algebraic aspects are subdivided into objectification, symbolization and generalization, while computational aspects are subdivided into structure and debugging. These aspects were analyzed using indicators produced during this research. Among the results obtained, we highlight that: a significant portion of the subjects claimed not to know Scratch, as well as fundamental concepts of Computational and Algebraic Thinking; the algorithms with more complex structures were produced by students who had previous experience with programming or showed interest in this topic; the programming activity proposed in this research allowed us to identify three types of difficulties for subjects in relation to programming and reflect on their causes, the difficulties occurred in relation to: understanding of mathematical objects, symbolization and elaboration of algorithms; we observed that in some cases the Scratch interface and structure helped students in the elaboration of the algorithm, since these subjects were not able to elaborate the algorithm in native language. Finally, we highlight that in this study we show that it is possible to address these themes in the initial training of Mathematics teachers without carrying out long and complex programming projects, since even short projects, such as the design of a square, work with a variety of concepts and practices of those thoughts.

Keyword: Teacher training; Mathematical Education; Computational Thinking; Algebraic Thinking; Educational Technologies.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE SIGLAS

TD	Tecnologias Digitais
PC	Pensamento Computacional
PM	Pensamento Matemático
ABP	Aprendizagem Baseada em Projetos
PCN	Parâmetros Curriculares Nacionais
SBIE	Simpósio Brasileiro de Informática na Educação
WIE	Workshop de Informática na Escola
RBIE	Revista Brasileira de Informática na Educação
WEI	Workshop sobre Educação em Computação
WEIT	Workshop Escola de Informática Teórica
RENOTE	Revista Novas Tecnologias na Educação
DesafIE	Workshop de Desafios da Computação Aplicada a Educação
PA	Pensamento Algébrico
SBC	Sociedade Brasileira de Computação

SUMÁRIO

INTRODUÇÃO

Atualmente a sociedade exige que professores sejam capazes de trabalhar de forma coerente com os corpos teóricos estabelecidos pela ciência e pela tecnologia, sabendo relacioná-los com a realidade educacional e social dos alunos usando diversas metodologias de ensino. Contudo, entendo que a graduação não é o suficiente para formar um profissional com estas competências, desse modo a formação continuada é fundamental.

Em vista disso, como licenciado recém-formado busquei conhecer melhor os tipos de pós-graduação e áreas de atuação existentes para o professor de Matemática. Após muita reflexão optei pela pós-graduação *strictu sensu*, primeiramente porque a ciência me fascina e desejo contribuir com a produção e divulgação de pesquisas, segundo porque acredito que esta modalidade de pós-graduação propicia uma formação com maior profundidade e rigor científico em relação à *lato sensu*. Essa escolha não foi fácil frente a um cenário de crescente desvalorização da ciência, evidenciado pelos discursos que desqualificam cientistas e seus trabalhos, cortes de verba e descontinuidade de políticas de incentivo à formação científica.

Optei pelo Programa de Pós-Graduação em Ensino de Ciências e Educação Matemática (PGECEM) da Universidade Estadual de Ponta Grossa (UEPG) devido a diversidade do corpo docente e discente, outra razão para essa escolha foi conhecer e gostar de trabalhar com minha orientadora antes de iniciar o mestrado.

Meu projeto inicial buscava explorar o uso de jogos digitais (*games*) para o ensino de matemática na formação continuada de professores de desta área, no entanto, ao aprofundar minha pesquisa percebi que há um significativo número de trabalhos sobre essa temática, além disso uma competência relacionada ao tema chamou minha atenção, o Pensamento Computacional (PC).

O PC é uma competência que pode ser desenvolvida por meio de *games* e tem ganhado espaço nos currículos nacionais e internacionais, por essas razões, eu e minha orientadora, optamos por investigar essa temática. Observamos que muitas vezes o PC é relacionado ao Pensamento Matemático (PM), no entanto, essa correlação costuma acontecer de forma superficial, isto é, sem uma caracterização do PM ou explicitação dos pontos comuns entre esses dois tipos de pensamento. Realizamos um levantamento bibliográfico e percebemos que

o PC pode ser correlacionado com o Pensamento Algébrico (PA), uma ramificação do PM. Assim, optamos por investigar a relação do PC com o PA.

Portanto, esta pesquisa visa responder a seguinte questão norteadora: Quais aspectos do PC e PA são evidenciados por licenciandos em Matemática na realização de atividades com o *Scratch*?

As Tecnologias Digitais (TD), dispositivos que permitem a transformação de qualquer tipo de informação em linguagem binária, tem se propagado exponencialmente ao ponto de permear quase todos os segmentos da sociedade. Segundo Pereira (2013) a incorporação dos equipamentos e das linguagens originadas da computação eletrônica no cotidiano das pessoas é o que denominamos cibercultura. Para esclarecer o que é a cibercultura faz-se necessária uma reflexão sobre os conceitos que sustentam essa ideia, isto é, sobre o virtual, o ciberespaço e as inteligências coletivas.

Pierre Lévy (1996) argumenta que o virtual está relacionado à cognição humana, virtualizar seria o ato de imaginar ou pensar possibilidades que podem ou não se concretizar. O autor defende que somos constituídos pelo virtual, pois nossa percepção de realidade (e de identidade) pode ser considerada como resultado de nossa capacidade virtualizante. Vale ressaltar que o mundo virtual não é perceptível pelo tato, mas sim pela nossa mente, com as TD o virtual deixa de se restringir a cognição humana, uma vez que os computadores também são capazes de simular possibilidades.

A realidade virtual composta pela memória e interconexão dos computadores compõe o ciberespaço, contudo o computador não é o centro desse ciberespaço, mas sim um fragmento (LÉVY, 2000). Os hipertextos, a diversidade de linguagens, as constantes transformações, os saberes e os sujeitos que os possuem também compõem o ciberespaço (LÉVY, 1996).

Segundo Lévy (1998) a inteligência coletiva é "[...] uma inteligência distribuída por toda parte, incessantemente valorizada, coordenada em tempo real, que resulta em uma mobilização efetiva das competências" (p. 28). A inteligência coletiva visa o reconhecimento das competências dos indivíduos, com o intuito de coordená-las para serem usadas em prol da coletividade. O ideal da inteligência coletiva "[...] é reconhecer que a diversidade das atividades humanas, sem nenhuma exclusão, pode e deve ser considerada, tratada, vivida como 'cultura' [...]" (LÉVY, 1996, p. 120).

Como a inteligência coletiva tem um caráter aberto e participativo ela favorece a cibercultura, todavia ela também acelera as modificações técnicas podendo ser um fator de exclusão para aqueles que não estejam envolvidos nesse contexto. A articulação das inteligências coletivas, a incorporação do ciberespaço, dos equipamentos e das linguagens oriundas da computação no cotidiano das pessoas é o que denominamos cibercultura (PEREIRA, 2013).

A presença constante de TD provocou mudança no modo que uma significativa parcela da população vive, pensa e entende a realidade. Isso gerou uma descontinuidade entre gerações, Prensky (2001) sugere que atualmente a maioria dos estudantes são nativos digitais, enquanto a maioria dos professores é imigrante digital (PRENSKY, 2001).

Os nativos digitais costumam estar conectados constantemente a ambientes virtuais, por isso encontram conforto no mundo online, recebem e processam informações num ritmo acelerado, contam com a internet para procurar qualquer informação que precisem e costumam expressar-se por meio das TD (FRANCO, 2013). Ainda, segundo Prensky (2001) os nativos aprendem, pensam e interagem com facilidade em contextos digitais.

Para Prensky (2001) a maioria dos imigrantes digitais são sujeitos que não nasceram no mundo digital, mas incorporaram alguns ou muitos aspectos das TD como usar a internet como segunda opção para a busca de informações, ou ler o manual de um *software* ao invés de assumir que aprenderá a usar ao explorá-lo (PRENSKY, 2001).

Contudo, Kesharwani (2020) aponta que uma significativa parcela de pessoas mais velhas pode ter maior domínio sobre as TD que os jovens, além disso muitos jovens também podem apresentar considerável dificuldade para utilizar esses recursos. Assim, não podemos distinguir nativos e imigrantes digitais com base na idade do sujeito, essa diferenciação depende de vários fatores como o interesse e condição socioeconômica.

Atualmente há um significativo crescimento de profissões que criam ideias e produtos digitais em vez de objetos, isso indica que o universo do trabalho está se vinculando cada vez mais à emergência do ciberespaço (MARTINS *et al.*, 2015). Mugayitoglu (2016) estima que até 2022 1,3 milhões de empregos relacionados à matemática e à computação sejam criados, representando um crescimento de 18% na demanda de profissionais para essas áreas nos Estados Unidos. Supomos que no Brasil esses campos de trabalho também tendem a ter um grande crescimento nos próximos anos.

Em visto disso, futuramente, enquanto cidadãos, espera-se que além de utilizar a informação para construir conhecimento, sejamos capazes de criar e modificar sistemas (programas, jogos, vídeos, entre outros) por conta própria.

O ambiente escolar ainda apresenta uma resistência às TD e aos conhecimentos que elas produzem, contudo, entendemos que é preciso transformar esse cenário. Os discentes devem ser instigados a investigar os objetos de conhecimento na sala de aula e fora dela, presencialmente ou online.

Diante desse contexto reconhecemos a necessidade de incorporar esses recursos à dinâmica escolar e de preparar o professor, em particular os futuros professores de matemática, para conseguirem potencializar os processos de ensino-aprendizagem utilizando as TD.

Os cursos de licenciatura em matemática possuem Diretrizes Curriculares que preveem que o acadêmico deve familiarizar-se com o uso de TD como instrumentos de trabalho, além disso, se espera que ele seja incentivado a utilizá-las para o ensino de matemática (GATTI; NUNES, 2009). Segundo Gatti e Nunes (2009) disciplinas que apenas realizam discussões teóricas acerca da informática no ensino, sem promover atividades práticas, podem não ser suficientes para que os licenciandos agreguem as TD em sua prática futura. Borba, Silva e Gadanidis (2014) reforçam essa colocação ao apontar que não basta explorar apenas os recursos inovadores de uma TD, é essencial explorar o uso de suas potencialidades no processo de ensino.

Resnick *et al.* (2009) afirmam que os nativos digitais, muitas vezes, são apenas consumidores de tecnologia e não produtores, ou seja, demonstram uma grande facilidade com as TD, porém não tem facilidade para se expressar e criar por meio das TD, portanto uma significativa parcela dos nativos sabe apenas “ler” o que é produzido com as TD e não têm ideia de como usar esses recursos para “escrever”.

De Lemos (2019) destaca que no início da era da internet os usuários eram apenas leitores ou expectadores, à medida que as TD evoluíram os sujeitos começaram a transitar de consumidores para produtores de conteúdo.

Atualmente é possível encontrar vídeos, jogos, aplicativos, *wikis*, resenhas, histórias, enfim, há uma imensidade de materiais produzidos pelos nativos digitais na internet. Os nativos

estão cada vez mais se tornando produtores de conteúdo e tecnologia, contudo, ressaltamos que ainda há estudantes que são ainda apenas consumidores.

Assim, os professores precisam ser preparados para buscar estratégias de ensino que atendam tanto os discentes que dominam as TD quanto àqueles que não dominam. Nesse sentido também vemos a necessidade de incorporar na dinâmica escolar ações que permitam o desenvolvimento da "escrita" digital e PC. A programação é um meio para desenvolver essas competências.

Vários países têm implementado o PC como componente curricular obrigatório, visando utilizar a programação para desenvolvê-lo (VALENTE, 2016). Na prática os esforços têm se concentrado na busca por meios de desenvolver o PC transversalmente ao conteúdo de outras disciplinas. Assim espera-se que os estudantes desenvolvam o PC enquanto aprendem os conteúdos específicos da disciplina, usar a programação para ensinar é apontada como uma das formas de se fazer isso (MORENO-LEÓN; ROMÁN-GONZÁLEZ; ROBLES, 2018). No Brasil o PC aparece brevemente no componente Matemática da Base Nacional Comum Curricular (BNCC).

As competências estimuladas pelo PC também estão relacionadas à resolução de problemas, pois abrangem a capacidade de ler, interpretar e compreender situações propostas e transpor estas informações para representações científicas, matemáticas ou sociais. No contexto da Educação Matemática, em particular, Mestre *et al.* (2015) apontam que as habilidades estimuladas pelo PC são similares às competências abordadas em avaliações baseadas em resolução de problemas, como o PISA.

Corrêa, Pereira e Grossi (2018) realizaram um levantamento das teses e dissertações nacionais que discutem PC, nesse estudo os autores coletaram os dados no Catálogo de Teses e Dissertações da CAPES e o analisaram publicações realizadas entre 2014 e 2018.

Os autores observaram sete eixos temáticos, são eles: elaboração de estratégias ou recursos pedagógicos para o desenvolvimento do PC; ensino de computação; avaliação do PC; desenvolvimento do PC; relação entre resolução de problemas matemáticos e PC; problematização do PC na Educação; e percepção de professores da Educação Básica (CORRÊA; PEREIRA; GROSSI, 2018).

Nesse artigo é constatado que estudos que problematizam o PC na Educação são escassos, assim como trabalhos que propõem instrumentos ou discutem como esta competência pode ser avaliada e estudos que utilizam abordagens desplugadas, na seção 1.2 caracterizamos esse tipo de atividade. Além disso, dentro do universo dessa pesquisa, observou-se uma significativa carência de estudos que abordem o PC na formação inicial e continuada de professores (CORRÊA; PEREIRA; GROSSI, 2018).

Barcelos (2014) aponta que poucos estudos com o objetivo de preparar professores para trabalhar conjuntamente a Matemática e o PC foram realizados. Considerando que várias instituições manifestam interesse e apoio para a incorporação do PC na Educação Básica (VALENTE *et al.*, 2017), entendemos que é necessário explorar esta temática.

Vale ressaltar que ainda há poucos estudos sobre a inserção da programação na formação inicial de professores (RAMOS; ESPADEIRO, 2014). Contudo, Kalelioğlu e Gülbahar (2014) apontam que a utilização do *Scratch*, um ambiente de programação baseada em blocos, possibilita aos professores aprenderem conceitos de programação de uma forma implícita, podendo se beneficiar de seus aspectos interdisciplinares ao serem aplicadas no currículo. Em vista disso optamos por utilizar o *Scratch*[1] nesse estudo.

[1] MASSACHUSETTS INSTITUTE OF TECHNOLOGY, **Scratch** - Imagine, Program, Share. Disponível em: https://scratch.mit.edu. Acesso em: 27 de agosto de 2019.

QUESTÕES, OBJETIVOS E PRESSUPOSTOS DE PESQUISA

Diante desse contexto, a **pergunta norteadora** desta pesquisa é: Quais aspectos do PC e PA são evidenciados por licenciandos em Matemática na realização de atividades com o *Scratch*?

Desta pergunta de pesquisa, ramificam-se outros questionamentos:

1. Quais relações podem ser estabelecidas entre o PC e PA?
2. Como o PA pode ser analisado em projetos do *Scratch*?

Com consequência as questões de pesquisa os seguintes objetivos foram definidos:

Objetivo Geral: Apontar aspectos do PC e PA mobilizados por licenciandos em Matemática ao realizar atividades com o *Scratch*.

Objetivos Específicos:

1. Apontar relações entre PC e PA;
2. Delinear indicadores de PC e PA em algoritmos produzidos em linguagem Materna e no *Scratch*.

Considerando que a literatura sobre o PC, PA e uso do *Scratch* na formação inicial de professores de Matemática ainda é escassa, nessa pesquisa optamos por não partir de hipóteses. Decidimos partir de três pressupostos fundamentais, sendo eles:

1. Os licenciandos desconhecem o que é PC e sua relação com a Matemática;
2. Os licenciandos estão familiarizados com a álgebra, no entanto, desconhecem o conceito de PA e sua relação com a programação;
3. A maioria dos licenciandos não tem experiência ou familiaridade com programação.

Estrutura da dissertação

A “Introdução” apresenta as motivações que levaram a produção desta pesquisa, a justificativa, questões, pressupostos e objetivos de pesquisa, além de aspectos referentes à organização textual do trabalho.

No primeiro capítulo, “Pensamento Computacional”, abordamos as definições de PC, seu desenvolvimento histórico, métodos para desenvolver e avaliar essa competência, e discorremos sobre o estado da arte das pesquisas sobre o PC na Educação Básica e na formação de professores.

No segundo capítulo, “Introdução ao *Scratch*”, será apresentaremos o Construcionismo, principal influência teórica sobre o desenvolvimento desse *software*. Além disso, abordamos as principais características do *Scratch* e discutiremos como seu uso pode contribuir para a Educação. No capítulo três, “Pensamento Algébrico”, discutiremos o que é PM, caracterizamos o PA, e correlacionamos o PA com o PC e o *Scratch*.

No capítulo quatro, “Metodologia de Pesquisa”, discorreremos sobre os procedimentos e aspectos metodológicos adotados neste trabalho, apresentaremos ainda os sujeitos de pesquisa, os instrumentos de coleta de dados e os métodos de análise que empregamos.

No quinto capítulo, “Resultados e Análise”, apresentaremos e discutiremos os resultados obtidos. No sexto capítulo, “Considerações Finais”, respondemos as questões de pesquisa, apresentamos nossas reflexões sobre as contribuições deste estudo e apontamos estudos futuros que podem ser desenvolvidos sobre essa temática.

CAPÍTULO 1 PENSAMENTO COMPUTACIONAL

Neste capítulo, discutiremos diversas definições acerca do PC e, partindo dessas reflexões, apresentaremos a definição que embasa esta pesquisa. Discutiremos também as metodologias que têm sido utilizadas para desenvolver e avaliar este tipo de pensamento, bem como o estado da arte de pesquisas nacionais e internacionais sobre o tema. Além disso, apresentaremos algumas contribuições que o PC pode trazer para a Educação e, em particular, para a formação de professores.

1.1 CARACTERIZAÇÃO DO PENSAMENTO COMPUTACIONAL

O termo PC se popularizou por meio de um artigo intitulado *Computational Thinking* publicado em 2006 por Jeannette Wing, no qual a autora delineia o que é PC e sua relevância para a sociedade. Os argumentos apresentados por Wing propiciaram uma nova perspectiva sobre a relação entre computadores e humanos e provocaram intensa discussão na comunidade científica (SHUTE; SUNA; ASBELL-CLARKE, 2017).

Apesar do termo ter sido cunhado por Wing em 2006, há evidências na literatura que sugerem que a ideia de PC existe há mais tempo. Denning (2009) relata que o PC se originou do Pensamento Algorítmico discutido nos anos 1950-1960. Segundo Teixeira (2017) durante esse período PC estava associado a ideia de pensar a solução de um problema como um algoritmo que converte uma entrada em uma saída.

Na década de 80, Tikhomirov (1981) apontou em seu trabalho que o computador possibilitaria uma nova relação entre o ser humano e o mundo, o autor previa que o computador instigaria uma nova forma de pensar. Neste período Papert (1980) associou práticas no campo da ciência da computação com o pensar, esse tema tornou-se central em seu trabalho com a linguagem de programação LOGO (YADAV *et al.*, 2017).

Papert (1980) sugeriu que a programação de computadores poderia influenciar a forma com que as pessoas pensariam futuramente e esse novo pensar iria além do ato de programar, abrangendo também problemas do cotidiano. Papert argumenta em seu trabalho, que na programação o erro é frequente e positivo, pois por meio dele o sujeito pode descobrir como acertar. Esse processo é chamado de depuração e promove o que ele denominou Pensamento Procedural, essas ideias embasam a teoria do Construcionismo proposta por Papert.

Em 1996, Papert relacionou pela primeira vez o termo PC com o Construcionismo num artigo sobre Educação Matemática (PAPERT, 1996). Neste artigo o autor sugere que PC está relacionado a usar computadores para "forjar ideias" que sejam explicativas e acessíveis, assim esse tipo de pensamento ajudaria às pessoas analisar e entender melhor problemas e suas soluções.

O PC se popularizou quando Wing (2006) sugeriu que esse estilo de pensamento pode ser exercido por qualquer pessoa, ou seja, seria uma atitude/habilidade acessível a qualquer indivíduo, mesmo sem formação específica em computação.

Quanto à caracterização, Wing (2006) ressaltou que PC refere-se a forma que humanos pensam, é uma habilidade fundamental e não-rotineira, trata-se de conceituação e não programação, complementa e combina o pensar matemático e das engenharias, e lida com ideias e não artefatos.

Ao longo dos anos o termo PC foi ganhando maturidade na comunidade e Wing foi reformulando a definição de PC. Posteriormente, a autora apresentou o PC como um tipo de pensamento analítico relacionado com o PM, Pensamento de Engenharia e Pensamento Científico (WING, 2008). Em 2014 a autora consegue sintetizar a definição de PC, quando no artigo considera que PC:

> [...] são os processos de pensamento que envolvem a formulação de um problema e na expressão de sua(s) solução(ões) de tal forma que um computador - humano ou máquina - possa executá-la(s) (WING, 2014, p. 1, tradução nossa)[2].

Ressaltamos que a definição proposta em 2008 é vaga, tendo em vista que Wing não esclarece o que caracteriza os tipos de pensamento que são relacionados ao PC. A definição proposta em 2014 é mais concreta que as anteriores, assenta-se na ideia de que PC tem uma íntima relação com a abstração, no entanto os processos de pensamento que compõem o PC não são explicitados.

Entendemos que é preciso caracterizar esse tipo de pensamento com maior rigor, uma vez que compreender o que é PC é essencial para discutirmos como ele pode ser desenvolvido e avaliado. Diante disso, buscamos esclarecer nosso entendimento sobre o que é PC a partir dos autores que serão apresentados na sequência

[2] Computational thinking is the thought processes involved in formulating a problem and expressing its solution(s) in such a way that a computer—human or machine—can effectively carry out.

Mannila *et al*. (2014) apontam algumas similaridades entre as definições propostas por Wing e Papert, ambas indicam claramente que PC não se resume a uma ferramenta, que ele tem a computação como elemento crucial e está intimamente relacionado ao desenvolvimento e explicação de um raciocínio. Para além disso, tanto Wing quanto Papert apontam a abstração como componente essencial do PC (ROMERO; LEPAGE; LILLE, 2017).

Percebemos que há pontos de convergência entre as concepções, no entanto, ressaltamos que Papert não propôs uma definição de PC e a definição elaborada por Wing (2014) não explicita os processos de pensamento que compõem o PC. Com intuito de esclarecer esses pontos recorreremos às definições de PC propostas por outros pesquisadores.

O National Research Council (2011) apontou a existência de quatro visões predominante sobre PC. A primeira visão o concebe o PC como a capacidade de entender o mundo como objetos ou peças que interagem entre si ou possuem alguma relação. A segunda entende PC como uma estrutura de colaboração facilitada por recursos computacionais. A terceira é influenciada por Papert e defende que o foco do PC é o processo de construção de mundos simulados. A quarta também enfatiza a simulação de situações, mas com objetivo de verificar possibilidades.

Brennan e Resnick (2012), por sua vez, descrevem o PC por meio de três dimensões essenciais: conceitos, práticas e perspectivas computacionais. Os conceitos computacionais são conceitos que designers empregam enquanto programam, os autores argumentam que estes conceitos são fundamentais para o PC, sendo eles: sequências, ciclos, paralelismo, eventos, condicionais, operadores e dados.

Práticas computacionais são as habilidades que os designers empregam enquanto programam, estas práticas colocam os conceitos fundamentais do PC em uso para desenvolver projetos, sendo elas: ação iterativa e incremental, teste e depuração, reutilização e reformulação, abstração e modulação (BRENNAN; RESNICK, 2012).

Ainda se faz necessário explicar que os designers formam as perspectivas computacionais sobre o mundo ao seu redor e sobre si mesmos a partir de tais ações, que são: expressar, conectar e questionar. Na seção 1.2 aprofundaremos nossas reflexões sobre as dimensões propostas por Brennan e Resnick (2012).

Outros autores que se debruçaram sobre o conceito foram Selby e Woollard (2014) realizaram uma ampla busca na literatura para identificar quais componente são utilizados para descrever o PC. Sua pesquisa resultou na seguinte definição de PC:

> [...] Pensamento Computacional é uma atividade associada à solução de problemas, geralmente resultando em um artefato ou produto. O Pensamento Computacional é um processo cognitivo que resulta em uma automação desenvolvida pelo uso da abstração, decomposição, design algorítmico, avaliação e generalização. (SELBY; WOOLLARD, p. 20, 2014, tradução nossa).[3]

Os autores também identificaram consenso no entendimento de que PC é um tipo específico de solução de problemas, similar a um conjunto de ferramentas cognitivas para resolver problemas. Há consenso também que as soluções elaboradas podem ser implementadas por dispositivos computacionais (SELBY; WOOLLARD, 2014).

Shute, Suna e Asbell-Clarke (2017) buscaram comparar o PC com o Pensamento Matemático (PM), pensamento de engenharia, pensamento de design e pensamento de sistema. Nesta pesquisa os autores tentam diferenciar o PC do pensamento de engenharia, de design e de sistema, e aproxima o PC do PM.

Estes autores entendem que o PM envolve a aplicação de habilidades matemática para resolver problemas matemáticos, e que ele é composto por três partes: crenças sobre matemática, processos de solução de problema e justificativa para essas soluções. A partir dessa concepção os autores sugeriram que os seguintes conceitos são comuns ao PM e PC: solução de problemas; modelagem; análise de dados e interpretação; e estatística e probabilidade (SHUTE; SUNA; ASBELL-CLARKE, 2017).

Romero, Lepage e Lille (2017) conceituam PC como um conjunto de estratégias cognitivas e metacognitivas relacionadas à identificação e enquadramento de problemas, letramento de códigos e programação criativa. Esse conjunto seria um meio de desenvolver novas estratégias de pensamento para analisar, identificar e organizar tarefas relativamente complexas ou mal definidas, assim como desenvolver a capacidade criativa para resolver problemas.

[3] […] computational thinking is an activity associated with problem solving, often resulting in an artefact or product. Computational thinking is a cognitive process resulting in an automation that is developed by the use of abstraction, decomposition, algorithmic design, evaluation, and generalization.

Os autores argumentam que o núcleo do PC é a capacidade de transpor um significado abstrato em um significado concreto, isto é, refinar um conceito abstrato específico em algo como um programa de computador ou algoritmos. Nesse sentido a relação entre a ciência da computação e PC é similar à relação entre pensamento algorítmico e a Matemática (ROMERO; LEPAGE; LILLE, 2017).

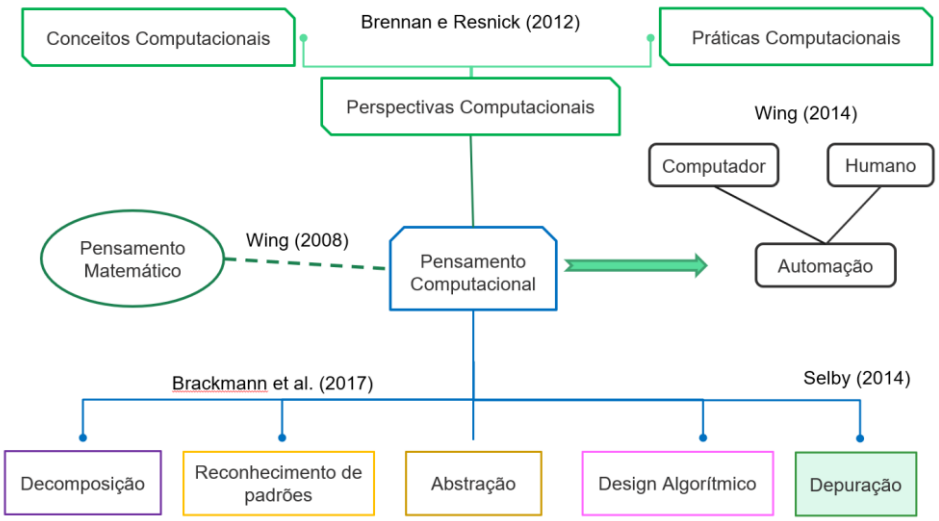
Para a Sociedade Brasileira de Computação (SBC) o PC é a "[...] habilidade de compreender, definir, modelar, comparar, solucionar, automatizar e analisar problemas (e soluções) de forma metódica e sistemática" (p. 2, 2017), em outras palavras, trata-se da capacidade de "[...] sistematizar a atividade de resolução de problemas, representar e analisar as soluções através de algoritmos" (p. 3, 2017). Ainda segundo a SBC (2017) o PC usa objetos abstratos para caracterizar informações e os processos que manipulam estas informações.

Brackmann *et al.* (2017). relatam estudos que propõem quatro dimensões (ou pilares) para o PC, sendo eles: Decomposição, Reconhecimento de Padrões, Abstração e Algoritmos.

> O Pensamento Computacional envolve identificar um problema complexo e quebrá-lo em pedaços menores e mais fáceis de gerenciar (Decomposição). Cada um desses problemas menores pode ser analisado individualmente com maior profundidade, identificando problemas parecidos que já foram solucionados anteriormente (Reconhecimento de padrões), focando apenas nos detalhes que são importantes, enquanto informações irrelevantes são ignoradas (Abstração). Por último, passos ou regras simples podem ser criados para resolver cada um dos subproblemas encontrados (Algoritmos). (BRACKMANN *et al.*, 2017, p. 983).

Na figura 1.1 apresentamos um esquema que relaciona os conceitos e ideias discutidos nesta seção que nos ajudaram a caracterizar o PC.

Figura 1.1 – Caracterização do PC.



Fonte: Autores.

Assim, no contexto deste trabalho, entendemos PC como um processo cognitivo associado à solução de problemas e com o PM. Esse processo é composto por cinco pilares fundamentais: decomposição, reconhecimento de padrões, abstração, depuração e *design* algorítmico. Esse tipo de pensamento é embasado em conceitos computacionais e práticas fundamentais da computação, podendo ocasionar perspectivas computacionais. O PC resulta em uma automação que pode ser implementada por algum agente de processamento de informações – computadores e humanos.

1.2 DESENVOLVIMENTO E AVALIAÇÃO DO PENSAMENTO COMPUTACIONAL

Assim como não há consenso na definição de PC, também não se tem consenso sobre quais habilidades devem ser trabalhadas ou quais metodologias devem ser adotadas para desenvolver o PC (SELBY, 2014), diversas metodologias de ensino, recursos e estratégias têm sido adotadas com o intuito de promover o desenvolvimento do PC.

Em relação às metodologias de ensino percebemos que dois tipos têm sido empregados para desenvolver o PC: resolução de problemas e a aprendizagem baseada em projetos. Na resolução de problemas os discentes são desafiados a resolver problemas abertos ou fechados, cabe ao docente decidir quando e quais deverão ser resolvidos, bem como auxiliar os estudantes na resolução do problema (FLOS; VILAHUR, 2016).

Na aprendizagem baseada em projetos os discentes assumem um papel ativo e criativo, desenvolvendo um projeto real que aborda questões essenciais para os estudantes e proporciona uma compreensão duradoura sobre a temática. Nessa metodologia o professor é um orientador e facilitador que ajuda os educandos quando necessário. No contexto da ciência da computação o projeto, por exemplo, pode ser o design de um *software*, um jogo digital ou uma modelagem computacional (FLOS; VILAHUR, 2016).

Constatamos que dois tipos de estratégias que têm sido utilizados com maior frequência: atividades computadorizadas e desplugadas. Atividades computadorizadas dependem de dispositivos digitais para serem executadas, enquanto as atividades desplugadas podem ser realizadas sem o auxílio desses recursos (KALELIOGLU; GÜLBAHAR; KUKUL, 2016).

Em relação aos recursos empregados notamos cinco tipos: programação baseada em texto, programação de objetos físicos, atividades desplugadas e ambientes visuais de programação baseados em seta ou em blocos (MORENO-LEÓN; ROMÁN-GONZÁLEZ; ROBLES, 2018).

As linguagens de programação textual são *softwares* nos quais os usuários precisam digitar os comandos e tomar cuidado com a sintaxe específica da linguagem de programação que está sendo utilizada. Como exemplo de linguagens textuais profissionais temos o *Java* e o *Python*, em relação às linguagens textuais educacionais podemos citar o Logo (MORENO-LEÓN; ROMÁN-GONZÁLEZ; ROBLES, 2018).

Os *softwares* de programação de objetos do mundo físico permitem que os usuários elaborem programas que controlam objetos concretos. Essa categoria pode ser dividida em duas subcategorias: brinquedos programáveis e robôs ou placas. Brinquedos programáveis são objetos com botões que possibilitam aos usuários programá-los diretamente; Robôs ou placas são equipamentos controlados por um código-fonte que foi escrito em dispositivo digital e, posteriormente, transferido para o dispositivo físico (MORENO-LEÓN; ROMÁN-GONZÁLEZ; ROBLES, 2018).

Atividades desplugadas são atividades que envolvem quebra-cabeças, jogos lógicos, jogos de cartas, entre outros. O objetivo delas envolve algum conceito da ciência da computação como algoritmos, transmissão de dados ou representação de dados. Como as atividades não visam a implementação desses conceitos, podem ser realizadas sem um dispositivo digital (MORENO-LEÓN; ROMÁN-GONZÁLEZ; ROBLES, 2018).

As atividades desplugadas usualmente exigem pouca infraestrutura e os materiais envolvidos são de baixo custo, se constituindo como uma boa alternativa para ampliar o acesso ao PC em situações ou ambientes que não se têm à disposição de uma estrutura física com computadores, dispositivos móveis ou internet.

Ambientes visuais baseados em seta são *softwares* que permitem ao usuário estabelecer sequências de comandos usando setas ou ícones, esses ambientes podem ser utilizados até mesmo por crianças que não aprenderam a ler, pois são intuitivos e fáceis de usar (MORENO-LEÓN; ROMÁN-GONZÁLEZ; ROBLES, 2018).

Ambientes visuais baseados em blocos são *softwares* que permitem ao usuário estabelecer uma sequência de comandos encaixando blocos ou pedaços de código. Os códigos produzidos podem ser personalizados por meio de parâmetros, esses ambientes exigem que o usuário saiba ler e escrever (MORENO-LEÓN; ROMÁN-GONZÁLEZ; ROBLES, 2018). Usualmente blocos que não fazem sentido lógico e, em conjunto, não podem ser encaixados uns nos outros, impossibilitando que o usuário cometa erros de sintaxe.

As linguagens de programação textuais, visuais ou relacionadas a objetos do mundo físico se encaixam nas estratégias computacionais, enquanto um amplo leque de recursos analógicos pode ser usado para promover atividades desplugadas. No Quadro 1.1 apresentamos uma síntese das metodologias, estratégias e recursos empregados para desenvolver o PC.

Quadro 1.1 - Síntese das metodologias, estratégias e recursos utilizados no desenvolvimento do PC.

Metodologias	**Estratégias**	**Instrumentos**
Aprendizagem Baseada em Projetos Resolução de problemas	Computadorizadas	Programação baseada em texto
		Programação de objetos físicos
		Ambientes visuais de programação baseados em setas
		Ambientes visuais de programação baseados em blocos.
	Desplugadas	Quebra-cabeças, jogos lógicos, jogos de cartas, entre outros.

Fonte: Autores.

Apesar da programação ser utilizada com bastante frequência, Selby (2014) aponta que não há consenso sobre a incorporação ou não da programação no desenvolvimento do PC. Ser capaz de programar é um dos benefícios do PC, no entanto, as habilidades do PC não se resumem às habilidades de programação (SHUTE; SUNA; ASBELL-CLARKE, 2017), visto que o PC também pode constituir a forma com que pensamos e resolvemos problemas do cotidiano (WING, 2006).

Curzon *et al.* (2009) sugerem que a programação é necessária para o desenvolvimento do PC embora sua importância seja vista como uma ferramenta. Lu e Fletcher (2009, p. 260, tradução nossa) argumentam que “[…] programar está para a ciência da computação do mesmo

modo que a construção de provas está para a matemática, e como a análise de literatura está para o inglês" [4].

Entendemos que embora o PC vá além da programação de computadores, aprender a programar contribui para o desenvolvimento e aplicação do PC (NATIONAL RESEARCH COUNCIL, 2011), pois o ato de programar está relacionado à resolução de problemas possibilitando um amadurecimento da capacidade de abstração, decomposição, codificação e depuração.

Selby (2014) aponta algumas dificuldades que principiantes podem apresentar enquanto programam, são elas: não entender que o computador executa um programa com base em seu estado anterior, não conseguir formular algoritmos com a precisão necessária para instruir o computador, não conseguir expressar sua lógica de pensamento usando alguma linguagem de programação, dificuldade em transpor uma representação de um sistema formal para outro e dificuldade em usar a lógica para compreender a estrutura de um código, rastrear e corrigir erros dentro desse código (depurar).

A autora também indica dificuldades relacionadas à resolução de problemas, sendo elas: entender o problema e suas limitações, decompor o problema em problemas menores e mais fáceis de se resolver, e generalizar a solução de um problema para aplicar em outro (SELBY, 2014).

As dificuldades apresentadas pelos estudantes na resolução de problemas podem acometer o processo de programação e assim prejudicar o desenvolvimento do PC. Com isso em vista, Selby (2014) sugere que as dificuldades que os estudantes apresentam em relação à resolução de problemas devem ser sanadas antes de trabalhar habilidades mais especializadas do PC.

Figueiredo (2017) ressalta que uma parcela significativa de professores ainda não domina habilidades necessárias ao PC. O autor sugere que devido a isso numerosos projetos com o objetivo de incentivar o estudo sobre programação têm sido desenvolvidos. Nesse artigo o autor elenca e descreve alguns desses projetos, sendo eles: Tacle 3, *Scratch*, Alice, *Code.org*, *Khan Academy*, CS *Unplugged*, *Tynker*, *Lightbot*, *Barefoot*, *Code Combat*, *Kodable*, MIT *App*

[4] "… programming is to Computer Science what proof construction is to mathematics, and what literary analysis is to English."

Inventor, *Live Code*, *Touch Develop*, *Blockly*, *Snap* (*build your own blocks*), *Greenfoot*, *Kodu* e *Cubetto*.

Devido à variedade de definições e conceituações não há consenso de como avaliar o PC. Isso dificulta a análise da eficácia de intervenções que objetivam desenvolver o PC de maneira confiável e válida, assim como a comparação de resultados de vários estudos sobre PC. Diante disso muitos pesquisadores tendem a elaborar e aplicar procedimentos próprios para avaliar o PC (SHUTE; SUNA; ASBELL-CLARKE, 2017).

Román-González, Moreno-León e Robles (2019) identificaram e classificaram instrumentos para avaliação do PC com ênfase no currículo K-12[5]. Seus resultados apontam a existência de ao menos sete categorias de instrumentos, sendo elas: instrumentos de diagnóstico, instrumentos somativos, instrumentos formativos-iterativos, instrumentos de mineração de dados, instrumentos de transferências de habilidades, escalas de percepções e atitudes e avaliação de vocabulário.

Os instrumentos de diagnóstico objetivam avaliar o nível de aptidão de PC de um estudante. Podem ser aplicados em condições pré-teste, isto é, com discentes que não possuem qualquer experiência prévia em programação. Podem também ser utilizados em situações de pós-teste, ou seja, após uma intervenção educativa para verificar se o nível de PC do indivíduo aumentou. (ROMÁN-GONZÁLEZ; MORENO-LEÓN; ROBLES, 2019).

Instrumentos somativos objetivam avaliar se o estudante é capaz de executar uma tarefa adequadamente após receber alguma instrução sobre habilidades do PC. Assim, esses instrumentos serão adequados para situações pós-teste. (ROMÁN-GONZÁLEZ; MORENO-LEÓN; ROBLES, 2019).

Os instrumentos formativos-iterativos são úteis para fornecer *feedback* aos estudantes, usualmente esses instrumentos são automatizados. Esse tipo de instrumento não avalia os sujeitos, mas os produtos de sua aprendizagem, como seus projetos de programação. Instrumentos dessa natureza são usados principalmente no decorrer do processo de aprendizagem e costumam ser projetos direcionados para ambientes de programação específicos. (ROMÁN-GONZÁLEZ; MORENO-LEÓN; ROBLES, 2019).

[5] K–12 é o termo utilizado para se referir aos níveis de ensino que precedem o Ensino Superior. Esse sistema de Educação é utilizado por alguns países como os Estados Unidos.

Instrumentos de mineração de dados também são focados no processo de aprendizagem. Diferentemente dos formativos-iterativos esses instrumentos analisam as atividades em tempo real. Segundo os autores esses recursos são especialmente úteis para detectar lacunas e equívocos nos processos de aquisição dos conceitos computacionais (ROMÁN-GONZÁLEZ; MORENO-LEÓN; ROBLES, 2019).

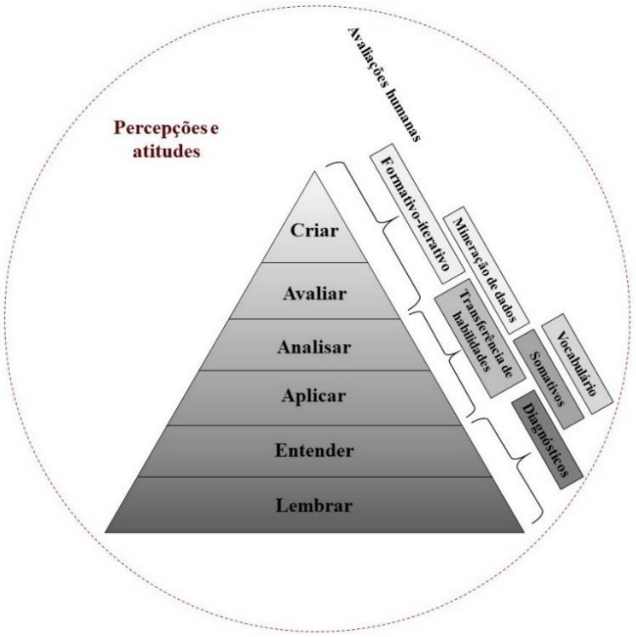
Os instrumentos de transferências de habilidades objetivam avaliar se os discentes são capazes de transferir suas habilidades de PC para diversos tipos de problemas, contextos e situações (ROMÁN-GONZÁLEZ; MORENO-LEÓN; ROBLES, 2019).

Escalas de percepção e atitudes são destinadas a avaliar as concepções e atitudes dos sujeitos sobre PC e sobre questões relacionadas ao PC, tais como a relação do sujeito com: computadores, ciência da computação, programação e até mesmo letramento digital (ROMÁN-GONZÁLEZ; MORENO-LEÓN; ROBLES, 2019).

As avaliações do vocabulário de PC objetivam avaliar como o sujeito expressa verbalmente seu entendimento sobre vários elementos e dimensões do PC. Essas expressões foram denominadas “linguagem do PC” (ROMÁN-GONZÁLEZ; MORENO-LEÓN; ROBLES, 2019).

Román-González, Moreno-León e Robles (2019) também produziram um esquema piramidal, apresentado na Figura 1.2, no qual relacionam as categorias de instrumentos de avaliação do PC à taxonomia revisada de Bloom. Na taxonomia revisada de Bloom os níveis cognitivos são: criar, avaliar, analisar, aplicar, entender e lembrar. Nela “lembrar” tem a menor complexidade cognitiva e criar a maior.

Figura 1.2 - Taxonomia de Bloom e instrumentos de avaliação do PC



Fonte: Adaptado de Román-González, Moreno-Léon e Robles (2019).

Segundo os autores os instrumentos de diagnóstico fornecem informações sobre como os alunos “lembram” e “entendem” alguns conceitos do PC. Os instrumentos de transferência de habilidades por sua vez podem avaliar as competências dos alunos para “analisar” e “aplicar” suas habilidades de PC em diferentes contextos (ROMÁN-GONZÁLEZ; MORENO-LEÓN; ROBLES, 2019).

Os instrumentos formativos-iterativos possibilitam aos alunos “avaliar” seus próprios projetos e de outras pessoas, assim como “criar” projetos melhores e mais complexos. Os instrumentos somativos e avaliação de vocabulário podem ser usados para a transição ente os níveis inferiores e intermediários da taxonomia, já os instrumentos de mineração de dados servem como uma transição entre os níveis intermediários e superiores (ROMÁN-GONZÁLEZ; MORENO-LEÓN; ROBLES, 2019).

Entretanto os autores ressaltam que alguns aspectos do PC como usabilidade e originalidade não podem ser avaliados pelos instrumentos, mas sim, pela sensibilidade humana. Nesse sentido, como as escalas de percepções e atitudes avaliam processos não-cognitivos esses instrumentos não são correlacionados ao longo da pirâmide, mas sim, envolvem e enquadram essa pirâmide (ROMÁN-GONZÁLEZ; MORENO-LEÓN; ROBLES, 2019).

Observamos a existência de uma variedade de instrumentos de desenvolvimento e avaliação do PC, uma vez que muitos autores elaboram procedimentos próprios de acordo com seu entendimento sobre o PC. Nessa pesquisa optamos por utilizar o método avaliativo proposto por Brennan e Resnick (2012).

Brennan e Resnick (2012) propuseram uma estrutura de avaliação e desenvolvimento do PC utilizando o *Scratch*, essa estrutura avalia tanto o entendimento conceitual quanto as aplicações das habilidades de PC e as perspectivas que o sujeito desenvolve nesse processo. O Quadro 1.2 apresenta um resumo das dimensões de avaliação propostas pelos autores.

Quadro 1.2 - Resumo das dimensões do PC propostas por Brennan e Resnick (2012).

Conceitos do PC	**Sequência:** Expressar uma ação como uma série de etapas individuais ou instruções ordenadas de modo que possam ser executadas por um agente de informação. **Ciclo:** Mecanismo para executar uma mesma sequência várias vezes. **Evento:** Gatilho para a execução de determinadas sequências. **Paralelismo:** Coocorrência de múltiplas sequências. **Condicional:** Restrições à execução de sequências, gerando resultados diferentes com base nas condições impostas. **Operador:** Operações matemáticas, lógicas e de *strings* (cadeias). **Dados:** Armazenamento, recuperação e atualização de dados.
Práticas do PC	**Ação iterativa e incremental**: Processos iterativos e adaptativos para projetar e implementar soluções passo a passo. **Teste e depuração**: Processos de tentativa e erro para testar e corrigir defeitos. **Reutilização e reformulação**: Elaborar um novo projeto com base em projetos já existentes. **Abstração e modulação**: Modelar sistemas complexos utilizando elementos básicos. Construir um sistema grande unindo conjuntos de sistemas pequenos e mais fáceis de manipular.
Perspectivas do PC	**Expressar:** Perceber a computação como uma forma de expressão e criação. **Conectar:** Perceber a computação como uma forma de interagir e trabalhar com os outros. **Questionar:** Levantar questões sobre tecnologias, pensar em como usar tecnologias para resolver problemas da vida real.

Fonte: Adaptado de Shute, Suna e Asbell-Clarke (2017).

Brennan e Resnick (2012) também propõem três abordagens para a avaliação dessas dimensões: análise de projetos, entrevistas baseadas em artefatos e desenhos de cenários.

Na análise de projetos a avalição é feita por meio do portfólio de projetos dos discentes, como exemplo Brennan e Resnick (2012) apresentam o *User Analysis* uma das ferramentas do *Scrape*. Essa ferramenta representa os comandos utilizados nos projetos de forma gráfica. Tal gráfico pode ser usado para avaliar a proficiência do programador. Assim, a análise de portfólios possibilita verificar quais conceitos computacionais foram utilizados pelos usuários, no entanto, não esclarece qual é a compreensão do usuário sobre estes conceitos.

As entrevistas baseadas em artefatos possibilitam maior esclarecimento quanto à compreensão da fluência do usuário em relação à conceitos e práticas específicas, visto que a presença de um elemento de código em um projeto não necessariamente indica que o *designer* possui uma compreensão profunda sobre este elemento. Essa abordagem possibilita também a expansão do foco para o processo de criação, enquanto a análise de projetos foca mais sobre o produto (BRENNAN; RESNICK, 2012).

Os desenhos de cenário consistem em apresentar ao estudante três conjuntos de projetos do *Scratch* com complexidade crescente, o discente deverá escolher um dos projetos e explicar o que ele faz, descrever como ele pode ser expandido, corrigir um erro e remixar o projeto original adicionando uma nova característica. Essa abordagem não se baseia em artefatos produzidos pelo estudante, logo esses projetos podem não estar conectados aos interesses pessoais do sujeito. No Quadro 1.3 apresentamos um resumo dos pontos fortes e fracos de cada abordagem.

Quadro 1.3 - Resumo dos pontos fortes e fracos de cada abordagem proposta por Brennan e Resnick (2012).

(continua)

	Conceitos	**Práticas**	**Perspectivas**
Análise de projeto	Presença de blocos de comando indica contato com conceitos computacionais.	-	-
Entrevistas baseadas em artefatos	Nuances de entendimento conceitual. Análise limitada a um conjunto pequeno de projetos.	Possibilita a análise com base em experiências de *design* autênticas. Avaliação sujeita a limitações de memória do entrevistado.	Talvez possam ser avaliadas, no entanto, é difícil perguntar diretamente.

Quadro 1.3 - Resumo dos pontos fortes e fracos de cada abordagem proposta por Brennan e Resnick (2012).

(conclusão)

	Conceitos	Práticas	Perspectivas
Desenhos de cenário	Possibilita avaliar nuances e variedade de entendimentos conceitual. Os projetos podem não ter relação com interesses pessoais do entrevistado.	Possibilita a análise em tempo real e em novas situação. Os projetos podem não ter relação com interesses pessoais do entrevistado.	Talvez possam ser avaliadas, no entanto, é difícil perguntar diretamente.

Fonte: Adaptado de Brennan e Resnick (2012).

As formas de avaliar o PC propostas por Brennan e Resnick inspiraram Moreno-León e Robles (2015) a elaborar um instrumento automatizado para a avaliação do PC, o Dr. *Scratch*[6]. O Dr. *Scratch* analisa projetos elaborados no *Scratch* e infere sobre o desenvolvimento de conceitos computacionais do PC, do *designer* e do projeto com base numa análise estática sobre o código. Esse instrumento fornece uma avaliação somativa e também pode ser utilizado para uma avaliação formativa (TEIXEIRA, 2017).

O Dr. *Scratch* utiliza o código estático do projeto e automatiza o processo de análise dos conceitos computacionais existentes nesse código. Pode ser usado para avaliar um projeto individual, diferentemente da proposta de Brennan e Resnick (2012) no qual se analisa um portfólio de projetos produzidos ao longo de um período (TEIXEIRA, 2017).

Esse instrumento analisa sete aspectos para avaliar o PC do *designer*, são eles: abstração e decomposição de problemas, paralelismo, raciocínio lógico, sincronização, controle de fluxo, interatividade com o usuário e representação de dados (BARCELOS; BORTOLETTO; ANDRIOLI, 2016). Vale ressaltar que os aspectos propostos por Moreno-León e Robles (2015) são diferentes dos conceitos computacionais estabelecidos por Brennan e Resnick (2012).

Cada um desses aspectos pode pontuar de 0 a 3 dependendo dos comandos utilizados na construção do código, o resultado é composto pela soma das pontuações de cada um desses aspectos (TEIXEIRA, 2017). A ferramenta se baseia nas rubricas de avaliação do PC propostas por Moreno-León e Robles (2015), que estão descritas no Quadro 1.4.

[6] DR. SCRATCH, **Dr. Scratch** - Analyze your Scratch projects here! Disponível em: http://drscratch.org. Acesso em 20 de agosto de 2019.

Quadro 1.4 - Rubricas de avaliação da utilização de conceitos do PC.

Conceito	Nível básico (1 ponto)	Nível intermediário (2 pontos)	Nível avançado (3 pontos).
Abstração e decomposição de problemas	Usar mais de um conjunto de códigos e programar mais de um objeto.	Definição de blocos.	Uso de clones[7].
Paralelismo	Associar a execução de dois conjuntos de códigos ao evento "*quando clicar na bandeira verde*".	Associar a execução de dois conjuntos de códigos ao pressionamento de teclas ou clique em um mesmo objeto.	Associar a execução de dois ou mais conjuntos de códigos ao recebimento de uma mensagem, ou ao criar um clone, ou quando um parâmetro (como cronômetro, áudio ou vídeo) for maior que outro ou quando a imagem de fundo for alterada.
Raciocínio Lógico	Uso da condicional "*se*"	Uso da condicional "*se... senão*".	Uso de operações lógicas (e, ou, não).
Sincronização	Uso do comando "*espere*".	Uso de algum dos comandos: "*envie mensagem a todos*", "*quando receber mensagem*" ou "*pare todos*".	Uso de algum dos comandos: "*espere até*", "*quando o pano de fundo mudar*" ou "*envie mensagem a todos e espere*".
Interação com usuário	Usar o comando "*quando clicar na bandeira verde*".	Uso de sensores de tecla pressionada, objeto clicado ou mouse, ou uso do comando "*pergunte e espere*".	Verificar as interfaces de cronômetro, áudio ou vídeo comparando-as com um parâmetro.
Representação de dados	Uso de modificadores de propriedades dos objetos.	Uso de variáveis nas operações.	Uso de listas de operações.

Fonte: Adaptado de Barcelos, Bortoletto e Andriolli (2016) e Teixeira (2017).

Vale ressaltar que as rubricas de avaliação apresentadas no Quadro 1.4 podem ser utilizadas tanto numa perspectiva quantitativa quanto numa perspectiva qualitativa. Além do método de Brennan e Resnick (2012) e do Dr. *Scratch* destacamos o teste de PC (ROMÁN-GONZÁLEZ, 2015; ROMÁN-GONZÁLEZ; PÉREZ-GONZÁLEZ; JIMÉNEZ-FERNÁNDEZ, 2016). O objetivo é mensurar o desenvolvimento do PC de um sujeito, o foco é sobre os conceitos computacionais, como sequências, ciclos, condicionais e funções simples.

O teste de PC foi desenvolvido para mensurar o PC de alunos com idade entre 10 e 16 anos. Trata-se de um questionário composto por 28 questões fechadas, onde cada questão possui 4 alternativas e uma única resposta correta, o tempo estimado para realização do teste é de 45 minutos (ROMÁN-GONZÁLEZ; PÉREZ-GONZÁLEZ; JIMÉNEZ-FERNÁNDEZ, 2016).

[7] O recurso de clonagem permite que um ator (objeto programável) crie uma cópia de si mesmo durante a execução de um código, essa cópia é chamada de clone.

As perguntas e respostas são construídas usando imagens de setas ou blocos de programação visual para indicar a sequência de comandos ou comandos isolados. Ao responder as perguntas do teste o aluno será confrontado com situações com três tipos de objetivo: elaborar uma sequência, completar uma sequência (apontar os comandos faltantes) e depurar uma sequência (corrigir os comandos problemáticos) (ROMÁN-GONZÁLEZ; PÉREZ-GONZÁLEZ; JIMÉNEZ-FERNÁNDEZ, 2016).

Considerando esse cenário, nessa pesquisa optamos por elaborar as atividades e instrumentos de coleta de dados baseados nas dimensões do PC propostas por Brennan e Resnick (2012) e no teste de PC.

1.3 CONTRIBUIÇÕES DO PENSAMENTO COMPUTACIONAL PARA A EDUCAÇÃO

O PC não se restringe à ciência da computação, pois pode ajudar as pessoas a pensar como um físico ou um artista, isto é, a compreender como usar a computação para resolver problemas específicos de suas áreas, assim como elaborar novos questionamentos. Desta forma, o PC pode ser usado para auxiliar a criação de novos medicamentos, modelar problemas matemáticos, proporcionar novas perspectivas para um artista, desenvolver soluções para problemas ambientais, entre outros contextos (VALENTE, 2016).

Furber (2012) argumenta que a computação é necessária para o desenvolvimento econômico e científico de um país. Valente (2016) relata que a influência da computação na economia foi um fator relevante para a inclusão da computação na Educação Básica da Inglaterra, uma das razões para essa decisão foi a falta de trabalhadores qualificados para o mercado de trabalho relacionado a esse campo.

Segundo Teixeira (2017) os problemas do século XXI farão com que surjam profissões que integram diversas áreas de conhecimento e são inexistentes atualmente. O autor sugere que a computação pode ser um meio para ligar essas áreas. Nesse sentido podemos supor que o PC é um caminho para preparar as gerações atuais e futuras para esse cenário.

Balanskat e Engelhart (2014) argumentam que o PC nos ajuda a perceber a influência das tecnologias digitais em nosso cotidiano e na sociedade, os autores também sugerem que o PC estimula competências essenciais para o século XXI como o pensamento lógico, a criatividade e a resolução de problemas.

Uma forma de desenvolver o PC é por meio da programação de computadores. Mannila *et al*. (2014) apontam alguns significados que podem ser atribuídos a programação na atualidade, são eles: meio de auto expressão e participação, instrumento criativo (porque permite projetar e criar produtos), ferramenta para desenvolver a metacognição[8] e como um tipo emergente de alfabetização.

A programação também pode ser utilizada para desenvolver entendimento sobre as potencialidades das tecnologias digitais para a exploração de diversos problemas relacionadas com múltiplas disciplinas curriculares (MANNILA *et al*., 2014). Nesse sentido a codificação da solução de um problema seria um objetivo secundário, isto é, o foco principal estaria sobre o conteúdo da disciplina que se deseja ensinar e não sobre a programação. A programação pode ser usada para desenvolver projetos multi ou interdisciplinares (TEIXEIRA, 2017).

Barr e Stephenson (2011) corroboram com o entendimento de Teixeira, em seu trabalho elaboraram um quadro com exemplos de como conceitos do PC podem ser incorporados em várias disciplinas.

Em particular na matemática apresenta os seguintes exemplos: registrar os resultados no lançamento de moedas ou dados (coleta de dados); contar ocorrências no lançamento de dados e moedas (análise de dados); representar dados por meio de conjuntos, listas ou gráficos (representação de dados); aplicar a ordem de operações para calcular expressões (decomposição de problemas); usar variáveis, identificar informações essenciais para a solução de um problema, estudar funções e resolver problemas de forma iterativa (abstração); fazer divisões longa fatorando um número (algoritmos e procedimentos); usar ferramentas digitais ou analógicas para programar algoritmos de solução (automação); resolver sistemas lineares e multiplicar matrizes (paralelização); explorar o gráfico de uma função no plano cartesiano modificando os valores das variáveis envolvidas (simulação) (BARR; STEPHENSON, 2011).

Barr e Stephenson (2011) também propõem algumas estratégias pedagógicas para estimular o PC na cultura escolar, são elas: usar com frequência vocabulários ligados ao PC para descrever problemas e soluções, entender tentativas fracassadas de solução como um caminho para se obter um bom resultado, usar explicitamente a decomposição, abstração, negociação e construção de consenso em trabalhos em equipe.

[8] Refere-se ao ato de refletir e compreender o próprio processo de aprendizagem, isto é, aprender a aprender.

Quanto à negociação os autores sugerem que as equipes sejam compostas por subgrupos que explorem partes menores do problema e depois trabalhem em conjunto para unir essas partes e solucionar o problema. Em relação à construção de consenso, os autores sugerem que a elaboração da solução conjunta pode estimular solidariedade entre os envolvidos (BARR; STEPHENSON, 2011).

Considerando a Matemática em particular, observamos que o uso de abstrações, lógica e a elaboração de algoritmos são comuns tanto ao desenvolvimento do PM quanto do PC (BARCELOS; SILVEIRA, 2012). Teixeira (2017) sugere que o PC pode servir como aplicação da Matemática e incentivo para os alunos se interessarem pela disciplina, nesse sentido, os professores de Matemática podem contribuir para o ensino do PC, desenvolvendo competências comuns aos dois campos de conhecimento.

Para além disso, Barcelos e Silveira (2012) correlacionaram as competências previstas pelos Parâmetros Curriculares Nacionais (PCN) do Brasil para a disciplina de Matemática no nível do Ensino Médio com algumas competências do PC. Seu trabalho apontou três competências que desenvolvem tanto competências matemáticas como competências relacionadas ao desenvolvimento do PC, sendo elas: articulação de símbolos, identificação de padrões e regularidades e construção de modelos representativos e explicativos.

Quanto à articulação dos símbolos e códigos os autores relatam que representar um problema como um algoritmo pode auxiliar os alunos a transpor informações representadas na linguagem materna para a linguagem Matemática, podendo contribuir para a compreensão da linguagem Matemática (BARCELOS; SILVEIRA, 2012).

Sobre a identificação de padrões e regularidades os autores destacam que para que os alunos estabeleçam padrões e outras relações Matemáticas eles devem ser estimulados a assumir uma postura exploratória (BARCELOS; SILVEIRA, 2012). Podemos correlacionar essa postura com as competências necessárias para o desenvolvimento do PC propostas por Brennan e Resnick (2012).

No tocante aos modelos explicativos e representativos, os autores apontam que *softwares* podem ser utilizados para explorar e validar conceitos e objetos matemáticos, além disso podem tornar esse processo dinâmico por meio da alteração de parâmetros dentro das próprias ferramentas. Barcelos e Silveira (2012) sugerem que os alunos devem ser estimulados

a elaborar seus próprios modelos matemáticos, isso pode ser feito através da modelagem matemática.

Vale ressaltar que a ênfase do PC e da programação como caracterizações da ciência da computação tem sido criticada por alguns pesquisadores (VALENTE, 2016). Hemmendinger (2010) argumenta que a proposta do PC vinculada à computação soa arrogante, segundo ele essa ideia transmite a mensagem de que cientistas da computação estão determinando como pessoas de outras áreas devem pensar. Entretanto, o autor ressalta que os cientistas da computação podem contribuir com outras áreas, e, sugere que talvez devêssemos concentrar mais esforços no fazer computacional que em discussões sobre o PC.

Denning (2009) entende que as descrições de PC remetem a práticas que são amplamente utilizadas por cientistas no geral, não se configurando como uma contribuição única da computação. O autor também argumenta que o PC é uma prática e não uma descrição dos princípios subjacentes da computação.

As críticas propostas por Hemmendinger (2010) e Denning (2009) não possuem grande relevância para esta pesquisa, visto que nosso entendimento de PC contempla também práticas computacionais que podem ser vistas como próximas ao fazer computacional, e nosso foco é a formação inicial de professores de Matemática. Essa formação prepara profissionais para atuar na Educação Básica, onde poderão executar abordagens sobre a Matemática orientadas pelo PC, sem a necessidade de uma correlação direta com a computação.

Além disso, entendemos que similar à forma com que o ensino de teatro nas escolas não implica que todos os estudantes serão atores, inserir o PC na educação não significa que desejamos preparar todos os discentes para serem programadores, mas sim, proporcionar a estes estudantes uma melhor compreensão sobre aspectos de seu cotidiano relacionados as tecnologias digitais.

1.4 ESTUDOS SOBRE O PENSAMENTO COMPUTACIONAL NA EDUCAÇÃO BÁSICA

Atualmente há esforços em todo o mundo para integrar habilidades relacionadas ao PC em currículos novos ou existentes, esse processo exige que diferentes sujeitos (professores, coordenadores, direção, estudantes de diversas faixas etárias, comunidade escolar, entre outros)

sejam preparados para lidar com os desafios relacionados ao desenvolvimento e aprendizagem dessa competência (KALELIOĞLU, 2018).

Nesse sentido, estudos vêm sendo desenvolvidos para buscar meios de integrar o PC ao cotidiano escolar. Dentre os levantamentos bibliográficos existentes, encontramos um estudo sobre os benefícios da programação com ênfase no *Scratch* para o currículo K-12 (MORENO-LEÓN; ROBLES, 2016), um trabalho que discute os resultados de pesquisas recentes sobre o PC (MORENO-LEÓN; ROMÁN-GONZÁLEZ; ROBLES, 2018) e artigos que investigam as relações entre o PC e a Matemática (BARCELOS; SILVEIRA, 2012; MESTRE *et al.*, 2015; BARCELOS *et al.* 2018a; BARCELOS *et al.* 2018b). No entanto, nessa seção pretendemos explorar alguns trabalhos que abordam as tendências temáticas das pesquisas sobre PC a nível internacional e nacional.

Em relação a estudos internacionais, destacamos a pesquisa desenvolvida por Kalelioğlu (2018) que teve como objetivo examinar artigos sobre PC publicados entre 2013 e 2016 nas bases de dados IEEE *Xplore Digital Library* ou *ScienceDirect*. O autor analisou ano, público-alvo, palavras-chaves, foco e objetivo da publicação, além de características relacionadas à natureza dos estudos (método de pesquisa, instrumentos de coleta de dados e método de análise de dados). Ao total foram analisados 65 artigos.

Quanto o público-alvo Kalelioğlu (2018) observou que os estudos foram realizados com qualquer grupo diretamente relacionado ao PC, a população mais investigada foi a de estudantes da graduação, seguida de estudantes da Educação Básica e em terceira posição professores.

No que tange as palavras-chave o autor constatou que “PC” é o termo mais utilizado, seguido de “programação” e em terceiro lugar “estratégias de ensino e aprendizagem”. Kalelioğlu (2018) acredita que “programação” é o segundo termo mais utilizado porque usualmente essa é a estratégia mais usada para desenvolver PC.

No tocante ao foco dos estudos o autor identificou cinco categorias: modelo, pedagogia, avaliação, pesquisa e opinião. A categoria modelo abrange trabalhos focados em discussões sobre o PC e seu escopo. Pedagogia inclui artigos que buscam explorar como o PC pode ser ensinado. Avaliação engloba estudos que procuram explorar como o PC pode ser avaliado e aprendido. Pesquisa reúne trabalhos que investigam as metodologias de pesquisa

empregadas e opinião apresenta artigos que discutem perspectivas sobre o PC (KALELIOĞLU, 2018).

Kalelioğlu (2018) estabeleceu oito categorias em relação ao objetivo das pesquisas, são elas: desenvolver habilidades do PC ou de programação em alunos, propor um método de ensino ou atividade para desenvolver o PC, investigar o efeito de variáveis sobre o conhecimento de PC ou conceitos de ciência da computação, avaliar o PC, desenvolver um currículo de PC, revisar a literatura sobre PC ou computação, discutir os impactos do PC no ensino de ciências da Educação Básica (currículo K-12) e relatar percepções ou experiências de professores sobre a ciência da computação.

Em relação ao método de pesquisa Kalelioğlu (2018) destacou que o método mais utilizado foi o estudo de caso, seguido de pesquisas experimentais. Levantamentos, pesquisas qualitativas e quase-experimentos também foram amplamente utilizados. No que se refere aos instrumentos de coletas de dados o autor apontou que o uso de questionários foi comum dentro da amostra, seguido de testes realizados pelo pesquisador, entrevistas e observações (KALELIOĞLU, 2018).

Por fim, no que tange aos métodos de análise de dados Kalelioğlu (2018) ressalta que para dados quantitativos a análise estatística descritiva foi a mais utilizada, seguida por métodos quantitativos como teste *t*, ANOVA e ANCOVA. Em relação aos dados qualitativos a análise de conteúdo foi o método mais empregado.

Outro estudo no contexto internacional que gostaríamos de destacar é o desenvolvido por Avila *et al.* (2017). Neste artigo os autores objetivaram identificar as principais abordagens sobre o PC no âmbito do Ensino Fundamental e Médio categorizando seus objetivos, práticas pedagógicas, instrumentos utilizados ou propostos, assim como conceitos e habilidades desenvolvidos. Os autores levantaram artigos publicados entre 2012 e 2016 nas bases de dados IEEE *Xplore Digital Library*, Scopus, ACM, *ScienceDirect* e *Springer*. Ao todo 45 artigos foram analisados (AVILA *et al.*, 2017).

Sobre os objetivos de pesquisa dos trabalhos analisados, Avila *et al.* (2017) identificaram onze categorias, a saber: avaliação do PC, introdução de conceitos da computação na Educação Básica para desenvolver o PC, PC por meio da programação, PC por meio de jogos, PC por meio da robótica, intervenção ou integração do PC a outras disciplinas, inserção da computação no currículo, metodologias de implantação, intervenção com professores da

Educação Básica, percepção da comunidade escolar sobre o tema e desenvolvimento do PC por meio de outras ferramentas.

Avila *et al.* (2017) identificaram os seguintes conceitos da computação nos trabalhos analisados: programação, algoritmos, simulação, decomposição, eventos, modelagem, paralelismo, depuração, análise de dados, representação de dados, modularização, lógica booleana, ramificação, números binários, eventos, refinamento, iteração, ordenação e reutilização. Os autores relacionaram à frequência com que estes conceitos aparecem nos trabalhos com as categorias dos objetivos. O conceito mais comum foi o de programação, seguido de algoritmos e simulação.

Além disso, Avila *et al.* (2017) identificaram as seguintes habilidades do PC nos artigos analisados: pensamento algorítmico[9], abstração, resolução de problemas, colaboração, pensamento lógico[10], simulação e generalização. Os autores também associaram a frequência dessas habilidades com as categorias dos objetivos. A habilidade mais frequente foi a abstração, seguida do pensamento algorítmico e da resolução de problemas.

Os autores destacam que há carência de um modelo conceitual comum sobre o PC. A ausência desse modelo ocasiona isolamento entre os trabalhos que propõem metodologias para o ensino do PC ou sua inserção no currículo, visto que os estudos partem de entendimentos diferentes sobre o que é PC e quais habilidades caracterizam esse tipo de pensamento (AVILA *et al.*, 2017). Moreno-León, Román-González e Robles (2018) também destacam que a falta de consenso sobre a definição de PC se configura como um desafio para avançar as pesquisas nessa área.

No que tange a estudos nacionais, Bordini *et al.* (2017) correlacionam os objetivos identificados na literatura internacional com os objetivos de estudos realizados no Brasil. Segundo os autores:

> Muitos objetivos de pesquisa encontrados neste levantamento são semelhantes aos encontrados no Brasil [Bordini *et al.* 2016b], tais como: atrair a atenção para a área da computação, alguns em especial das meninas; utilizar linguagens de programação

[9] O pensamento algorítmico engloba análises sobre o propósito de um algoritmo e os possíveis caminhos para construí-lo ou aplicá-lo (FUTSCHEK, 2006), consiste numa atividade criativa e reflexiva ao invés de um exercício de pura memorização.

[10] O pensamento lógico não é caracterizado como um tipo de PM porque é entendido como um elemento intrínseco ao próprio PM, visto que é essencial na complexa composição da estrutura formal da Matemática (COLOMBIA, 2006).

> visual em experiências de ensino de programação; e buscar formas de avaliar o desenvolvimento de habilidades do PC (BORDINI *et al*, 2017, p. 8).

Avila *et al.* (2016) buscaram levantar artigos publicados entre 2010 e 2015 que objetivassem disseminar o PC no Brasil. Nesse trabalho analisaram os objetivos das pesquisas, o público-alvo e número de participantes, os instrumentos adotados para desenvolver ou avaliar o PC e o resultados reportados por estas produções.

Os trabalhos foram coletados de oito anais de eventos e revistas, são eles: Simpósio Brasileiro de Informática na Educação (SBIE), Workshop de Informática na Escola (WIE), Workshop de Ensino em Pensamento Computacional, Algoritmos e Programação (WAlgProg), Revista Brasileira de Informática na Educação (RBIE), Workshop sobre Educação em Computação (WEI), Workshop Escola de Informática Teórica (WEIT), Revista Novas Tecnologias na Educação (RENOTE) e Workshop de Desafios da Computação Aplicada a Educação (DesafIE). Ao todo 45 artigos foram analisados.

Quanto ao público-alvo, Avila *et al.* (2016) identificaram que os experimentos tinham como sujeitos alunos do 1º ano e do 4º a 9º anos do Ensino Fundamental, bem como todos os anos do Ensino Médio, alunos de cursos técnico em informática e graduandos em Ciência da Computação.

No que se refere aos instrumentos ou atividades empregadas para desenvolver o PC os autores identificaram que o *Scratch* foi a ferramenta mais utilizada, seguido da computação desplugada. Segundo Avila *et al.* (2016) na maioria das vezes o *Scratch* era usado para desenvolver conceitos fundamentais da Computação, sendo implementado por meio de oficinas. Já a computação desplugada era usada para abordar conceitos básicos da computação, como números binários ou representação de imagens.

Sobre os resultados, os autores destacam que a maioria dos trabalhos reportam resultados positivos sobre a efetividade de suas ações em relação ao PC, indicando que os estudantes apresentaram desempenho satisfatório nas atividades e avaliações propostas. No entanto, Avila *et al.* (2016) relatam que apenas dois trabalhos não demonstraram melhoria no desempenho nos testes realizados após as atividades.

Sobre as teses e dissertações publicadas no Brasil destacamos o estudo desenvolvido por Corrêa, Grossi e Pereira (2018). Nesse artigo os autores identificam quais tendências temáticas e recursos pedagógicos têm sido utilizados nas dissertações e teses, defendidas entre

2014 e metade de 2018, que investigaram o PC na Educação Básica. O banco de dados utilizado foi o Catálogo de Teses e Dissertações da CAPES, ao todo trinta trabalhos foram analisados.

Quanto as tendências temáticas das pesquisas, os autores apontam sete categorias, sendo elas: elaboração de recursos pedagógicos ou estratégias para desenvolver o PC, ensino de computação, avaliação, aprendizagem, resolução de problemas matemáticos para desenvolver o PC, problematização do PC na educação e percepção de professores sobre o PC (CORRÊA; GROSSI; PEREIRA, 2018).

No que tange às estratégias ou recursos pedagógicos utilizados, os autores identificaram seis categorias, sendo elas: atividades desplugadas, ambientes visuais de programação baseados em blocos, linguagens textuais de programação, programação de objetos pertencentes ao mundo físico, jogos analógicos ou digitais e situações-problemas (CORRÊA; GROSSI; PEREIRA, 2018).

Corrêa, Grossi e Pereira (2018) destacam que há poucas teses e dissertações que problematizam o PC na educação, desenvolvam instrumentos de avaliação ou discussões sobre a avaliação do PC, e proponham atividades desplugadas para o desenvolvimento do PC. Essas lacunas são apontadas como temas de pesquisa a serem explorados.

Em particular, destacam que há uma significativa escassez de estudo que abordem o PC na formação continuada ou inicial de professores (CORRÊA; GROSSI; PEREIRA, 2018). Barcelos (2014) também destaca a necessidade de realizar estudos sobre o Pensamento Computacional na formação continuada ou inicial de professores de Matemática.

Comparando os estudos explorados nessa seção percebemos que os focos temáticos a nível internacional e nacional estão alinhados, visto que as categorias propostas por Kalelioğlu (2018), Avila *et al.* (2016, 2017) e Corrêa, Grossi e Pereira (2018) são semelhantes. Em relação aos recursos empregados para desenvolver o PC também percebemos alinhamento, as categorias elaboradas por Corrêa, Grossi e Pereira (2018) corroboram com os resultados obtidos por Moreno-León, Román-González e Robles (2018).

1.5 PENSAMENTO COMPUTACIONAL NA FORMAÇÃO DE PROFESSORES

A introdução de PC nos cursos de formação de professores é um campo de pesquisa a ser explorado. Segundo Gretter e Yadav (2016), apesar da existência de alguns estudos sobre essa temática, investigações sobre como promover o PC na formação de professores são oportunas e necessárias. Barcelos, Bortoletto e Andrioli (2016) apontam que a adequação da formação de professores para trabalhar o PC ainda é um desafio. Consideramos que há poucos estudos sobre essa temática no contexto da educação Brasileira, entendemos que a presente pesquisa vem ao encontro dessa necessidade.

Na seção 1.4 apresentamos um panorama das pesquisas sobre PC na educação básica, nessa seção centramos nossa atenção sobre alguns desafios e perspectivas específicos da formação de professores. Discutiremos também alguns estudos que estão diretamente relacionados à introdução do PC na formação inicial ou continuada de professores.

O público jovem que frequenta as escolas atualmente, em sua maioria, está acostumado a ser estimulado pelos dispositivos digitais diariamente, habituados com a praticidade e velocidade que esses recursos proporcionam (MANDAJI *et al.*, 2018). Esses sujeitos muitas vezes demonstram facilidade em aprender a usar esses recursos, porém podem apresentar dificuldades para se expressar usando esses recursos ou criar artefatos digitais (RESNICK *et al.*, 2009).

Segundo Arruda (2013) a escola deve formar os indivíduos “nas e para as mídias”, uma vez essas são portadoras de significados, cultura e conteúdos apreendidos pelas pessoas. O autor também aponta que a formação de professores deve ser repensada de modo a lhes permitir incorporar, compreender e estabelecer relações críticas com as tecnologias digitais. Entendemos que a formação de professores deve prepará-los para compreender as implicações sociais e psicológicas das tecnologias digitais no processo de ensino e aprendizagem (BELINE; COSTA, 2010).

Entretanto, muitas vezes esses recursos são abordados apenas de forma teórica nos cursos de formação inicial, sem atividades práticas que explorem seus potenciais (GATTI; NUNES, 2009). Em relação à formação continuada, Paz (2017) relata que os cursos sobre informática na educação costumam abordar as tecnologias digitais como um fim, ao invés de um meio, ou seja, como instrumentos que apoiarão o processo de ensino e aprendizagem.

Considerando a licenciatura em Matemática, observamos que usualmente os cursos trabalham com as tecnologias digitais numa abordagem utilitária (NAITO *et al.*, 2011). Gatti e Nunes (2013) relatam que, no Brasil, as Diretrizes Curriculares para as licenciaturas em Matemática preveem que os futuros professores devem se familiarizar com o uso do computador e de outras tecnologias para o ensino de Matemática, com ênfase na resolução de problemas.

Desenvolver o PC é um meio para aprimorar a resolução de problemas e lógica (MANDAJI *et al.*, 2018), nesse sentido atividades que busquem desenvolver o PC podem ser um meio de explorar as tecnologias digitais de forma prática, ampliar o domínio e compreensão que os professores tem sobre esses recursos, ao mesmo tempo em que se fornece exemplos de como utilizá-los para desenvolver habilidades de resolução de problemas.

Manilla *et al.* (2014) argumentam que a integração do PC só será bem-sucedida quando os professores entenderem e forem capazes de perceber sua relevância para as disciplinas que ensinam. Os autores sugerem que os cursos de formação visem esclarecer os conceitos relacionados ao PC, forneçam bons exemplos de atividades que usem o PC e apoiem os professores na criação e implementação de atividades dessa natureza.

O National Research Council (2011) aponta que o principal desafio para os professores é colocar o interesse dos estudantes no centro das atividades a serem desenvolvidas, visto que em abordagens nas quais os discentes resolvem e propõem problemas os professores tendem a perder o controle que tradicionalmente têm sobre a turma e o processo de ensino e aprendizagem.

Nesse sentido, os professores podem precisar assumir papéis que desconhecem em sala de aula, para isso precisam de um ambiente onde possam aprender junto com seus estudantes e, para tanto, necessitam do apoio dos gestores e colegas de trabalho para entrar em situações onde eles não dominam (MANNILA *et al.*, 2014).

Em relação às pesquisas relacionadas à inserção do PC na formação de docentes, destacamos no contexto internacional os trabalhos de Yadav *et al.* (2014), Mugayitoglu (2016), Teixeira (2017) e Angeli e Jaipal-Jamani (2018).

Yadav *et al.* (2014) introduziram um módulo sobre PC em um curso de formação de professores. O módulo objetivou apresentar ideias sobre computação e discutir como essas ideias podem ser utilizadas em suas futuras carreiras.

Os autores relatam que o módulo ajudou os professores em formação a perceber o PC como uma forma de resolução de problemas. Os participantes também entenderam que podem ensinar conceitos de computação na Educação Básica (no caso, no currículo K-12) sem fazer uso de computadores, e que o PC pode ser integrado a todas as disciplinas (YADAV *et al.*, 2014).

A tese de abordagem quantitativa, elaborada por Mugayitoglu (2016), teve por objetivo examinar se a introdução de atividades envolvendo o PC em cursos de formação inicial de professores mudam suas atitudes e compreensões sobre este tema. Para isso o autor implementou uma unidade de ensino sobre PC, utilizando o *Scratch* e atividades desplugadas.

Mugayitoglu (2016) constatou que a intervenção foi eficaz, todos os professores apresentaram aumento de atitudes positivas em relação ao PC ao final do estudo. O autor ressalta que o uso de atividades desplugadas e do *Scratch* permitiram que os professores desenvolvessem suas próprias habilidades de PC, bem como adquirissem experiência na implementação de conceitos e práticas de PC.

Em Portugal, Jaylson Teixeira produziu uma tese com o objetivo de "avaliar as influências das sucessivas práticas adotadas no desenvolvimento do pensamento computacional em um ambiente de aprendizagem para futuros professores de matemática" (TEIXEIRA, 2017, p. 26). Seu trabalho resultou num modelo adaptável de ensino baseado em roteiros de atividades, provas de conhecimentos e projetos de programação no *Scratch*.

Para a elaboração dos projetos Teixeira (2017) propôs também o uso de *storyboards*. Um *storyboard* é um documento para auxiliar o planejamento e acompanhamento dos projetos. Segundo o autor esse recurso se mostrou útil para auxiliar o processo de desenvolvimento de projetos, além de estimular independência e capacidades metacognitivas dos alunos.

Angeli e Jaipal-Jamani (2018) desenvolveram um estudo para analisar como atividades envolvendo robótica influenciam o PC de professores formação inicial. Para desenvolver as atividades usaram o kit LEGO WeDo. Os autores apontam que os participantes apresentaram um desenvolvimento significativo de habilidades do PC mesmo sem ter

experiências prévias com esse tipo de pensamento. Além disso, sugerem que atividades que visem o desenvolvimento do PC podem ser inseridas na formação de professores sem a necessidade de reestruturar os programas dos cursos de formação.

Existem diversos outros trabalhos que abordam a integração do PC na formação de professores no contexto internacional, Valente (2016) apresenta e discute outras pesquisas realizadas nesse sentido. No contexto nacional destacamos a pesquisa de Prado e Costa (2016), Barcelos, Bortoletto e Andrioli (2016) e Mandaji *et al.* (2018).

Prado e Costa (2016) apresentam um artigo que mostra a análise de duas atividades envolvendo programação e conceitos matemáticos aplicadas na formação continuada de professores de Matemática. As autoras fazem uso do termo "episódio" para se referir a cada uma dessas atividades. O trabalho teve como objetivo "compreender as possibilidades das atividades de programação utilizadas nos episódios, em relação à (re)construção de conhecimentos necessários para a docência com tecnologia" (PRADO; COSTA, 2016, p. 898).

Na primeira atividade as autoras utilizaram o *software* Geogebra e na segunda o *Scratch*. Segundo Prado e Costa (2016) as atividades contribuíram para que os professores explorassem as tecnologias digitais para além do domínio operacional. O *Scratch* possibilitou que os participantes vivenciassem o processo de criação de um programa educacional, propiciando a reconstrução de conhecimentos pedagógicos tecnológicos sobre o conteúdo matemático.

Barcelos, Bortoletto e Andrioli (2016) analisaram o desenvolvimento e aplicação de um curso online voltado para a formação, inicial e continuada, de professores de Matemática. A aplicação relatada nesse artigo foi um experimento piloto do curso. O curso foi realizado na plataforma *Moodle* e trabalhou o PC e conceitos matemáticos por meio do desenvolvimento de jogos digitais.

Segundo os autores, os resultados indicaram que os participantes apresentaram um maior envolvimento em atividades que eram diretamente relacionadas à construção de jogos digitais, além disso, os projetos finais apresentaram indícios de domínio intermediário ou avançado do PC (BARCELOS; BORTOLETTO; ANDRIOLI, 2016).

Mandaji et. al. (2018, p. 1) elaboraram um artigo com o objetivo de "apresentar uma ação de formação de professores que teve como ponto de partida a imersão dos docentes no

pensamento computacional de forma a integrar a linguagem de programação em atividades plugadas e desplugadas ao currículo escolar". Essa ação faz parte do movimento Programaê!, fruto de uma parceria entre a Fundação Lemann e a Telefônica Vivo, esse movimento visa disseminar conhecimentos de computação e programação para crianças, jovens e professores.

Neste artigo os autores relatam os resultados de uma oficina que foi aplicada com o propósito de fornecer subsídios para a introdução da programação e PC nas práticas pedagógicas dos professores participantes. Mandaji *et al.* (2018) apontam que a oficina possibilitou que os participantes percebessem que o PC pode ser trabalhado em diversas disciplinas curriculares, e que mesmo sujeitos que nunca tiveram contato anteriormente com programação são capazes de aprender a programar.

Diante dessas pesquisas percebemos que a comunidade científica está buscando soluções para integrar o PC na Educação Básica. Em nossa visão o PC é fundamental para a século XXI, tendo em vista a alta dependência que diversos ramos da sociedade têm em relação às tecnologias digitais. A atual versão da BNCC menciona o PC apenas no componente Matemática, entendemos que essa competência deve ser trabalhada de forma transversal não se restringindo apenas a uma das áreas de conhecimento.

Por fim, destacamos que ainda há muitos desafios a serem superados como ampliar ou melhorar a infraestrutura das escolas, preparar a equipe escolar para desenvolver o PC, estabelecer objetivos de aprendizagem claros e desenvolver instrumentos de avaliação para as diferentes etapas da Educação Básica. No capítulo seguinte discutimos o Construcionismo de Papert e o *software Scratch*.

CAPÍTULO 2 INTRODUÇÃO AO *SCRATCH*

Ao final da década de 1960 Seymour Papert, um educador e matemático, desenvolveu uma linguagem de programação visual chamada Logo para incentivar crianças a aprender a programar. Após quase meio século depois que a "tartaruga gráfica" de Papert foi apresentada ao mundo, Resnick *et al*. (2009) criaram um ambiente virtual de programação por blocos chamado *Scratch*, tomando como base a linguagem Logo, o Construcionismo proposto por Papert e o ambiente *Etoys*[11].

O *Scratch* pode ser considerado uma ferramenta para o ensino e aprendizagem de matemática (CURCI, 2017), além de estimular o desenvolvimento da fluência tecnológica (CURCI, 2017) e PC (TEIXEIRA, 2017) dos estudantes.

Neste capítulo apresentamos a teoria de aprendizagem que foi utilizada como base para o desenvolvimento do *Scratch*, assim como algumas características gerais desse *software* e suas contribuições para a educação.

2.1 INTRODUÇÃO AO CONSTRUCIONISMO

Os primeiros relatos do uso de computadores na educação são da década de 1920. Nessa época Sidney Pressey produziu uma máquina para automatizar a correção de exercícios de múltipla escolha (BRESSAN, 2016). Em 1930 foram desenvolvidas máquinas de ensinar, essas máquinas eram baseadas na instrução programada e utilizadas como recursos auxiliares nos processos de ensino-aprendizagem (SOUSA; FINO, 2008).

A instrução programada consiste na organização de diversas informações em módulos, os quais são estudados pelos alunos que ao final de cada módulo devem responder um questionário sobre a temática estudada (BRESSAN, 2016). Se o discente acertar as respostas ele prossegue para o módulo seguinte, caso erre é informado sobre as respostas corretas e convidado a estudar novamente o mesmo módulo ou outros módulos relacionados a mesma temática (VALENTE, 1998). Esse método de ensino pode ser implementado usando livros, computadores, entre outros meios.

[11] Trata-se de um ambiente de programação orientada a objetos voltado para crianças.

O auge da instrução programada ocorreu na década de 1980 com a inserção de computadores em centros educacionais que rapidamente foram utilizados para promover instruções programadas (VALENTE, 1998). Segundo Valente (1998), quando o computador é utilizado para informatizar e replicar os meios tradicionais de ensino, para complementar ou reforçar o ensino em sala de aula, ele está sendo usado numa perspectiva instrucionista.

Em 1967 Seymour Papert e sua equipe criaram a primeira versão da linguagem de programação Logo (LOGO FOUNDATION, 2015). Em 1980 o livro *Mindstorms* (PAPERT, 1980) foi publicado, esse livro contribuiu para popularizar o uso do Logo na educação. Nessa mesma década Papert propôs um modelo de ensino denominado Construcionismo como alternativa ao instrucionismo que era amplamente utilizado na época.

No Construcionismo os estudantes são convidados a criar seus próprios projetos, explorar estratégias, se comunicar com o computador por meio de uma linguagem de programação e refletir sobre seus erros. As ideias de Papert (1985) e a linguagem Logo abriram novos caminhos para o uso da informática na educação. Porém, isso não significa que a instrução programada ou as máquinas de ensinar tenham desaparecido, pode-se dizer que elas ainda são amplamente utilizadas na educação, embora atualmente sejam implementadas por meio de *softwares* mais avançados.

2.1.1 Introdução ao Instrucionismo

No instrucionismo o computador é usado para reforçar ou complementar os conhecimentos abordados em sala de aula. Nessa abordagem o professor determina quais informações o computador deverá apresentar ao discente, em que ordem essas informações deverão ser fornecidas e como o aluno será testado. O discente, por sua vez, deve interagir com o computador para estudar toda a informação que lhe for apresentada e ao final ser testado (VALENTE, 1998). A Figura 2.1 apresenta o modelo de ensino instrucionista.

Figura 2.1: Modelo instrucionista.

Fonte: (BRESSAN, 2016).

Nesse modelo, o processo de ensino-aprendizagem é centrado no professor que determina o que será estudado, como será estudado, como será avaliado e repassa essas informações ao computador. Assim, os alunos assumem papéis passivos e devem interagir com o computador para obter e memorizar informações que posteriormente serão utilizadas para responder perguntas avaliativas. Esse modelo se assimila à educação bancária segundo Freire (2011). O autor destaca que a educação bancária é inadequada para o desenvolvimento de autonomia do discente.

Como o computador acaba sendo utilizado como instrumento para informatizar processos tradicionais de ensino, não há quebra na rotina escolar, desta forma não é preciso investir na formação de professores, basta apenas realizar cursos sobre a utilização de determinados *softwares* (VALENTE, 1997). O modelo instrucionista é caracterizado pela ausência de estímulos à autonomia dos educandos.

Mugayitoglu (2016) ressalta que o instrucionismo se baseia na memorização e que essa estratégia pode ser eficiente para ensinar conceitos e fundamentos essenciais das áreas de conhecimento. Segundo o autor, uma vez memorizados esses conhecimentos podem ser utilizados para construir aprendizados significativos. Um exemplo disso é a memorização do significado de palavras de uma segunda língua, para posteriormente utilizar esse vocabulário para construir frases e sentenças (MUGAYITOGLU, 2016).

Entretanto, Valente (1997) ressalta que o instrucionismo não prepara os educandos para lidar com desafios sociais ou profissionais, além disso, devido à ausência de incentivo à formação continuada esse modelo instiga os professores a continuar utilizando as mesmas técnicas e aulas. Moran (1998) esclarece que a crítica ao instrucionismo não é, necessariamente, à utilização de computadores e outros recursos digitais na escola, mas sim, à forma com que esses recursos são abordados.

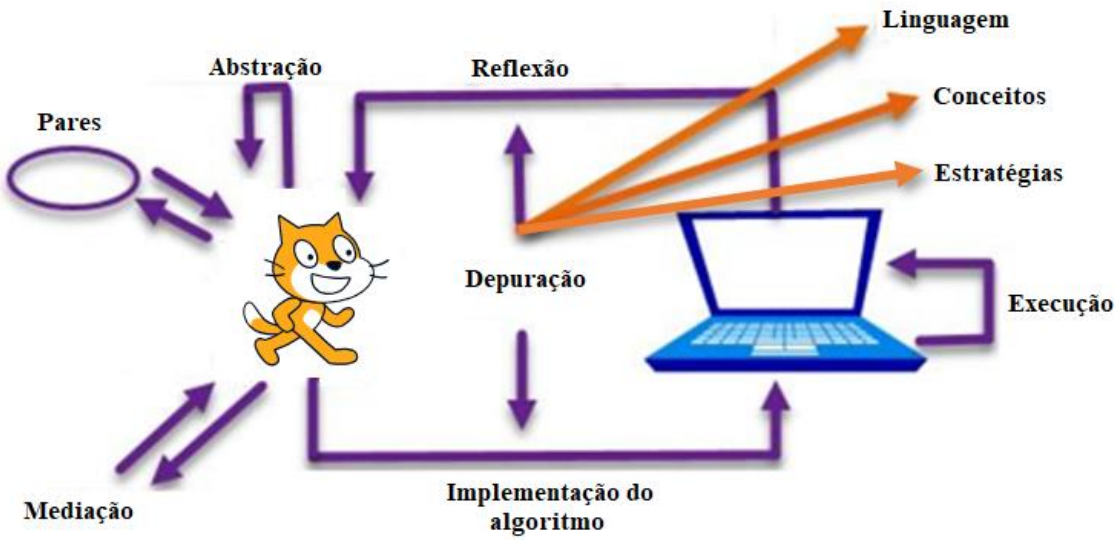
Papert acreditava que os computadores poderiam contribuir para o desenvolvimento da autonomia dos educandos por este motivo o autor propôs o Construcionismo (BRESSAN, 2016; CURCI, 2017), esse modelo será apresentado e discutido na próxima subseção.

2.1.2 Caracterização do Construcionismo

Diferentemente do instrucionismo, no Construcionismo os discentes assumem um papel ativo em seu processo de aprendizagem (BRESSAN, 2016). Nesse modelo apresenta-se um problema ao estudante e, este deve elaborar uma estratégia de solução na forma de um algoritmo computacional usando alguma linguagem de programação (SILVA *et al.*, 2016). O computador deixa de ser utilizado como meio para transmitir informações e passa a ser tratado como um artefato que precisa ser "ensinado" a executar uma determinada tarefa.

Segundo Valente (1997) o termo Construcionismo foi utilizado por Papert para remeter a possibilidade de construir o conhecimento a partir da produção de artefatos como um programa de computador. Papert recomenda que os artefatos produzidos estejam relacionados aos interesses pessoais dos discentes. Na Figura 2;2 apresentamos o modelo de ensino-aprendizagem construcionista.

Figura 2.2 - Modelo construcionista.



Fonte: Adaptado de Bressan (2016).

No Construcionismo a aprendizagem acontece de forma cíclica. Para melhor entender esse modelo suponha que o discente entrou em contato com o problema, elaborou um algoritmo para solucioná-lo (algoritmo inicial) e o *implementou* usando um computador. Na sequência o computador *executa* esse algoritmo enquanto o educando observa a execução e os resultados obtidos. Com base nas observações realizadas durante a execução do algoritmo os estudantes refletem se o problema foi resolvido ou não. Essas *reflexões* podem produzir diferentes níveis de *abstração* no discente, a saber: empírica, pseudo-empírica e reflexiva (VALENTE, 1997).

Na abstração empírica o educando extrai informações simples sobre um objeto, como cor ou forma, ou informações relacionadas a ações realizadas sobre o objeto. A abstração pseudo-empírica vai além da extração de informações, nesse nível o discente consegue deduzir algum conhecimento a partir de características do objeto ou das ações que realizou sobre o objeto. A abstração reflexiva é um nível mais elevado em relação as demais. Nela o discente reorganiza o conhecimento deduzido com base em conhecimentos prévios, pode-se dizer que na abstração reflexiva o educando modifica suas próprias ideias a partir daquilo que consegue perceber (VALENTE, 1997).

O processo de reflexão pode chegar a duas conclusões: o problema foi resolvido ou não foi resolvido. Se o problema foi resolvido o educando não precisa modificar o algoritmo inicial, pois sua estratégia foi eficaz. Se o problema não foi resolvido o estudante precisa iniciar um processo de *depuração* (VALENTE, 1997).

Conforme Selby (2014) depurar é uma estratégia de resolução de problemas que parte do sintoma para encontrar e corrigir o erro. A autora destaca que ao depurar os discentes não corrigem apenas erros em sua própria lógica, ou na estrutura do código que produziram, eles também começam a entender o modo que o computador executa um programa.

Valente (1997) aponta que a depuração pode acontecer em relação à *linguagem* de programação (erro lógico entre comandos), sobre um *conceito* relacionado ao problema (o discente desconhece ou não entende o conceito) ou ainda sobre as *estratégias* empregadas (o estudante não conhece ou não sabe implementar técnicas para resolução de problemas).

Assim, o processo de depuração pode levar o estudante a analisar seu programa em termos de coerência lógica dos comandos, efetividade das ideias e estratégias empregadas, podendo fazer com que o discente pense sobre suas próprias ideias, isto é, analisar seu

pensamento em um nível metacognitivo (VALENTE, 1997). Esse processo acontece por meio da *mediação* do professor e a *interação com seus pares* (BRESSAN, 2016).

Segundo Papert (2008), o Construcionismo se baseia no pressuposto de que o sujeito aprende quando encontra os conhecimentos específicos que necessita e dessa forma seus esforços são recompensados intelectualmente e psicologicamente. Além disso, Curci (2017) aponta dois processos essenciais do conhecimento no Construcionismo: a construção dos projetos e seu compartilhamento (processo externo) e a reflexão sobre experiências originadas na interação com o meio (processo interno).

Papert (2008) aponta que dois tipos de conhecimento são construídos ao se utilizar o computador numa perspectiva construcionista: o conhecimento matemático e o matético. Para o autor o termo "matética" significa a arte de descobrir como se aprende, assim o conhecimento matético está relacionado à consciência que o sujeito desenvolve sobre seu próprio processo de aprendizado (metacognição), bem como ao desenvolvimento da autonomia do discente.

Os interesses pessoais dos estudantes são reconhecidos e valorizados no Construcionismo, pois eles podem ser utilizados para elaborar problemas que estimulem os sujeitos a mobilizar sua imaginação, raciocínio lógico e pensamento crítico (PAPERT, 2008). Nesse sentido, os educadores precisam planejar ambientes de aprendizagem que estimulem a criatividade e instiguem os educandos a explorar e experimentar (BRESSAN, 2016).

No Construcionismo o estudante deixa de ser um receptáculo de informações e passa a ser produtor de conhecimentos (BRESSAN, 2016). Isso pode gerar mudanças na rotina escolar visto que o papel dos discentes e docentes diverge do tradicional, assim, diferentemente do instrucionismo, o Construcionismo exige investimento na infraestrutura escolar e formação de docentes. Além disso, pode contribuir para instigar e desenvolver a autonomia dos educandos (SILVA *et al.*, 2016).

O modelo construcionista foi estruturado com base no construtivismo de Piaget. É possível perceber algumas similaridades entre essas teorias, como o raciocínio lógico-abstrato por meio da dimensão concreta, o foco do processo de ensino e aprendizado na experiência do discente e a valorização do desenvolvimento pessoal do educando (BRESSAN, 2016; MUGAYITOGLU, 2016).

Apesar de semelhantes essas teorias são distintas, assim, por isso, para melhor caracterizar o Construcionismo elaboramos o Quadro 2.1 onde diferenciamos o Construcionismo do Instrucionismo e Construtivismo. Para elaborar essa tabela recorremos ao trabalho de Curci (2017) para caracterizar o construtivismo, e os trabalhos de Valente (2017) e Mugayitoglu (2016) para caracterizar o instrucionismo e o construtivismo.

Quadro 2.1 - Comparação entre Instrucionismo, Construtivismo e Construcionismo.

Característica	**Instrucionismo (VALENTE, 1997) (MUGAYITOGLU, 2016)**	**Construtivismo (CURCI, 2017)**	**Construcionismo (VALENTE, 1997) (MUGAYITOGLU, 2016)**
Interação	Interação do discente com o computador (VALENTE, 1997).	Interação do discente com o objeto.	Interação do discente com o computador e o objeto.
Papel do professor	Determinar qual conteúdo será abordado e em que ordem deve ser estudado. Determinar como o discente será avaliado.	Observador não intervencionista.	Mediar a interação do discente com o computador e o objeto.
Papel do discente	Memorizar as informações que lhe são apresentadas. Responder aos testes corretamente. Seguir as orientações do professor.	Atuar de forma ativa no processo de construção de seu conhecimento.	Elaborar um algoritmo computacional para resolver um problema.
Intervenção	O professor deve explicitar o que os discentes devem fazer e como fazer. O direcionamento é feito de forma explícita.	Apenas analisa e tenta entender o máximo que puder sobre os processos mentais e o desenvolvimento do discente de modo clínico.	O professor intervém quando o discente não consegue avançar no desenvolvimento de seu projeto. O direcionamento é feito de forma indireta.
Socialização	Discentes não interagem entre si.	O indivíduo está isolado em sua comunidade de convívio.	O discente interage com seus pares de forma colaborativa.
Foco	Memorização das informações.	Compreensão do sujeito e das construções mentais.	Produção dos algoritmos.

Fonte: Autores.

A partir do Quadro 2.1 observamos que o Construcionismo promove a interação do estudante com o objeto de estudo e com o computador simultaneamente. Nesse modelo o docente deve mediar essa interação, além disso no Construcionismo o professor intervém na forma com que o discente aborda o problema se achar necessário. Destacamos que o papel do docente também abrange algumas funções relacionadas ao instrucionismo, sendo elas a determinação do que será estudado e a forma de avaliação.

Quanto ao papel do discente, destacamos que para elaborar um algoritmo ele precisará explorar o objeto de conhecimento para entendê-lo, esse processo implicará na memorização de algumas informações e em uma mobilização ativa do estudante na construção de seu conhecimento, por meio dos diferentes tipos de abstração ocasionados na etapa de reflexão. Assim, percebemos que o papel do discente no Construcionismo contempla aspectos do instrucionismo e construtivismo.

No Construcionismo a colaboração pode ir além dos limites físicos da sala de aula, pois os discentes podem utilizar programas feitos por estudantes em outras partes do mundo para melhorar ou elaborar seus próprios algoritmos.

O foco do Construcionismo é a produção de algoritmos, mais especificamente programas de computador, diferentemente do instrucionismo que visa a memorização e o construtivismo que objetiva a compreensão e aperfeiçoamento das construções mentais do sujeito. Além disso, destacamos que no Construcionismo o artefato produzido pelo discente pode ser utilizado para a avaliação dele. Na seção seguinte apresentaremos e discutiremos as características do *Scratch*.

2.2 CARACTERÍSTICAS DO *SCRATCH*

O *Scratch* foi idealizado no ano de 2003 e lançado oficialmente em 2007 com versões *online* e *offline*, ambas gratuitas. Segundo Resnick *et al.* (2009), o nome *Scratch* se origina da "técnica de arranhar" (*Scratching technique*) usada por *DJs* para a remixar sons de forma criativa mexendo discos de vinil para frente e para trás. Desta forma o *Scratch* remete à ideia manipular, ajustar, mexer ou misturar, pois a atividade de programação dentro desse ambiente envolve a manipulação de gráficos, animações, fotos, músicas, sons e personagens.

Segundo Maloney *et al.* (2010) um dos principais objetivos do *Scratch* é introduzir a programação para pessoas sem experiência prévia, muitos aspectos do *design* desse ambiente foram tomados com base nesse objetivo como: a escolha de uma linguagem de blocos visuais, o *layout* da interface e a forma com que o erro é abordado. Rocha (2017) aponta outro objetivo chave desse *software*, facilitar e incentivar a aproximação de jovens e adultos às tecnologias digitais.

Para alcançar esses objetivos o *Scratch* foi construído sobre três princípios básicos de *design*: ser mais fácil de usar, mais significativo e mais social que outros ambientes de programação (RESNICK *et al.*, 2009).

Em relação ao primeiro princípio, a fim de tornar o *Scratch* mais fácil de usar, Resnick *et al.* (2009) se basearam no brinquedo Lego para decidir o *design* dos comandos. Na perspectiva dos autores, as crianças usualmente interagem com Lego de forma orgânica, manipulam as peças intuitivamente para construir diferentes tipos de estruturas, conforme brincam e constroem seus planos e objetivos evoluem naturalmente, o *Scratch* foi pensado para estimular um processo semelhante.

Usualmente as linguagens de programação possuem uma sintaxe própria e implícita, isso é, o programador precisa conhecer a sintaxe previamente para conseguir programar. De modo geral, sintaxe está relacionada a estruturação de uma sentença de modo que ela tenha sentido, analogamente, na programação a sintaxe está relacionada a estruturação de comandos de modo que a sequência de comandos tenha sentido lógico.

Rocha (2015) e Vanhove (2018) relatam que uma das principais dificuldades de iniciantes é aprender a sintaxe de uma linguagem de programação. Em vista disso, os comandos do *Scratch* são representados como blocos que se encaixam uns aos outros como se fossem um quebra-cabeça. Além de tornar a programação mais atrativa, fácil e intuitiva, esse *design* evita que erros de sintaxe sejam cometidos, pois somente comandos que tem sentido sintático podem ser encaixados (BRESSAN; AMARAL, 2015). Assim, pode-se programar no *Scratch* mesmo sem ter experiência ou conhecimentos prévios sobre o *software*.

Para tornar o *Scratch* mais significativo, ou seja, cumprir com o segundo princípio, dois critérios foram considerados prioritários: a diversidade e a personalização (RESNICK *et al.*, 2009). O *Scratch* possibilita que diversos tipos de projetos sejam desenvolvidos, tais como: a criação de jogos, quadrinhos e animações. Além disso, o *software* permite ao programador

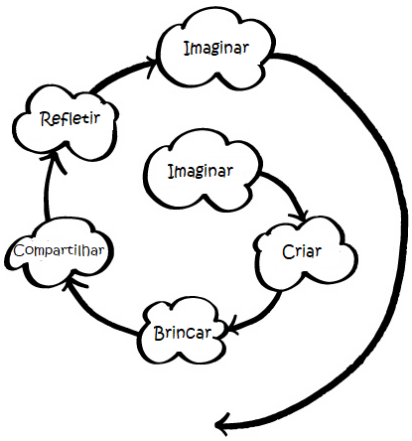
fazer *upload* de imagens, fotos, sons e músicas feitas por ele mesmo ou por outras pessoas, além de contar com uma biblioteca de recursos audiovisuais própria (RESNICK *et al.*, 2009; ROCHA, 2017). Esses aspectos permitem que as pessoas se expressem da forma que se sentirem mais confortáveis e estimula a criatividade.

O último princípio, ser mais social, está refletido no site e na interface do *Scratch.* O site possui uma biblioteca com projetos (ou programas) produzidos e disponibilizados pela comunidade de usuários. Todos os projetos postados na biblioteca do *Scratch* estão sujeitos a licenças *Creative Commons*, por essa razão qualquer pessoa pode comentar, avaliar ou baixar um projeto postado, além de acessar o código do projeto na íntegra (RESNICK *et al.*, 2009).

De acordo com Maloney *et al.* (2010), os usuários são estimulados a expor seus projetos e conhecer as produções de outras pessoas, assim, podem expandir seus conhecimentos e se inspirar explorando projetos presentes na biblioteca. Para estimular o compartilhamento e colaboração internacional, o *Scratch* conta com uma infraestrutura que traduz automaticamente os blocos do programa para mais de 40 linguagens diferentes (VANHOVE, 2018).

Segundo Resnick (2014) o ciclo de aprendizagem no *Scratch* pode ser representado por uma espiral. Essa espiral é chamada de “espiral de aprendizagem criativa”, apresentada na Figura 2.3 Os usuários imaginam o que querem fazer, criam um projeto com base nessas ideias, brincam com seus projetos, compartilham suas ideias e produções e refletem sobre suas experiências, isso os leva a iniciar um novo ciclo.

Figura 2.3 - Espiral de aprendizagem criativa.



Fonte: Adaptado de Resnick (2014).

Imaginar está relacionado a concepção de um produto e a elaboração de um objetivo a ser alcançado. O criar se relaciona com à elaboração de meios próprios para atingir esse objetivo. No brincar o usuário experimenta e explora possibilidades, pode-se dizer que brincar e criar são processos simultâneos. Após finalizar o processo de criação o usuário é incentivado a compartilhar seu projeto, estimulando outras pessoas e contribuindo para que desenvolvam novos projetos ou façam releituras sobre projetos existentes. Por fim, o refletir é o momento em que o sujeito analisa sua experiência com o processo de produção do seu projeto, essa reflexão pode gerar novos conhecimentos e novas ideias para projetos futuros (ORO *et al*., 2015; ROCHA, 2017).

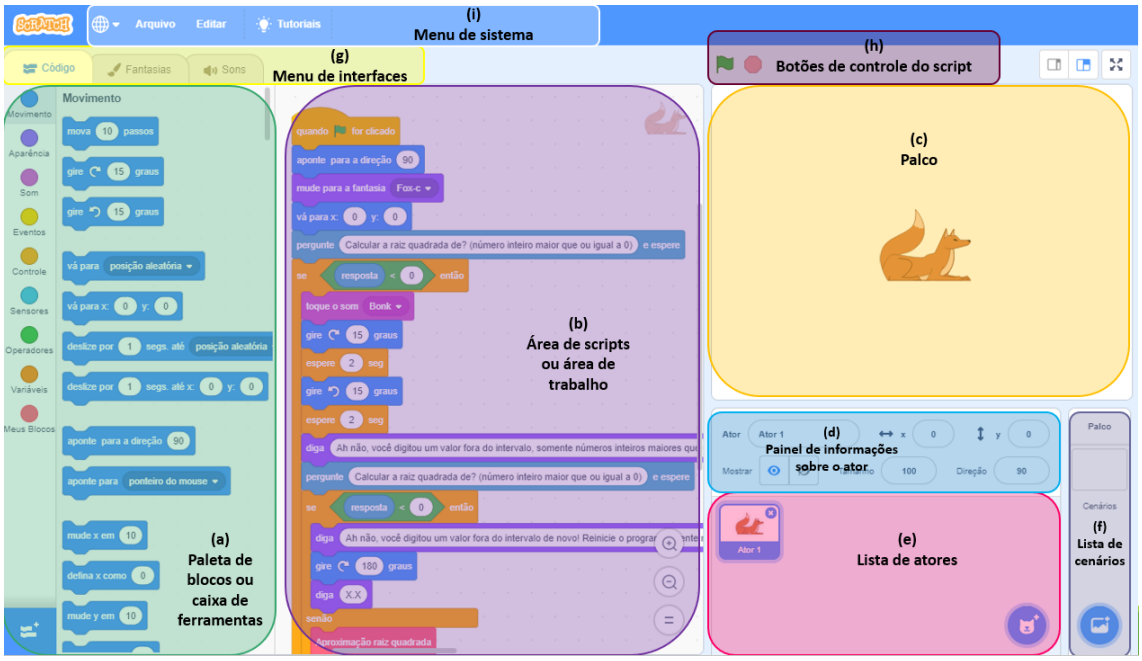
2.2.1 Interface do *Scratch* 3.0

O design da interface de programação do Scratch estimula a exploração e criação de códigos (ou *scripts*), além disso permite que o usuário faça a depuração do código de forma visual e intuitiva (MALONEY *et al.*, 2010).

A interface permite trabalhar sobre imagens 2D ao invés de 3D, uma vez que as primeiras são mais fáceis de manipular, criar e exportar para o ambiente de programação. Vale ressaltar que apesar do sistema ser projetado para imagens 2D, há uma grande diversidade de projetos onde os usuários usam técnicas de perspectiva para simular objetos 3D (JOHN MALONEY *et al.*, 2010), como o uso de figuras isométricas.

A versão 3.0 do *Scratch* foi lançada em janeiro de 2019. Na Figura 2.4 apresentamos a interface principal do editor do *Scratch* 3.0 com indicadores de seus principais componentes.

Figura 2.4: Interface principal *Scratch* 3.0.



Fonte: Autores.

Essa versão apresenta um novo *layout* em relação a 2.0 como a mudança da área de *scripts* para o centro da interface, segundo Vanhove (2018) essa alteração incentiva os usuários a fazer mais experimentos. Os indicadores que aparecem na Figura 2.4 são detalhados no Quadro 2.2.

Quadro 2.2 - Descrição dos componentes apresentados na Figura 2.4

(continua)

ID	Componente	Descrição
(a)	Paleta de blocos ou caixa de ferramentas.	Menu onde o usuário pode selecionar blocos de comando. Os blocos são divididos em diferentes categorias e funções. Cada categoria tem uma cor específica. Cada função tem uma forma específica (VANHOVE, 2018).
(b)	Área de *scripts* ou área de trabalho.	Espaço onde o usuário pode manipular os blocos de comando. Foi projetada de modo a lembrar a área de trabalho de um *desktop*. O usuário pode deixar blocos extras ou pilhas de blocos espalhadas nesse espaço sem que interfiram na execução dos códigos (VANHOVE, 2018).
(c)	Palco**.**	Espaço onde a execução do código é apresentada. Ele é composto um cenário (pano de fundo) e por objetos programáveis (códigos ou atores). A posição dos atores no palco é determinada por coordenadas cartesianas.

Quadro 2.2 - Descrição dos componentes apresentados na Figura 2.4

(conclusão)

ID	Componente	Descrição
(d)	Painel de informações sobre o ator.	Esse painel mostra a identificação (nome), posição (coordenadas), tamanho e direção (em graus) do ator selecionado, além disso permite ao usuário selecionar se o ator será exibido ou ocultado durante a execução do código.
(e)	Lista de atores.	Permite ao usuário selecionar o ator que deseja programar/modificar.
(f)	Lista de cenários.	Permite ao usuário selecionar o cenário que deseja programar/modificar.
(g)	Menu de interfaces	Esse menu permite ao usuário transitar entre três painéis: painel de *scripts*, painel de fantasias e painel de sons (VANHOVE, 2018). O painel de fantasias permite ao usuário selecionar, editar ou produzir diferentes aparências para um ator. Essas fantasias podem ser utilizadas para simular o movimento dos atores. O painel sons permite ao usuário selecionar, carregar e editar seus próprios arquivos de áudio. O painel de *scripts* permite ao usuário produzir códigos e programas, é apresentado na Figura 2.4.
(h)	Botões de controle do *script*	São botões que permitem controlar a execução do programa. A bandeira verde serve como um sensor, seu efeito dependerá de como e se o usuário usa o comando "quando a bandeira verde for clicada". O botão vermelho para a execução de todos os códigos.
(i)	Menu de sistema	Esse conjunto de abas permitem ao usuário carregar e salvar códigos, utilizar o modo turbo para acelerar a velocidade de execução de códigos, mudar a linguagem do sistema e acessar tutoriais.

Fonte: Autores.

De modo geral a atualização trouxe inovações positivas, ampliando as funcionalidades e potencialidades do *Scratch*. Essas mudanças podem gerar certa estranheza para quem está acostumado com a versão 2.0, contudo elas logo se tornam familiares ao usuário. Na seção seguinte detalhamos a sintaxe do *Scratch*.

2.2.2 Introdução aos Comandos do *Scratch* 3.0.

O *design* dos comandos do *Scratch* foi inspirado no Lego, assim programar no *Scratch* significa conectar os blocos numa determinada sequência. Os blocos só podem ser encaixados se houver uma relação sintática entre eles, ou seja o *software* não os conecta, assim não é preciso aprender a sintaxe para poder programar.

Os blocos são categorizados em relação as suas funções sintáticas (categorias sintáticas), pode-se saber a qual categoria sintática um bloco pertence pelo seu formato (VANHOVE, 2018). No Quadro 2.3 apresentamos as categorias sintáticas, suas funções e formatos.

Quadro 2.3 - Categorias sintáticas dos comandos Scratch 3.0.

Nome	**Contexto sintático**	**Forma**
Blocos de chapéu	Iniciar *scripts*.	
Blocos de pilha	Comandar.	
Blocos-C	Conectar conjuntos de blocos.	
Blocos de tampa	Finalizar *scripts*.	
Blocos booleanos	Atribuir os valores lógicos "sim" ou "não".	
Blocos de relatório	Retorna um valor numérico ou lógico.	

Fonte: Adaptado de VANHOVE (2018).

Os blocos de comando também são classificados em relação a suas funções práticas (categorias funcionais). Na versão 3.0 existem 9 categorias funcionais regulares, a saber: movimento, aparência, som, controle, sensores, operadores, variáveis e meus blocos. Além dessas, existem 7 categorias funcionais de extensão que permitem ampliar o conjunto de funções do *Scratch*, sendo elas: caneta, música, sensor de vídeo, Google tradutor, *micro:Bit*, Lego *Mindstorms* EV3 e Lego *WeDO* 2.0 (VANHOVE, 2018). O Quadro 2.4 descreve as categorias funcionais em relação a cor e função prática dos blocos.

Quadro 2.4 - Categorias funcionais dos comandos Scratch 3.0.

Categoria	Cor	Função
Movimento	Azul escuro	Movimentar *sprites*[12].
Eventos	Amarelo	Iniciar a execução de um código (sequência de comandos).
Aparência	Roxo	Alterar a aparência de *sprites*. Mudar cenários. Executar efeitos gráficos (como balões de fala).
Controle	Laranja claro	Criar ou eliminar clones de atores. Finalizar a execução de um código. Repetir a execução de um código. Gerar condições lógicas para a execução ou não de um código. Forçar um tempo de espera entre a execução de dois comandos.
Som	Lilás	Tocar ou parar arquivos de áudio. Controlar o volume do som.
Sensores	Azul claro	Detectar condições de *hardware* e *software*.
Operadores	Verde	Realizar operações matemáticas numéricas ou lógicas.
Variáveis	Laranja	Criar, armazenar ou acessar variáveis numéricas e não-numéricas. Gerar listas de variáveis.
Meus blocos	Rosa	Criar processos personalizados.
Caneta	Verde azulado	Desenhar com base no movimento do ator.
Música	Verde azulado	Compor músicas. Simular instrumentos musicais.
Sensor de vídeo	Verde azulado	Detectar movimentos captados pela câmera do computador.
Google Tradutor	Verde azulado	Traduzir textos para outras linguagens.
micro:Bit	Verde azulado	Programar um *micro:Bit*[13].
Lego Mindstorms EV3	Verde azulado	Programar um Lego *Mindstorm*[13].
Lego WeDO 2.0	Verde azulado	Programar sensores e motores.

Fonte: Adaptado de VANHOVE (2018).

Vale ressaltar que no *Scratch* as variáveis podem ser numéricas ou não-numéricas. As variáveis numéricas podem ser criadas usando comandos da categoria “variáveis”, enquanto as

[12] Os objetos programáveis do *Scratch* são chamados de atores ou *sprites*.
[13] Objetos físicos que podem ser programados com o *Scratch*.

variáveis não-numéricas podem ser estabelecidas usando blocos de comando lógico (ou booleanos).

Os blocos booleanos funcionam de forma binária (“sim” ou “não”), são utilizados para validar perguntas geralmente associadas a comparações ou verificações, esses blocos são apresentados na Figura 2.5 parte (a). Os blocos booleanos podem ser encaixados ao interior de um bloco-c para verificar uma condição, possibilitando ao usuário utilizar operadores matemáticos para comparar variáveis como exemplificado na Figura 2.5 parte (b) (ROCHA, 2017).

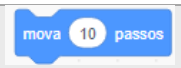
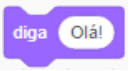
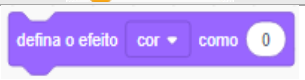
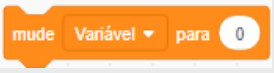
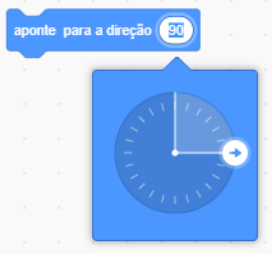
Figura 2.5 - Exemplos de comandos booleanos (a) e sua aplicação (b).

Fonte: Autores.

Resnick *et al.* (2009) apontam que minimizar o número de comandos disponíveis é importante, pois facilita o processo de programação. Para diminuir visualmente os comandos, alguns blocos aparecem somente quando necessário, seja na forma de um conjunto de comandos de extensão ou de seletores. Na versão 2.0 os comandos do tipo caneta faziam parte das categorias funcionais regulares, na versão 3.0 esse conjunto de blocos passou a integrar as extensões e precisa ser adicionado via menu de extensões.

Outro artifício utilizado para reduzir o número de blocos são os argumentos. Um argumento é um espaço em que o usuário pode inserir diversos tipos de informação. Além de economizar espaço, os argumentos facilitam o processo de programação, pois ajudam o usuário a perceber que tipo de informação ele precisa inserir. Os tipos de argumento e de seletores são apresentados no Quadro 2.5.

Quadro 2.5 - Tipos de argumentos e selecionadores no Scratch 3.0.

Tipo	Descrição	Formato
Argumento numérico	Aceita somente números.	
Argumento de texto	Aceita somente textos.	
Argumento de cor	Aceita números e cores. Esse bloco também funciona como seletor de cores.	
Argumento de áudio	Esse bloco funciona como seletor de arquivos de áudio.	
Argumento lógico	Aceita apenas blocos booleanos.	
Argumento de efeito	Relaciona efeitos com valores numéricos.	
Argumento de variável numérica	**Atualiza o valor de variáveis numéricas.**	
Argumento de direção	Aceita apenas números. Esse bloco também funciona como seletor de ângulos.	

Fonte: Adaptado de Vanhove (2018).

Os comandos do *Scratch* foram planejados para serem intuitivos e fáceis de manipular, além disso o *software* permite a produção de variados tipos de projeto. Destacamos que o *Scratch* fornece recursos para estimular e possibilitar que seus usuários aprendam de forma criativa, ou seja: imaginem, criem, brinquem, compartilhem e reflitam sobre suas experiências. Na sequência discutiremos as possíveis contribuições que o uso desse *software* pode trazer para a educação.

2.3 USO DO *SCRATCH* NA EDUCAÇÃO

O *Scratch* é um dos instrumentos de ensino mais utilizados para o desenvolvimento do PC, tem sido utilizado principalmente para ensinar Matemática e introduzir a computação em cursos de nível superior (BOMBASAR *et al.*, 2015; ELOY *et al.*, 2017).

Pode ser usado para desenvolver habilidades e conhecimentos que vão além programação (RESNICK, 2013), tais como a literacia digital e outras habilidades essenciais do século XXI, também pode ser empregado no ensino de conteúdos específicos de algumas disciplinas curriculares (CALDER, 2010; KALELIOĞLU; GÜLBAHAR, 2014; J. MALONEY *et al.*, 2004).

Pesquisas apontam que o *Scratch* pode contribuir com o desenvolvimento de três conjuntos de habilidades essenciais para o século XXI, são elas: informação e comunicação, pensamento e resolução de problemas, e interpessoais e auto direcionáveis (FRANÇA; AMARAL, 2013; PINTO; ESCUDEIRO, 2014). As habilidades de informação e comunicação podem ser desenvolvidas porque o *Scratch* permite aos educandos criar e gerenciar diversos tipos de mídias como textos, animações, imagens e gravações de áudio de forma criativa (PINTO; ESCUDEIRO, 2014).

Para elaborar projetos os discentes precisam coordenar interações entre vários objetos (ou atores), esse processo exige que o educando resolva problemas e sistematize soluções, contribuindo para o desenvolvimento de habilidades de pensamento e resolução de problemas (FRANÇA; AMARAL, 2013). Como o processo de elaboração de projetos pode ser complexo muitas vezes os discentes buscam ajuda de seus pares, assim o *Scratch* pode contribuir para estimular posturas colaborativas e investigativas desenvolvendo habilidades interpessoais e auto direcionáveis (PINTO; ESCUDEIRO, 2014).

Lummertz (2016) aponta que a utilização do *Scratch* pode desenvolver as seguintes habilidades da Literacia Digital: simulação, inteligência coletiva, julgamento, flexibilidade, multitarefa, navegação transmídia, cognição distribuída, jogabilidade e apropriação. A Literacia Digital é a capacidade de atuar de forma intuitiva em ambientes digitais, acessando de forma rápida e eficiente as informações contidas nesses espaços (VAN DEURSEN *et al.*, 2017).

Além disso, a programação pode ser utilizada para criar projetos relacionados a conteúdos de disciplinas curriculares como matemática, ciências e artes (MUGAYITOGLU,

2016). No entanto, Zoppo (2016) ressalta que o ensino por meio da programação não deve visar somente o conteúdo, deve ser pensado de modo a proporcionar ao educando oportunidades para experimentar e ser criativo.

Brennan (2015) aponta dois desafios no uso de linguagens de programação para ensinar conteúdos específicos das áreas de conhecimento, chamados pelo autor de tensões, são elas: tensão entre conhecimento técnico e conhecimento pedagógico sobre o instrumento (técnica x ensino) e tensão sobre direcionamento e descoberta (direcionamento x descoberta).

A tensão entre técnica e ensino diz respeito à necessidade de alcançar um equilíbrio entre estes conhecimentos, o que se tem sobre a linguagem de programação e sobre como usar essa linguagem para engajar os educandos em atividades criativas que proporcionem um aprendizado sobre conteúdos matemáticos (BRENNAN, 2015).

Benton *et al.* (2019) ressaltam que apesar dos *softwares* de programação atuais serem mais acessíveis essa tensão ainda é uma questão relevante a ser discutida. Resnick *et al.* (2009) apontam que o domínio da sintaxe de uma linguagem de programação, e a falta de conhecimentos ou habilidades específicas para orientar ou elaborar atividades, são problemas que comprometem a exploração do potencial desses recursos.

Bustillo e Garaizar (2014) apontam que muitas vezes docentes e discentes tem dificuldade em perceber o caráter transcurricular desses instrumentos. Por exemplo, usualmente professores fazem uso do *Scratch* para desenvolver um jogo, porém, a abordagem feita sobre a linguagem se limita a um passo-a-passo que deve ser seguido pelo estudante.

Benton *et al.* (2019) apresentam outra questão relacionada a essa tensão, para os autores além de dominar os aspectos técnicos e pedagógicos do instrumento, os docentes precisam pensar em tipos de conhecimento matemático que possam ser expressos pela programação. Em nosso entendimento, isso significa que os professores precisam reelaborar conhecimentos já adquiridos sob novas perspectivas.

A segunda tensão de Brennan diz respeito ao equilíbrio entre o direcionamento dos estudantes (fornecer informações sem o discente pedir, antecipar e direcionar as necessidades dos aprendizes) e a descoberta (intervir apenas quando o estudante pedir) (BENTON *et al.*, 2019).

Segundo Rader *et al.* (1997) crianças podem criar programas sem muito conhecimentos sobre programação, contanto que tenham ferramentas que as auxiliem. Alguns estudos sugerem que estudantes que já tiveram contato com o *Scratch* se sentem mais seguros para explorar ideias, enquanto aqueles que não possuem experiência prévia agem com maior cautela e evitam realizar ações que não foram requisitadas ou explicadas pelo professor (BENTON *et al*., 2016; BRENNAN, 2015).

Resnick (2014) aponta quatro elementos essenciais para promover experiências criativas de aprendizagem com o *Scratch*: projetos (*projects*), pares (*peers*), paixão (*passion*) e brincar (*play*). Esses elementos são chamados de 4Ps, devido as iniciais dos elementos em inglês.

Segundo o autor, as pessoas aprendem melhor quando se envolvem em *projetos* significativos, colaborando e compartilhando ideias com seus *pares*. Preferencialmente esses projetos devem estar relacionados a temas pelos quais o sujeito cultiva *paixões*, pois, deste modo os discentes são estimulados a persistir frente aos desafios. Por fim, na perspectiva de Resnick (2014) aprender envolve se divertir, *brincar*, tentar coisas novas, experimentar diferentes abordagens e assumir riscos.

Assim constatamos que esse *software* pode ser utilizado para enriquecer as práticas de ensino de Matemática e desenvolver habilidades fundamentais para o século XXI. No próximo capítulo discutiremos a relação entre o PM e o PC, partindo dessa relação refletimos sobre como o *Scratch* pode ser utilizado para desenvolver esses dois tipos de pensamento.

CAPÍTULO 3 PENSAMENTO ALGÉBRICO

A ideia de que a programação está relacionada com a Matemática não é recente. Benton *et al.* (2016) apontam que nas décadas de 70 e 80 pesquisadores já exploravam contribuições da programação para a aprendizagem de Matemática. Essa relação também pode ser estendida ao PC, uma vez que este engloba a programação.

Wing (2008) argumenta que o PC é um tipo de pensamento analítico que está relacionado ao PM. Gadanidis *et al.* (2016) corroboram com essa proposição ao sugerir que há uma conexão natural entre PM e o PC, tanto em sua estrutura lógica quanto em seu potencial de modificar relações entre objetos. Teixeira (2017) reforça esse argumento ao afirmar que há competências comuns entre o PC e o PM, tais como: criação de algoritmos, abstração, resolução de problemas, simulação, articulação de símbolos, modelagem, identificação de padrões e articulação lógica.

Apesar de diversos autores sugerirem uma correlação entre o PC e o PM, nenhum deles esclarece o que é o PM. Estudando a literatura sobre a área percebemos que não há um consenso sobre a definição de PM. Nesse sentido faz-se preciso caracterizar nosso entendimento sobre este tipo de pensamento para então analisar suas possíveis relações com o PC. Assim, neste capítulo discutiremos concepções gerais acerca do PM, para então voltarmos nossa atenção sobre o PA e sua relação com o PC.

3.1 PENSAMENTO MATEMÁTICO

Na cultura escolar conhecer matemática é entendido como lembrar e aplicar a regra correta na ordem correta, fazer matemática é visto como seguir as regras apresentadas pelos professores e a verdade matemática é determinada pela confirmação do resultado final pelo professor (LAMPERT, 1990), em suma, no senso comum a matemática é associada à certeza, exatidão e memorização. No entanto, entendemos que o PM é diferente da concepção popular.

O PM não se limita ao “fazer matemática” que normalmente compreende a aplicação de procedimentos e manipulações simbólicas, ele se refere a uma forma específica de pensar o mundo (DEVLIN, 2012). Segundo Krantz (2017) o carro chefe do PM é a lógica, pois por meio

dela o processo torna-se verificável, reprodutível e confiável. Porém, isso não significa que a intuição ou a imaginação não sejam elementos essenciais desse tipo de pensamento.

Monteleone, White e Geiger (2018) apontam algumas características do PM, a saber: usar a matemática e outros conhecimentos para gerar, avaliar, conectar e criar novas ideias; estabelecer generalizações entre conceitos; reconhecer relações; abordar problemas complexos de novas formas; raciocinar criticamente; considerar soluções alternativas; ser capaz de identificar e executar diversos caminhos para resolver problemas matemáticos; elaborar razões ou justificativas; usar estratégias matemáticas para provar que uma solução é ou não possível; se autoavaliar usando evidências matemáticas e raciocínio lógico; construir ideias por meio da explicação, questionamento, inferências, hipóteses e avaliações; e atribuir sentido a resultados ou procedimentos.

Mello (2015) discute o PM sob a perspectiva da solução de problemas, para ela esse tipo de pensamento envolve conceitos e princípios matemáticos, e uma atividade mental de alto nível. Segundo a autora, o sujeito analisa o problema de maneira sintética (estabelecimento de relações entre elementos do problema) e analítica (isolamento e hierarquização de diferentes elementos).

Sternberg e Ben-Zeev (1996) apontam que o PM envolve a habilidade de pensar de forma flexível. Os autores sugerem dois aspectos essenciais: a abstração e a linguagem. A abstração é necessária para resolver problemas desde a aritmética elementar até tópicos avançados de matemática.

A linguagem é essencial para conhecer, compreender e manipular os objetos matemáticos, a ponto de a matemática contar com uma linguagem própria (STENBERG; BEM-ZEEV, 1996). A linguagem matemática é diferente da língua materna, tem uma natureza bastante formal e necessita ser aprendida junto do conteúdo matemático estudado, se aproxima muito dos objetos matemáticos, no entanto a linguagem e os objetos matemáticos são distintos.

A Matemática é diferente das ciências sociais e naturais, possuindo especificidades cognitivas e epistemológicas. Duval (2013) destacam dois pontos que permitem evidenciar essas diferenças: a forma com que os objetos matemáticos são acessados e o modo como o conhecimento matemático é construído.

Segundo Duval (2013) não há sentidos físicos ou instrumentos que permitam o acesso direto aos objetos matemáticos, ou seja, só conseguimos interagir com esses objetos de forma indireta (DUVAL, 2013). A única forma de acessá-los é por meio de representações semióticas, contudo, a representação e o objeto são distintos e não devem ser entendidos como uma mesma coisa (DUVAL, 2013). Vale destacar que segundo a SBC (2019) há uma similaridade entre os objetos computacionais e os objetos matemáticos em relação a sua acessibilidade, uma vez que ambos só podem ser acessados por meio de representações.

A física, química, biologia e geografia são ciências que usualmente desenvolvem pesquisas com base em experimentos realizados no mundo material. Por exemplo, o modelo atômico foi aprimorado ao longo dos anos conforme surgiram instrumentos que possibilitaram coletar dados sobre os átomos no mundo físico.

Na matemática os problemas de pesquisa normalmente são resolvidos sem recorrer a verificações empíricas, uma vez que esse tipo de experimento está sujeito a aproximações excessivas (DUVAL, 2013). Ou seja, os experimentos são realizados sobre as representações semióticas de um objeto e validados com base na coerência lógica entre os argumentos e relações estabelecidos.

Além disso, nem sempre é possível correlacionar um problema matemático com o mundo material de forma imediata, algumas pesquisas tratam de problemas puramente abstratos. Como exemplo citamos a Teoria dos Números que atualmente é utilizada na criptografia, porém, na época de seu surgimento não se sabia se esse corpo teórico poderia ou não ter uma aplicação fora da matemática pura.

Considerando essas especificidades epistemológicas, Duval (2013) relatam a existência de duas faces da matemática: a face exposta e a face oculta. A face exposta corresponde aos objetos, demonstrações, propriedades e algoritmos matemáticos, ou seja, ao corpo teórico da matemática que usualmente embasa os currículos escolares.

A face oculta corresponde aos aspectos epistemológicos e cognitivos específicos da matemática, é chamada de oculta porque se manifesta de forma indireta. Os processos cognitivos que compõem a face oculta são independentes dos procedimentos que garantem a confiabilidade científica da matemática (DUVAL, 2013).

O PM não se resume aos conhecimentos matemáticos e é composto de vários aspectos cognitivos e psicológicos. Nesse sentido, PM pode ser entendido como uma competência composta por diversas habilidades cognitivas que não necessariamente estão subordinadas a algum conhecimento Matemático específico.

Em nossa pesquisa observamos que o PM é abordado em diversos trabalhos científicos, contudo, há uma escassez de estudos que discutam a definição desse tipo de pensamento. Devido a isso recorremos a definição proposta pelo currículo de Matemática do Ministério de Educação Nacional da Colômbia de 2006.

Nesse documento o PM é subdividido em cinco tipos: pensamento numérico, pensamento métrico, pensamento aleatório, pensamento espacial e pensamento variacional. O pensamento lógico não é caracterizado como um tipo de PM porque é entendido como um elemento intrínseco ao próprio PM, visto que é essencial na complexa composição da estrutura formal da Matemática (COLOMBIA, 2006).

O pensamento numérico está diretamente relacionado ao conceito de unidade, medição e número. A transição do conceito de número natural para o conceito de número racional, por exemplo, exige uma reconceitualização da ideia de unidade e do processo de medição em si, bem como uma extensão do conceito de número. O conceito de número negativo, por sua vez, é resultado da quantificação de certas mudanças nas medidas de uma quantidade, ou da medida relativa de uma quantidade em relação a um ponto de referência, identificado como zero (COLOMBIA, 2006).

Assim, entendemos que o pensamento numérico pode ser caracterizado como um conjunto de processos (operatórios e cognitivos), conceitos, proposições, modelos e teorias (COLOMBIA, 2006) que permitem ao sujeito configurar estruturas conceituais sobre os diferentes sistemas numéricos e aplicar essas estruturas em diferentes contextos.

Pensamento métrico está relacionado a compreensão geral de uma pessoa sobre as magnitudes e grandezas, conceito de medição e uso flexível de sistemas métricos diferentes em diversos contextos. Esse tipo de pensamento difere do numérico porque transcende o tratamento exclusivamente numérico dos sistemas de medição. Além disso, faz uso da estimativa por meio de procedimentos numéricos de truncamento e arredondamento, tratamento de erro, técnicas de enquadramento, notações científicas, entre outros (COLOMBIA, 2006).

O pensamento métrico tem uma íntima relação com as ciências, com o mundo físico e com contextos nos quais não é necessário estabelecer medidas numéricas exatas (COLOMBIA, 2006). Nesse sentido o pensamento métrico pode ser entendido como um conjunto de processos cognitivos que auxiliam o sujeito a compreender e manipular a dimensão, grandeza e magnitude de objetos físicos ou mentais.

O pensamento aleatório, ou probabilístico, é empregado para tomar decisões em situações de incerteza, acaso, risco ou ambiguidade devido à falta de informações confiáveis, nas quais não é possível prever com certeza o que vai acontecer. Esse tipo de pensamento está relacionado de forma direta a estatística inferencial e teoria da probabilidade, de forma indireta a combinatória e estatística descritiva (COLOMBIA, 2006). Assim, o pensamento aleatório pode ser caracterizado como um conjunto de processos cognitivos que ajudam o sujeito a pensar a complexidade que vem da incerteza, tomar decisões frente a causalidade múltipla e incontrolável.

O pensamento espacial ou geométrico engloba as relações estabelecidas entre objetos, suas transformações e representações materiais. O pensamento espacial opera em modelos internos de espaço interativo, analisando movimentos corporais e deslocamentos de objetos por meio de diferentes registros de representação e sistemas notacionais ou simbólicos. Dessa forma, o pensamento espacial pode ser entendido como um conjunto de processos cognitivos que permitem a construção e manipulação de representações mentais sobre objetos no espaço (COLOMBIA, 2006).

O pensamento variacional ou algébrico está relacionado a identificação e caracterização das variações de um conjunto de fatores em relação a outro conjunto de fatores, não se limita a variáveis numéricas. Além disso, esse tipo de pensamento se relaciona com a descrição, modelagem e representação de variações usando diversos tipos de sistemas ou meios simbólicos como ícones, linguagem materna, gráficos ou linguagem matemática. Esse tipo pensamento é essencial na resolução de problemas relacionados a variação e mudança, assim como na modelagem de processos cotidianos, ciclos naturais e sociais e, na investigação sobre a própria Matemática (COLOMBIA, 2006).

Neste sentido, o pensamento algébrico tem uma estreita relação com os outros tipos de PM, isso ocorre porque a análise de variações requer conceitos e procedimentos relacionados a diferentes sistemas numéricos, geométricos, métricos e probabilísticos, visto que esses sistemas

podem ocorrer de forma estática ou dinâmica. Além da íntima relação com o PM em geral, o PA está associado com pensamentos típicos das ciências naturais e sociais, especialmente por meio de modelagem de processos e situações naturais ou sociais (COLOMBIA, 2006). Ademais, Brandt e Moretti (2018) ressaltam que o PA não é caracterizado como uma extensão do pensamento aritmético[14].

Desta forma, podemos entender o PA como um conjunto de processos cognitivos que ajudam o sujeito a identificar semelhanças e diferenças entre objetos, sintetizar essas relações em procedimentos, algoritmos e expressões simbólicas. Vale ressaltar que a simbologia empregada na representação dessas relações vai além dos símbolos matemáticos formais, contemplando também expressões pictóricas ou verbais.

Com base nas definições e conceitos apresentados e discutidos nessa seção, no contexto desse trabalho, entendemos PM como um conjunto de estruturas e processos cognitivos que extrapolam a própria Matemática, esse tipo de pensamento pode ser subdivido em cinco outros, sendo eles: pensamento numérico, pensamento geométrico, pensamento métrico, pensamento probabilístico e PA.

O PM diferencia-se do pensamento empregado nas ciências naturais e sociais devido à natureza epistemológica dos conhecimentos matemáticos, visto que os objetos matemáticos não são observáveis, não existem no mundo físico e só podem ser acessados por registros de representação. Sendo assim, o PM está intrinsicamente relacionado à abstração. Reforçamos que o PM está relacionado à imaginação, à criatividade e à resolução de problemas, refutando assim a concepção de que esse pensamento é fechado, exato e livre de incertezas.

Dentre os tipos de PM, nesse trabalho optamos por nos aprofundar no PA devido sua relação intrínseca com os outros tipos de pensamento e com a programação. Na seção seguinte caracterizamos o PA para então correlacioná-lo com o *Scratch* e o PC.

[14] O Pensamento Aritmético está ligado ao cálculo e à realização de operações matemáticas para obter resultados numéricos (KIERAN, 1992), esse tipo de pensamento atua sobre sistemas e conjuntos numéricos (COLOMBIA, 2006). Enquanto o algébrico se relaciona com as estruturas e a generalização de relações entre objetos numéricos ou não-numéricos (KIERAN, 1992).

3.2 CARACTERIZAÇÃO DO PENSAMENTO ALGÉBRICO

Não há consenso sobre a definição de PA na literatura, uma vez que a álgebra abrange um extenso escopo de objetos de estudo e há diversos modos de conceber o conceito de pensamento algébrico (RADFORD, 2006). Nessa seção buscamos caracterizar nosso entendimento sobre PA partindo da perspectiva de dois pesquisadores: Luis Radford e James Kaput.

3.2.1 Pensamento Algébrico Segundo Luis Radford

Radford (2011) entende o PA como uma forma de pensar que foi refinada sucessivamente ao longo do tempo antes de atingir sua atual sofisticação, assim não é algo que desenvolvemos naturalmente. Em um artigo publicado em 2006 o autor buscou diferenciá-lo de outros tipos de pensamento, além de propor três elementos próprios do PA, são eles: senso de indeterminação, manipulação analítica de objetos desconhecidos e representação de objetos por meio de símbolos (RADFORD, 2006).

O senso de indeterminação está relacionado a objetos algébricos essenciais como parâmetros, incógnitas e variáveis. A manipulação analítica de objetos desconhecidos acontece quando o sujeito consegue operar objetos indeterminados como se fossem determinados, por exemplo, operar incógnitas ou variáveis como se fossem números conhecidos (RADFORD, 2006).

Diferentemente dos objetos geométricos que podem ser representados por meio de figuras, gráficos, objetos físicos, entre outros, os objetos algébricos só podem ser representados de forma indireta, isto é, designando determinados símbolos a determinados objetos (RADFORD, 2006). Brandt *et al.* (2018) relatam que aluno deve reconhecer a incógnita do problema, aceitar designá-la por uma letra e operar sobre ela como ele faz com um número e, também, saber traduzir os dados mais correntemente verbais em uma cadeia de operações escritas com letras e números.

Radford (2006) apresenta alguns conceitos para caracterizar o PA, a saber: generalização, objetificação, camadas de generalização, indução ingênua, generalização aritmética, generalização factual, generalização contextual e generalização simbólica.

A generalização consiste na percepção de uma regularidade local que é estendida para todos os termos de uma sequência, com base nessa percepção constrói-se expressões de elementos desconhecidos da sequência (RADFORD, 2006). Generalizar pode ser entendido como compreender toda uma sequência com base em uma parcela dela e, expressar essa compreensão por meio de símbolos (sintetizar a regularidade da sequência).

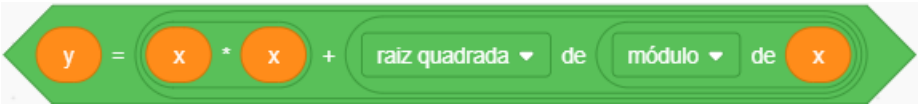
Objetificação, na perspectiva de Radford (2006), pode ser entendida como tornar um certo aspecto de um objeto aparente, no caso de objetos concretos isso pode ocorrer apontando sua cor ou seu tamanho, no caso de objetos matemáticos podemos usar diferentes tipos de sinais e artefatos (gráficos, palavras, desenhos, símbolos matemáticos, entre outros) para expressar uma característica. Como exemplo considere um número par, esse tipo de número pode ser representado por $2n$, sendo $n \in \mathbb{Z}$. Os artefatos, gestos, signos e outros recursos empregados para objetivar o conhecimento são chamados por Radford (2006) de meios semióticos de objetificação.

No tocante aos meios de representação semiótica, Duval (2004) destaca que os símbolos não são importantes, mas sim as designações em relação aos objetos que estes símbolos permitem estabelecer. Ademais utilizar vários modos de designação não é fácil e nem espontâneo (DUVAL, 2004), pois o sujeito precisa apreender e perceber o objeto de perspectivas distintas exigindo a mobilização de diferentes habilidades cognitivas.

Brandt e Moretti (2018) descrevem uma operação cognitiva relacionada a designação: a conversão. Em nossa concepção essa operação é amplamente utilizada no ensino de álgebra usando programação. Na conversão o sujeito transforma um registro que referencia um determinado objeto de um sistema semiótico para outro, preservando a referência (BRANDT; MORETTI, 2018).

Para ilustrar a operação de conversão considere que desejamos representar a equação $y = \mathrm{x}^2 + \sqrt{|x|}$ no *Scratch*, na Figura 3.1 apresentamos essa representação. Nesse caso a conversão consistirá na transposição dessa representação em linguagem matemática para a linguagem do *Scratch*, ou seja, se refere ao esforço cognitivo do sujeito de pensar como essa relação pode ser escrita com os comandos do *software*.

Figura 3.1 - Representação da equação $y = x^2 + \sqrt{|x|}$ no Scratch.



Fonte: Autores.

Para isso o sujeito deverá garantir que as características, objetos e relações presentes na equação sejam preservados e mantenham seu sentido léxico dentro do *Scratch*. Ademais, o sujeito precisa perceber que pode utilizar comandos de variável para designar x e y, e que pode combinar comandos de operadores para construir a igualdade, a potência, o módulo e a raiz quadrada.

É preciso também definir as variáveis dentro do *Scratch* utilizando o comando "criar variável", relacionar os diferentes operadores de modo que a ordem de operações seja mantida, e posicionar as variáveis dentro dessa estrutura preservando as relações indicadas na equação.

Brandt e Moretti (2018) apontam que essa operação cognitiva está relacionada a dois elementos fundamentais: sentido e referência. Segundo os autores as transformações que o registro sofre podem gerar mudanças de sentido, mas não de referência.

Como exemplo considere a sentença "todos os números positivos", em linguagem matemática essa relação pode ser indicada como $x \geq 0$, o conjunto numérico referenciado permanece o mesmo, porém o sentido das representações é diferente. Na representação materna o quantificador é evidenciado, além de explicitar que todos os elementos são positivos, na representação matemática essas características ficam implícitas.

Indução ingênua se refere aos casos em que a regularidade de uma sequência é obtida a partir da tentativa e erro, o sujeito não a estende para toda a sequência, e a expressão do padrão de repetição é formada por adivinhação (RADFORD, 2006). Vale ressaltar que a indução ingênua é diferente de outros tipos mais sofisticados de indução.

A generalização aritmética consiste na generalização de uma regularidade local, observada em um grupo limitado de elementos, sem estabelecer uma expressão que forneça qualquer elemento da sequência (RADFORD, 2006). Nesse caso, a generalização se restringe aos elementos observáveis.

Na generalização factual o sujeito consegue estender a regularidade a um conjunto limitado de elementos, como até o elemento 1.000, até o elemento 10.000, assim por diante. Nesse tipo de generalização a indeterminação (ou incógnita) permanece implícita, ou seja, é expressa apenas como ações (RADFORD, 2006).

Generalizações contextuais são generalizações estabelecidas sobre objetos contextuais como o "elemento seguinte" ou "elemento superior", nesse caso a descrição da regularidade expressa a perspectiva do sujeito sobre os elementos. Nas generalizações contextuais as expressões simbólicas usualmente são uma mistura de símbolos matemáticos e termos da linguagem materna (RADFORD, 2006).

Se diferencia da generalização factual tanto no tratamento sobre a indeterminação, quanto pelos meios semióticos empregados. Na generalização contextual a expressão algébrica é uma descrição do termo geral da sequência, enquanto na generalização factual a descrição se limita a um conjunto de elementos particulares (ALMEIDA; SANTOS, 2017).

Na generalização simbólica a regularidade é expressa por meio de símbolos alfanuméricos, diferente dos outros tipos de generalização que fazem uso de vários tipos de recursos semióticos. Nesse tipo de generalização o sujeito atribui um significado aos símbolos atrelado à sua percepção sobre a sequência. No entanto, essas representações não necessariamente são totalmente abstratas, elas ainda podem estar relacionadas a percepções contextuais dos sujeitos, ou seja, os termos como "superior" ou "seguinte" empregados na generalização contextual podem ser codificados em símbolos alfanuméricos (RADFORD, 2009).

Segundo Radford (2006) os conceitos e objetos de conhecimento objetificados são compostos de camadas de generalidade. Assim, nosso conhecimento sobre um determinado objeto conceitual é coincidente com as camadas de generalidade nas quais conseguimos trabalhar com o objeto. As camadas de generalidade são caracterizadas pelos meios semióticos de objetificação usados pelo sujeito para estabelecer generalizações. Radford (2006) propõe um esquema sobre as camadas de generalização apresentado no Quadro 3.1.

Quadro 3.1 - Estratégias para generalização de regularidades e subdivisão das camadas de generalidade.

Indução Ingênua	**Generalização**			
Adivinhação (tentativa e erro)	Aritmética	Algébrica		
	-	Factual	Contextual	Simbólica

Fonte: Adaptado de Radford (2006).

A partir do Quadro 3.1 percebemos duas camadas de generalização: aritmética e algébrica. Notamos também que a indução ingênua não é considerada uma camada de generalização, conforme Radford (2006) a adivinhação não caracteriza álgebra.

Na camada da generalização aritmética o sujeito consegue reconhecer uma regularidade entre elementos conhecidos, mas não a estende para elementos desconhecidos. Na camada factual, a indeterminação não é nomeada e permanece implícita, contudo, o sujeito consegue fazer uso de palavras, gestos e outros meios semióticos para estender a regularidade a um conjunto limitado de elementos desconhecidos (RADFORD, 2006).

Na camada contextual a generalidade contempla todos os elementos da sequência e é expressa em relação a objetos perceptíveis usando termos chave como “superior” ou “seguinte”. Nessa camada o sujeito faz uso da linguagem simbólica e materna para descrever os objetos. Na camada simbólica o sujeito utiliza e associa símbolos alfanuméricos aos objetos (RADFORD, 2006).

Segundo Radford (2006) as camadas de generalização não são formas de dizer a mesma coisa usando códigos diferentes, elas indicam formas mais profundas de consciência do sujeito sobre o objeto analisado, estão relacionadas ao uso de sinais para pensar de forma diferente. Partindo dessas ideias em Radford (2009) organiza o PA em três vertentes: PA factual, PA contextual e PA padrão.

O PA factual ocorre dentro da camada factual de generalização, Radford (2009) aponta que esse tipo de pensamento faz uso de mecanismos sofisticados de percepção e articulação de recursos semióticos. Nesse tipo de pensamento a compreensão da regularidade e a imaginação do sujeito permanecem ancoradas em um processo de mediação sensorial, não se resumindo a uma forma simples de reflexão Matemática.

De modo forma similar, o PA contextual ocorre na camada contextual de generalização. Nesse tipo de PA o processo de objetificação é mais profundo que o realizado no PA factual, o sujeito deixa de pensar em elementos particulares e passa a pensar em elementos gerais, além disso a indeterminação passa a ser descrita de forma explícita (RADFORD, 2009).

O PA padrão ocorre na camada simbólica de generalização, nesse tipo de pensamento o sujeito faz uso da linguagem simbólica alfanumérica, explicitando as relações de forma

contextual ou totalmente abstrata. No quadro 3.2 sintetizamos a descrição de todos os termos discutidos nessa subseção.

Quadro 3.2 - Conceitos e termos que caracterizam o PA com base em Radford (2006).

Termo	Descrição
Senso de indeterminação	Relaciona-se à ideia de parâmetros, argumentos, incógnitas e variáveis.
Manipular objetos desconhecidos de forma analítica	Ser capaz de operar objetos indeterminados como se fossem determinados.
Objetificar	Evidenciar determinado aspecto de um objeto matemático usando algum recurso semiótico.
Generalizar	Perceber uma regularidade local com base em alguns elementos de uma sequência. Estender a regularidade para um conjunto mais abrange de elementos, ou, para todos os termos da sequência. Construir expressões para representar elementos desconhecidos da sequência.
Camadas de generalização	Refletem a abrangência e complexidade da generalização estabelecida, são elas: aritmética, factual, contextual e simbólica. Podem ser caracterizadas pelos meios semióticos de objetificação empregados na representação da regularidade.
Indução ingênua	Determinação da regularidade de uma sequência com base na tentativa e erro. Não é considerada como um tipo de generalização.
Generalização aritmética	A generalização se restringe à elementos observáveis. A indeterminação não é enunciada.
Generalização factual	A generalização é estendida a um conjunto limitado de elementos. A indeterminação não é enunciada.
Generalização contextual	A generalização pode ser estendida a um conjunto infinito de elementos. A descrição da regularidade reflete a perspectiva do sujeito sobre a sequência. A representação usualmente é uma mistura de símbolos matemáticos e termos da linguagem materna.
Generalização simbólica	A generalização pode ser estendida a um conjunto infinito de elementos. A descrição não depende da perspectiva do sujeito. A regularidade é expressa por meio de símbolos alfanuméricos.
PA factual	PA relacionado a camada de generalização factual. A compreensão e imaginação do sujeito permanecem ancoradas em um processo de mediação sensorial.
PA contextual	PA relacionado a camada de generalização contextual. O sujeito deixa de pensar em elementos particulares e passa a pensar em elementos gerais. A indeterminação é descrita de forma explícita
PA padrão	PA relacionada a camada de generalização simbólica. O sujeito faz uso da linguagem simbólica alfanumérica. As relações são explicitadas de forma contextual ou totalmente abstrata.

Fonte: Autores.

Na perspectiva de Radford o desenvolvimento do PA começa no pensamento factual, passa pelo contextual até chegar no simbólico. Na seção seguinte discutimos o PA na perspectiva Kaput.

3.2.2 Pensamento Algébrico na Perspectiva de James Kaput

Para Kaput (2000) generalizar pode ser entendido como estender deliberadamente a amplitude de um raciocínio sobre uma situação específica, para o conjunto de todas as situações em que esse mesmo raciocínio é aplicável. Ou seja, generalizar é identificar e explicitar um "caso geral" a partir de um caso particular.

Outra forma que Kaput (2000) propõe para caracterizar a generalização é: elevar um raciocínio para um nível no qual as regularidades sobre um conjunto de situações sejam o foco, ao invés de focar sobre uma situação particular. Assim, ao generalizar o sujeito deixa de analisar o caso particular e passa a analisar o "caso geral" ou a regularidade.

Kaput (2000), assim como Radford (2006), entende que a linguagem utilizada para expressar as regularidades pode ser composta por diferentes meios semióticos. Uma criança usualmente faz uso da língua materna ou de gestos para comunicar as regularidades que percebe, um matemático costuma usar a linguagem matemática formal, segundo o entendimento dos autores supracitados ambas são formas aceitáveis de comunicação de generalizações.

Segundo Kaput (2005) a álgebra deve ser pensada como: (a) generalização e formalização de restrições e padrões, e aritmética generalizada; (b) manipulação de formalismos com base em sintaxe; (c) estudos de sistemas e estruturas de cálculos e relações; (d) estudo de relações, funções e variações conjuntas; e (e) conjunto de modelagens e mecanismos linguísticos.

As formas (a) e (b) são o núcleo da atividade algébrica e dão suporte às demais, pois grande parte das atividades matemáticas estão relacionadas com a generalização e formalização de regularidades (KAPUT, 2000). Entretanto, o autor ressalta que os formalismos podem não ser suficientes para gerar generalizações. No formalismo nossa atenção se concentra sobre as regras sintáticas e os símbolos em detrimento daquilo que representam (SOARES, 2018).

As formas (c) e (d) usualmente são empregadas como conteúdos curriculares. A forma (c) pode ser entendida como álgebra abstrata e costuma ser abordada no nível universitário, (d) caracteriza uma visão prática sobre a álgebra e costuma ser abordada na Educação Básica. A forma (e) caracteriza a álgebra como uma rede de linguagens (SOARES, 2018). Na sequência explicaremos como essas formas permeiam o entendimento de PA de Kaput (2008).

Para Kaput (2008) o PA é uma atividade exclusivamente humana, intimamente relacionada ao estabelecimento de generalizações por meio de argumentos e conjecturas. Nesse sentido, pensar algebricamente é um processo no qual uma relação matemática é generalizada com base em um conjunto particular de exemplos, e o estabelecimento dessas generalizações pode ser feito usando diferentes meios semióticos (BLANTON; KAPUT, 2005). Esse processo pode ocorrer em diversos contextos matemáticos como aritmética, geometria, modelagem, entre outros (KAPUT, 2008).

O conhecimento matemático está no sujeito, não no objeto (KAPUT, 2008). Assim, uma pessoa não percebe uma incógnita em uma equação como um objeto algébrico de forma natural, para que isso ocorra é preciso que o indivíduo construa um significado para essa estrutura. Almeida e Santos (2017) apontam que para pensar algebricamente um indivíduo precisa entender a equação como uma relação de equivalência entre os dois membros, e compreender que para torná-la verdadeira precisa descobrir o valor da incógnita de modo que essa equivalência seja preservada.

Kaput (2008) reformula e sintetiza as cinco formas de pensar a álgebra em três vertentes, a saber: o pensamento quantitativo, o pensamento funcional e a modelação. O pensamento quantitativo, ou aritmética generalizada, está relacionado ao caráter algébrico da aritmética. Nessa vertente estão as generalizações acerca de propriedades e operações aritméticas, bem como relações entre números (KAPUT, 2008).

O pensamento funcional é caracterizado pela generalização de regularidades numéricas para descrever relações de variação e relações funcionais. Nessa vertente a generalização está intimamente relacionada a ideia de função (KAPUT, 2008).

A modelação está relacionada à sintaxe algébrica (KAPUT, 2008), isto é, à estrutura das expressões e formalizações empregadas para representar generalizações. Essa vertente está intimamente relacionada com as duas primeiras, considerando que ela é utilizada para modelar as generalizações nas demais. Almeida e Santos (2017) destacam que a modelagem pode ser feita utilizando outros meios semióticos além da linguagem simbólica formal, como linguagem pictórica, gestual, natural, entre outras.

A modelação pode ser subcategorizada em três tipos: (i) modelagem de situações aritméticas; (ii) generalização de regularidades inerentes à própria matemática ou de fenômenos sociais ou naturais e; (iii) generalizações com base em parâmetros (SOARES, 2018).

No tipo (i) a generalização pode ser representada utilizando uma incógnita, no tipo (ii) podemos representá-la usando uma ou mais variáveis, no tipo (iii) por um ou mais parâmetros (SOARES, 2018). As equações abaixo exemplificam os tipos de modelação:

Exemplo tipo (i): $a + b = b + a$;

Exemplo tipo (ii): $Área = Comprimento \times Largura$ ou $Força = Massa \times Aceleração$;

Exemplo tipo (iii) $f(x) = kx$ (neste caso x é um argumento, k é uma constante ou parâmetro dependendo do contexto e $f(x)$ é o valor resultante).

Kaput (2008) também caracteriza dois aspectos essenciais do PA que se relacionam com essas vertentes, a saber: a generalização e formalização simbólica gradual (aspecto A) e a ação metódica sobre sistemas simbólicos (aspecto B).

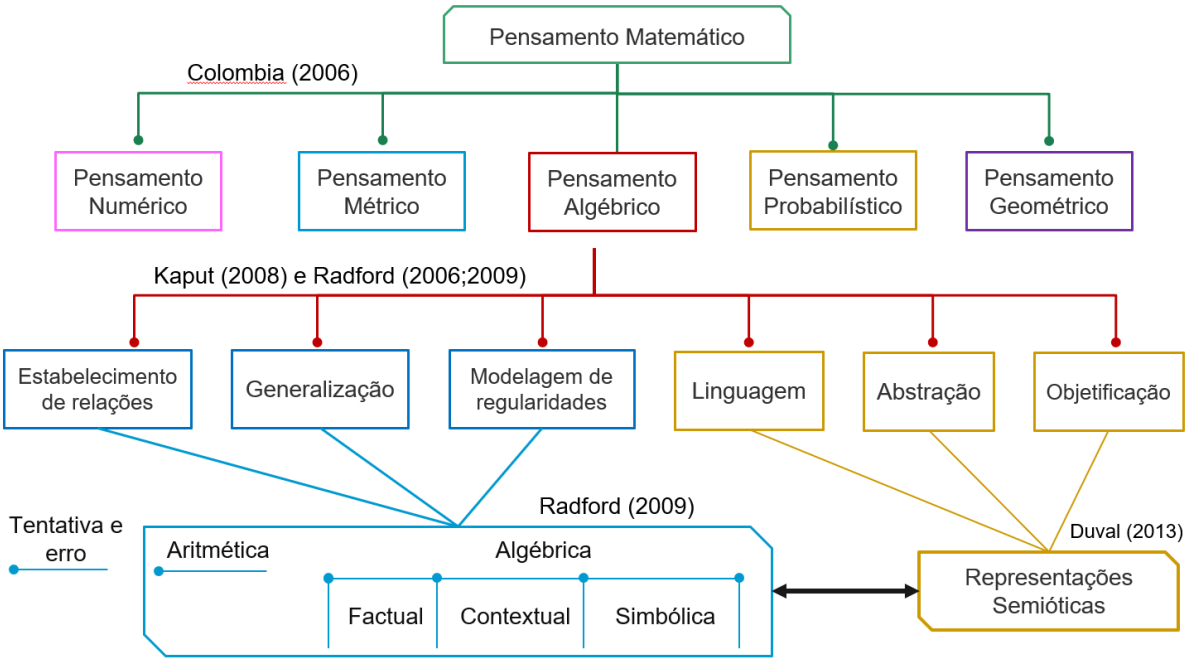
O aspecto A diz respeito à capacidade de estabelecer relações entre objetos e expressá-las usando diferentes tipos de símbolos. Além disso, esse aspecto também sugere que conforme o indivíduo amadurece seu PA os símbolos que ele utiliza vão se tornando cada mais formais, e os meios semióticos informais vão deixando de ser utilizados.

O aspecto B está relacionado à capacidade do sujeito de elaborar e manipular sistemas simbólicos, podemos dizer que esse aspecto também tem relação com a sintaxe algébrica. Embora alguns autores entendam que o aspecto B é um forte indicativo de PA independentemente de como o indivíduo o usa, Kaput (2008) sugere que esse aspecto só pode ser considerado uma manifestação de PA quando o sujeito o usa para generalizar ou modelar uma relação.

Kaput (2008) aponta que usualmente o aspecto A é desenvolvido após o B, visto que a ação metódica sobre símbolos depende de conhecimentos prévios sobre o significado e propriedades das relações entre esses símbolos. Vale ressaltar que o mais importante para o PA é a construção de significados para a linguagem simbólica empregada (KAPUT; BLANTON; MORENO, 2008). Assim, o PA consiste em analisar por meio dos símbolos e não em analisar os símbolos (ALMEIDA; SANTOS, 2017).

Na figura 3.2 apresentamos um esquema que relaciona os conceitos e ideias discutidos nas seções 3.1 e 3.2 que nos ajudaram a caracterizar o PA a partir do PM e suas subcategorias.

Figura 3.2 - Caracterização do PA.



Fonte: Autores.

Entendemos que o PA é uma atividade essencialmente humana e intrinsecamente relacionada ao estabelecimento de relações, à generalização, à objetificação, à abstração, à linguagem e à modelagem de regularidades. Além disso, o PA pode ser manifestado por diferentes meios semióticos e caracterizado por níveis de generalidade.

Em relação ao ensino de álgebra, devemos focar sobre como os alunos constroem sua compreensão sobre a sintaxe algébrica e, como atribuem sentido e significado aos símbolos e sintaxe. Ademais, também destacamos a necessidade de se considerar aspectos psicológicos e epistemológicos da álgebra no planejamento de atividades de ensino, além dos aspectos pedagógicos e teóricos. Consideramos que analisar as manipulações simbólicas sobre uma perspectiva linguística também é importante para o desenvolvimento do PA, assim atentar-se a forma com que os estudantes realizam e entendem as interações e relações entre símbolos é uma forma de avaliar seu PA.

3.3 PENSAMENTO ALGÉBRICO, PENSAMENTO COMPUTACIONAL E *SCRATCH*

Neste subcapítulo buscamos estabelecer pontes entre o PA, PC e o *Scratch*. Na seção 3.3.1 relacionaremos o PC com o PA, além disso discutiremos como a programação (em um

sentido amplo) pode ser usada para desenvolver esses dois tipos de pensamento. Na seção 3.3.2 apresentamos exemplos de como o *Scratch* pode ser usado para desenvolver o PA e o PC.

No contexto deste subcapítulo o termo “programador” designará o sujeito que elabora o programa, enquanto o termo “usuário” designará o indivíduo que usa o programa.

3.3.1 Relações entre Pensamento Computacional e Pensamento Algébrico

Partindo do entendimento de que a programação pode ser utilizada para abordar diversos aspectos do PC, Kilhamn e Brâting (2019) realizaram um estudo com o intuito de identificar aspectos comuns ao PC e o PA a partir de definições contemporâneas nas respectivas áreas. O Quadro 3.3 apresenta a comparação entre as definições de PC com as definições de PA.

Quadro 3.3 - Comparação entre definições de PC e PA.

Característica	**Definições de PC**		**Definições de PA**	
	Grover e Pea (2013)	**Hoyles e Noss (2015)**	**Radford (2018)**	**Kaput (2008)**
Estrutura	Decomposição estrutural de problemas (modularização).	Decomposição. Reconhecimento de padrões.	Lidar com quantias indeterminadas de forma analítica.	Estudo das estruturas.
Simbolização	Sistemas simbólicos e representações.	(Não há menção sobre simbolização).	Modos idiossincráticos de simbolização.	Trabalhar sobre sistemas de símbolos com sintaxe definida.
Generalização e abstração	Abstração e reconhecimento de padrões. Lógica condicional.	Implica abstração.	Lidar com quantias indeterminadas de forma analítica.	Usar sistemas de símbolos para expressar generalizações.
Funções e variáveis	(Não há menção sobre funções).	(Não há menção sobre funções).	(Não há menção sobre funções).	Estudo de relações, funções e variações conjuntas.
Algoritmos	Noções algorítmicas de fluxo de controle.	Pensamento algorítmico.	(Não há menção sobre algoritmos).	(Não há menção sobre algoritmos).

Fonte: Adaptado de Kilhamn e Brâting (2019).

No que tange às convergências as autoras pontuam que tanto o PC quanto o PA estão relacionados à estrutura, generalização e simbolização. Apesar desses elementos aparecerem tanto nas definições de PM quanto nas definições de PC não fica claro se representam as mesmas ideias nos dois tipos de pensamento, assim como se a semântica e sintaxe das linguagens estão alinhadas ou não (KILHAMN; BRÂTING, 2019).

Observando o Quadro 3.3 percebemos que no PC as estruturas são associadas à ideia de modularização, enquanto no PA são associadas ao conceito de incógnita ou sintaxe. Usualmente o estudo da estrutura em álgebra (por exemplo, aritmética generalizada) não ocorre de forma explícita na Educação Básica, nesse sentido, Kilhamn e Brâting (2019) sugerem que programação pode ser usada para deixar essas estruturas visíveis, considerando que ao dividir cálculos complexos em pequenos passos os discentes podem analisar essas estruturas de forma mais minuciosa.

Um exemplo de abordagem sobre o estudo de estruturas é a elaboração de um algoritmo que gere todos os números ímpares num intervalo limitado. Para produzir um código assim o programador precisa refletir sobre o que é um número ímpar, e como ele pode ser representado. Assim, o programador precisa refletir sobre como a estrutura dos números ímpares pode ser generalizada e representar essa regularidade usando uma linguagem de programação específica (KILHAMN; BRÂTING, 2019).

Na comparação das definições apresentadas no Quadro 3.3 observamos que abstração e generalização são características consolidadas nos dois tipos de pensamento, no entanto, são entendidas de formas diferentes.

As funções e variáveis não são mencionadas de forma explícita nas definições de PC, aparecendo apenas na definição de PA de Kaput (2008) conforme observamos no Quadro 3.3. Segundo Kilhamn e Brâting (2019) essas características são recorrentes e amplamente empregadas para a elaboração de algoritmos. As autoras supõem que, talvez devido ao caráter onipresente de funções e variáveis no exercício do PC, elas sejam subentendidas como essenciais ao ponto de não precisarem ser explicitamente mencionadas.

Quanto às variáveis, vale destacar que segundo Usiskin (1988) esse conceito pode ser utilizados para diversos propósitos, a saber: (i) generalizar padrões aritméticos; (ii) representar valores constantes desconhecidos; (iii) representar argumentos ou parâmetros; (iv) representar

objetos arbitrários em uma estrutura de relações e (v) indicar um endereço de registro de memória.

Normalmente na Matemática ensinada na educação básica os propósitos (i), (ii) e (iii) costumam ser abordados, o propósito (iv) tende a ser discutida na Matemática ensinada no ensino superior. Usiskin (1988) aponta que usualmente na computação todos os cinco propósitos são abordados. Entendemos que esse estudo corrobora com Kilham e Brâting (2019) quando as autoras argumentam que o conceito de variável pode ser expandido ao usar programação, uma vez que os discentes poderão perceber que as variáveis não se restringem a números.

Além disso, com a programação o papel da linguagem para representar os objetos pode tornar-se mais evidente, visto que os estudantes podem perceber que os objetos matemáticos não são os símbolos, ou seja, mas sim que os símbolos são representações dos objetos.

No Quadro 3.3 é possível observar que o pensamento algorítmico não é mencionado nas definições de PA, embora esteja relacionado a aspectos fundamentais da álgebra como a capacidade de analisar um problema, especificá-lo com precisão e encontrar ações básicas adequadas.

Kilhamn e Brâting (2019) relatam que o termo algoritmo fazia parte do currículo escolar nacional da Suécia e estava relacionado a aritmética. Em 2011 esse termo foi removido e em 2017 reinserido como um conteúdo central da álgebra. Para as autoras isso implica em uma mudança no entendimento sobre algoritmo que antes era visto como um aspecto procedimental da aritmética e, atualmente, é relacionado ao pensamento algorítmico.

O pensamento algorítmico engloba ponderações sobre o propósito de um algoritmo e os possíveis caminhos para construí-lo ou aplicá-lo (FUTSCHEK, 2006), consiste numa atividade criativa e reflexiva ao invés de um exercício de pura memorização.

Vale destacar que o quadro 3.3 não faz menção sobre depuração. Segundo Selby (2014) a depuração consiste em uma estratégia de resolução de problemas que parte do sintoma para encontrar e corrigir o erro. Entendemos que apesar desse termo não ser mencionado de forma explicita nas definições de PA, ele pode ser um ponto comum entre PA e PC.

Na programação isso significa que o indivíduo corrige erros lógicos nos comandos a partir da execução do algoritmo. Na álgebra isso pode significar estabelecer um caminho lógico

entre uma hipótese e uma tese, ou seja, conhecendo um resultado o sujeito pode pensar em como alcançá-lo partindo da hipótese.

Para ilustrar essa situação, considere esse exercício: considerando a equação (1) $\text{sen}^2\, x + \cos^2 x = 1$ demonstre a equação (2) $\text{tg}^2\, x + 1 = \sec^2 x$. Essa demonstração pode ser feita de forma direta, isto é, manipulando a hipótese para obter a tese. Para resolver esse problema o sujeito precisa se perguntar: Como farei para chegar à tese partindo da hipótese? Quais operações precisarei realizar? Quais propriedades trigonométricas podem ser úteis? Consigo fazer o caminho reverso, partindo da tese até a hipótese?

Ademais, uma vez que o discente tenha elaborado uma demonstração para justificar a igualdade entre essas expressões é comum que ele verifique se os passos estão corretos e, caso identifique algum erro, pense em como corrigi-lo mantendo a linha de raciocínio ou reformulando toda a demonstração. Esse processo poderia ser caracterizado como uma forma de depuração? Vale ressaltar que este estudo não tem por objetivo verificar se a depuração é ou não um ponto comum do PA e PC, apresentamos essa ideia como uma provocação e questionamento para pesquisas futuras.

Quanto à simbolização, percebemos que nas definições de PC não fica claro se o sistema de representação é uma linguagem de programação formal ou materna, a natureza do sistema não é explicitada. No PA a simbolização pode ser feita utilizando diferentes meios semióticos (RADFORD, 2018), indicando que o PA não está limitado exclusivamente a símbolos alfanuméricos e à sintaxe algébrica convencional, assim, as linguagens de programação podem ser consideradas como formas de expressão do PA.

Ainda em relação à simbolização, Kilhamn e Brâting (2019) apontam que as diferenças entre sistemas simbólicos algébricos e os adotados em linguagens de programação podem gerar dificuldades para os alunos aprenderem álgebra.

Para esclarecer algumas dessas diferenças as autoras elaboraram três categorias sobre as diferenças entre o significado de algumas operações (semântica) e as representações dessas operações (sintaxe) que aparecem na álgebra e na programação, essas categorias são apresentadas no Quadro 3.4.

Quadro 3.4 - Diferenças entre a semântica e sintaxe simbólica entre programação e álgebra segundo Kilhamn e Brâting (2019).

Categoria	*Exemplo em álgebra*	*Exemplo em programação*
Símbolos diferentes com mesmo significado	A operação de módulo é representada como $mod(x)$ ou $\|x\|$.	A operação de módulo pode ser representada como %
Símbolos iguais com significados diferentes	O sinal de igualdade (=) pode ser usado para representar uma relação de equivalência.	O sinal de igualdade (=) pode ser usado para representar uma atribuição.
Símbolos com significados exclusivos a um dos dois domínios.	Símbolo de aproximadamente (≈).	Pode não existir um símbolo para aproximadamente.
	Não existe sinal ou símbolo para o operador incremento.	Operador de incremento (++).

Fonte: Autores.

Kilhamn e Brâting (2019) ressaltam que essas diferenças podem desenvolver ou restringir o PA dependendo de como forem trabalhadas. Por exemplo, se forem exploradas para demonstrar que o significado de um conceito matemático não depende do símbolo empregado para representá-lo, as diferenças contribuem para o desenvolvimento do PA. Porém, caso esses diferentes significados sobre um mesmo símbolo não sejam bem trabalhados podem deixar os estudantes ainda mais confusos (KILHAMN; BRÂTING, 2019).

O ensino de álgebra por meio da programação possibilita aos alunos aprender a manipular e expandir entendimentos sobre variáveis, estruturas e padrões. Para elaborar os programas os sujeitos precisam simplificar, generalizar e simbolizar relações matemáticas usando uma determinada linguagem de programação (KILHAMN; BRÂTING, 2019).

Para que a programação seja usada de modo a favorecer o desenvolvimento do PA e PC é preciso atentar-se para a formação inicial e contínua de professores, visto que esse tipo de abordagem exige competências pedagógicas específicas e conhecimentos sobre álgebra e programação.

3.3.2 Relações entre o *Scratch*, Pensamento Computacional e Pensamento Algébrico

Embora muitas linguagens de programação usem sistemas simbólicos similares, esses sistemas não são equivalentes, ou seja, não é usual que as linguagens de programação tenham mesma sintaxe (KILHAMN; BRÂTING, 2019). Ao evitar que dois blocos que não têm relação sintática sejam conectados, o *Scratch* possibilita ao programador manipular os comandos de

forma intuitiva. Deste modo, o estudante pode focar sobre as relações matemáticas intrínsecas a um programa, sem precisar se preocupar com a sintaxe da linguagem de programação.

Os blocos de comando do *Scratch* possuem uma variedade de funções algébricas, pode-se dizer que as relações entre os blocos formam uma "álgebra de formas". Além disso, relacionar a função dos blocos com sua forma e cor facilita a identificação e compreensão de erros, além de auxiliar o programador a analisar a estrutura do código.

Blocos de movimento podem ser utilizados para trabalhar com coordenadas cartesianas, ângulos e transformações geométricas. No Logo os comandos para movimentar a tartaruga são "para frente", "para a direita", "para a esquerda" ou "para trás", no *Scratch* o programador usa comandos como "mova", "gire", "vá para (x,y)" ou "deslize por x segundos" para movimentar os atores.

Similar ao Logo, no *Scratch* é preciso usar um comando que direcione o ator para a direção que se deseja ir antes de fazê-lo andar. Assim, o *Scratch* exige que o usuário explore relações entre objetos no plano cartesiano para movimentar um ator, essas relações podem ser a distância entre pontos, distância entre retas, áreas, relações entre ângulos e retas, entre outras.

Os blocos operadores têm uma íntima relação com a aritmética e com a lógica. Esses blocos podem ser usados para explorar operações aritméticas como a soma entre duas parcelas e o arco tangente de um número real. Ademais, podem ser usados para estabelecer condições lógicas com base em comparações entre números reais ou conectivos lógicos "e", "não" e "ou". Os blocos de controle podem estabelecer condicionais lógicas e sintetizar processos repetitivos.

Observamos que os blocos de variáveis e blocos de sensores (como o bloco "resposta") podem ser utilizados para introduzir e manipular variáveis numéricas. Além disso, sensores podem representar variáveis não numéricas, como o comando "o mouse está pressionado?". Os blocos de evento também estabelecem condicionais lógicas, porém essas condicionais são baseadas na interação do usuário.

Para aprofundar nossas reflexões sobre as relações entre o *Scratch* e o PA exploraremos que aspectos da álgebra podem ser percebidos sobre dois programas. O primeiro é um programa que usa um ator qualquer para desenhar um quadrado, o segundo é um programa que fornece a aproximação da raiz quadrada de um número real positivo qualquer.

3.3.2.1 Análise sobre o desenho de um quadrado no *Scratch*

A programação de um quadrado no *Scratch* exige que o programador abstraia as propriedades desse objeto matemático e as represente usando os comandos do *software*. A forma com que os estudantes precisam pensar o desenho de um quadrado usando o *Scratch* é diferente do modo com que pensam esse processo usando lápis e papel.

Ao invés de pensar nos movimentos que faria com o lápis, o sujeito precisa pensar a partir da perspectiva do ator (objeto que será programado). Assim, o sentido que o ator está apontado no início da execução do código (para cima, baixo, esquerda, direita ou em alguma diagonal) afeta a forma com que o quadrado será representado, podendo ser paralelo aos lados da tela ou não (MILLER; LARKIN, 2017). Apesar da posição e sentido iniciais não afetarem significativamente o processo de construção do quadrado, eles podem afetar o processo de construção de outras figuras geométricas. Para ilustrar a diferença entre desenhar o quadrado no papel e no *Scratch* apresentamos, na Figura 3.2 e Figura 3.3, cinco formas diferentes de programá-lo no *Scratch*.

Figura 3.3 - Exemplo de programas que desenham um quadrado usando ângulos internos e segmentos de reta.

Fonte: Autores.

No programa 1, apresentado na Figura 3.3, a posição inicial do ator não faz diferença. Em nossa visão esse é o programa mais simples dentre os cinco apresentados. Para elaborá-lo o indivíduo precisa atentar-se as propriedades que caracterizam o quadrado, e refletir sobre o processo de construção desse objeto usando lápis, régua e papel.

Quanto às propriedades, o estudante precisa perceber que o quadrado é um quadrilátero regular, isto é, possui quatro lados de mesmo comprimento e que todos os ângulos internos medem 90º. Pensando no *Scratch*, isso significa que o ator sempre andará uma mesma distância (como exemplo, 100 passos), girará sempre no mesmo sentido (apenas no sentido horário ou anti-horário), o giro sempre terá a mesma medida de graus (90 graus) e esse processo deve se repetir até que os quatro lados sejam desenhados.

Quanto ao processo de construção no papel, o discente precisa perceber que ao desenhar o quadrado numa folha, inicialmente se faz um segmento de reta e a partir de um dos extremos desse segmento constrói um novo segmento de reta com mesmo comprimento do primeiro, de modo que o ângulo entre segmentos seja de 90º, esse processo é repetido até que todos os lados do quadrado sejam construídos. No programa 1 esse processo é representado pela sequência movimento-giro. Além disso, o discente precisa perceber que para desenhar o quadrado ele risca o papel com o lápis, por isso o comando “use a caneta” antes de iniciar a sequência de movimentos é essencial.

O programa 2 difere do programa 1, conforme apresentado na Figura 3.3, nele a sequência movimento-giro é sintetizada usando um comando de repetição. O estudante que desenvolve um programa dessa forma percebe todos os aspectos mencionados no programa 1, ademais identifica a repetição dessa sequência e compreende que ela pode ser representada por meio do comando “repita __ vezes”.

No programa 3 da Figura 3.3, além de perceber todos os aspectos mencionados anteriormente o aluno generaliza o processo de construção, ou seja, o nível de abstração do programa 3 é mais elevado do que nos programas anteriores. Nesse caso, o comando “pergunte” é utilizado para que o usuário possa determinar o comprimento dos lados do quadrado que deseja construir, e o bloco “resposta” é tratado como a variável “comprimento do lado”.

Ao observar a Figura 3.4, visualizamos dois programas que constroem um quadrado, diferente dos programas anteriores estes não utilizam os comandos mova e gire, são baseados nas coordenadas dos vértices do quadrado.

Figura 3.4 - Exemplo de programas que desenham um quadrado usando coordenadas cartesianas.

Fonte: Autores.

O processo de construção dos programas 4 e 5, apresentados na Figura 3.4, é diferente do utilizado nos anteriores, nesses programas a construção do quadrado é pensada sobre o plano cartesiano. Assim, o estudante precisa determinar quatro pontos no plano cartesiano. Suponhamos que estes pontos sejam A, B, C e D, de modo que esses pontos satisfaçam duas condições: $d_{AB} = d_{AC} = d_{BC} = d_{CD}$ e $\hat{A} = \hat{B} = \hat{C} = \hat{D} = 90º$.

No programa 4 da Figura 3.4 esse processo é executado usando comandos “vá para x:__ e y:__”, além disso é preciso definir um ponto inicial para que a caneta não risque o caminho que percorre entre a posição que estava no início do programa e o primeiro ponto do quadrado. No programa 5, esse processo é generalizado utilizando uma variável.

Apesar da variável “resposta” representar o comprimento do lado nos programas 3 e 5, existe uma diferença na forma com que ela é abordada. No programa 3 da Figura 3.3, a variável é associada ao número de passos que o ator precisa percorrer, enquanto no programa 5, da Figura 3.4, ela é associada às coordenadas dos vértices do quadrado, essa diferença pode fazer com que o discente amplie seu entendimento sobre o conceito de variável. No Quadro 3.5 comparamos os programas 1, 2, 3, 4 e 5 em relação aos aspectos algébricos, computacionais e de codificação no *Scratch*.

Quadro 3.5 - Comparação entre os programas que desenham um quadrado.

Referência	*Aspectos algébricos*	*Aspectos computacionais*	*Aspectos de codificação no Scratch*
Programa 1	Representar as propriedades de um quadrado. Testar e mensurar comprimentos. Testar e mensurar ângulos.	Estabelecer sequências lógicas. Estabelecer respostas para a ação do usuário (clicar na bandeira verde).	Mensurar em pixels. Movimentar um ator com base no número de "passos". Usar a caneta. Usar o comando quando a bandeira verde for clicada para iniciar o programa.
Programa 2	Todos os aspectos do Programa 1. Reconhecer padrões.	Todos os aspectos do programa 1. Ciclos computacionais.	Todos os aspectos do programa 1. Usar a função "repetir".
Programa 3	Todos os aspectos do programa 2. Generalizar uma regularidade.	Todos os aspectos do programa 2. Coletar e atualizar dados (variável resposta).	Todos os aspectos do programa 2. Usar o comando "pergunte" para obter dados numéricos. Utilizar o bloco "resposta" como variável numérica.
Programa 4	Representar as propriedades de um quadrado. Calcular a distância entre dois pontos no plano cartesiano.	Todos os aspectos do programa 2.	Mensurar em pixels. Movimentar um ator com base em coordenadas cartesianas. Usar a caneta. Usar o comando quando a bandeira verde for clicada para iniciar o programa.
Programa 5	Todos os aspectos do programa 4. Reconhecer padrões. Generalizar uma regularidade.	Todos os aspectos do programa 3.	Todos os aspectos do programa 4. Usar o comando "pergunte" para obter dados numéricos. Utilizar o bloco "resposta" como variável numérica.

Fonte: Autores.

Analisando o Quadro 3.5 observamos que, apesar de parecer simples, programar um quadrado no *Scratch* exige a mobilização de diversas habilidades algébricas e computacionais. Os programas 1, 2, 3, 4 e 5, da Figura 3.3 e Figura 3.4, mostram que uma mesma tarefa pode ser empregada para trabalhar com diferentes níveis de generalização, os programas 1, 2 e 4 exigem uma generalização aritmética e os programas 3 e 5 generalizações simbólicas. Além disso, o programa 1 aborda os conceitos computacionais sequência e evento, os programas 2 e 4 exploram sequência, evento e ciclo, e os programas 3 e 5 abordam sequência, evento, ciclo e

dados. Em todos os programas instigam as práticas computacionais de iteração, depuração e abstração.

Miller e Larkin (2017) apontam que ao programar o quadrado os educandos podem perceber a "estrutura" desse objeto matemático, pois a atividade exige que as propriedades desse objeto sejam abstraídas e representadas como comandos. Isto é, perceber que o quadrado é um quadrilátero regular (porque possui quatro lados iguais) com todos os ângulos internos iguais a 90º. Apesar de parecer trivial essa estrutura não aparece de forma explícita quando se desenha um quadrado no papel. Além disso, os autores ressaltam que estudar o quadrado dessa forma pode levar os alunos a generalizar e abstrair outras propriedades como o perímetro e a área.

3.3.2.2 Análise sobre a aproximação da raiz quadrada de um número positivo no *Scratch*

A elaboração de um programa que forneça a aproximação da raiz quadrada de um número real positivo qualquer é, em nossa visão, uma tarefa mais complexa do que o desenho de um quadrado. Para elaborar um programa assim o sujeito precisa: conhecer algum método numérico para estimar a raiz quadrada, representar esse método utilizando comandos do *Scratch* e pensar em uma forma de coletar os dados referentes ao número cuja aproximação da raiz quadrada deseja-se determinar.

Para ilustrar essa situação apresentamos na Figura 3.5 e Figura 3.6 um programa que estima a raiz quadrada de um número real positivo qualquer. Na Figura 3.5 são apresentados os comandos gerais do programa e a chamada da rotina denominada "aproximação da raiz quadrada" que será apresentada separadamente.

Figura 3.5 - Exemplo de programa que estima a raiz quadrada de um número inteiro positivo – Parte 1.

Fonte: Autores.

Para que o discente consiga elaborar um programa similar a este, ele precisa pensar em um modo de coletar os dados referentes ao número que o usuário deseja calcular a aproximação da raiz quadrada, para isso usamos o comando “pergunte” fazendo com que o bloco “*resposta*” representasse nossa variável inicial, conforme é apresentado na Figura 3.5.

O comando “aproximação da raiz quadrada” é uma subrotina do código principal e o método de aproximação adotado foi o método de Newton. Nesta parte do programa, utilizamos o comando pergunte para que o usuário pudesse informar o número que deseja descobrir a aproximação da raiz quadrada, a resposta fornecida será o valor inicial da variável “*resposta*”.

Além disso, programamos mensagens de erro caso o usuário insira um valor negativo, como pode ser visto na Figura 3.5, os operadores de comparação foram utilizados como variáveis não-numéricas (verdadeiro/falso). Trabalhar com variáveis não-numéricas pode desafiar e expandir a concepção de função dos estudantes visto que, no contexto escolar, esse conceito usualmente é associado apenas às variações entre quantidades que podem ser representadas em gráfico cartesiano (KILHAMN; BRÂTING, 2019).

O Método de Newton consiste num método iterativo que possibilita calcular a aproximação da raiz quadrada de números reais. Considere uma função diferenciável $f(x)$ e uma aproximação inicial $x^{(0)}$ da raiz $\bar{x}$ de f. Então a sequência gerada pela fórmula recursiva:

$$x^{(k+1)} = x^{(k)} - \frac{f\left(x^{(k)}\right)}{\left(x^{(k)}\right)}, \quad k = 0, 1, 2, \ldots \qquad (1)$$

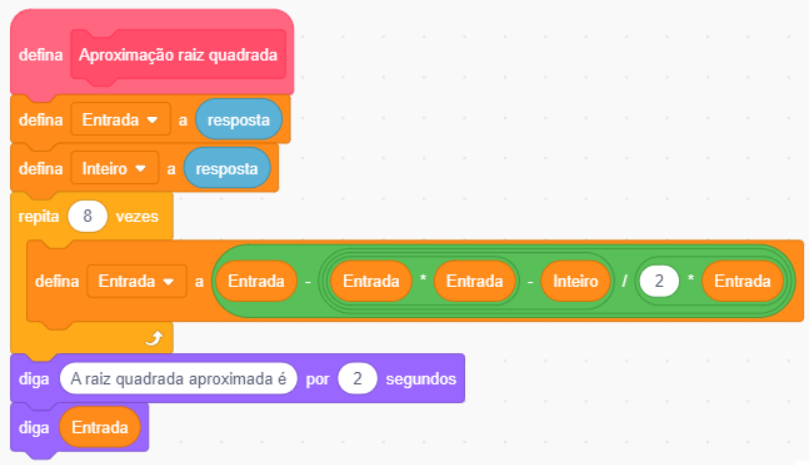
A sequência $x^{(k)}$ convergirá para a raiz de $f(x)$ (BURDEN; FAIRES, 2013).

Para ajustar a equação (1) de modo que possamos calcular a raiz quadrada de um número n, sendo $n \in \mathbb{Z}_+$, basta tomar $x = \sqrt{n}$ assim $n = x^2$, ou seja, $x^2 - n = 0$. Considerando $f(x) = x^2 - n$, temos que $f'(x) = 2x$. Substituindo na equação (1) segue que:

$$x^{(k+1)} = x^{(k)} - \frac{\left(x^{(k)}\right)^2 - n}{2x^{(k)}} \qquad (2)$$

A equação (2) foi utilizada como referência para elaborar o programa apresentado nas figuras 3.5 e 3.6. Os comandos utilizados para encontrar a raiz quadrada com o método de Newton são apresentados na Figura 3.6.

Figura 3.6 - Exemplo de programa que estima a raiz quadrada de um número inteiro positivo – Parte 2.



Fonte: Autores.

Após resolver o problema da coleta de dados, o educando precisa *representar* a equação (2) usando os comandos do *Scratch*, conforme apresentado na Figura 3.5. Observe que a equação (2) tem uma variável "fixa" (n) e uma variável "iterativa" ($x^{(k)}$), por isso criamos uma variável chamada *inteiro* para representar n e outra chamada *entrada* para representar $x^{(k)}$.

Na Figura 3.6, vemos que na primeira iteração a variável *entrada* corresponde à variável *resposta*, para associar as duas variáveis usamos o comando "defina *entrada* a *resposta*" antes que o processo iterativo se inicie. Note que n representará o valor de entrada

em todas as iterações, por essa razão ele foi associado a variável *inteiro*, para isso usamos o comando "defina *inteiro* a *resposta*" antes que o processo iterativo se inicie. O comando "repita 8 vezes" marca o início do processo iterativo, quanto maior o número de vezes que esse processo se repete melhor é a aproximação, a quantidade de repetições foi escolhida arbitrariamente[15].

Para representar a equação (2) usamos os comandos de operadores, conforme mostra a Figura 3.6, destacamos que no *Scratch* colocar um operador dentro de outro funciona de forma similar aos parênteses em álgebra. Por exemplo, inserir uma soma dentro de um operador de multiplicação significa que a soma será resolvida primeiro para então realizar a multiplicação, colocar uma multiplicação dentro de um operador de soma significa que a multiplicação ocorrerá primeiro para então realizar a soma.

Como a variável *entrada* muda a cada iteração utilizamos o comando "defina *entrada* a ..." para garantir que a cada ciclo o valor anterior da *entrada* será atualizado, pois a equação (2) trata-se de uma equação de recorrência.

Na representação algébrica da equação (2) a relação entre a nova aproximação ($x^{(k+1)}$) e a anterior ($x^{(k)}$) é estabelecida usando o sinal de igualdade, porém, no *Scratch* o estabelecimento da relação é feito pelo comando "defina a ...", conforme apresenta a Figura 3.5.

Vale destacar que na álgebra não faz sentido atualizar o valor de uma variável já existente. Considerando a equação $y = x + 1$ pode ser percebido que apesar de ser possível alterar os valores das variáveis x e y, cada par de valores assumidos estaria relacionado a diferentes casos da equação. Por exemplo, se $x = 1$ então $y = 2$, mas se $x = 2$ então $y = 3$. Assim, não podemos atribuir um valor a x e depois substituir esse valor por outro no mesmo caso, isso constituiria uma contradição ($x = 1$ e $x = 2$ implica que $1 = 2$), no entanto isso usualmente acontece na programação quando é preciso fazer um laço ou processo iterativo como o deste exemplo (KILHAMN; BRÂTING, 2019).

Por fim, para informar ao usuário o resultado da aproximação utilizamos os comandos "diga". A aproximação da raiz quadrada do número n informado no começo do programa será

[15] Optamos por repetir o comando 8 vezes para facilitar o processo de elaboração do programa. Contudo, geralmente o número de iterações é definido com base em um critério de parada como $|x^k - x^{k-1}| \leq \mathcal{E}$, sendo $\mathcal{E}$ um número infinitamente pequeno quanto se queira, representando a tolerância do erro.

a variável *entrada* após todas as iterações, por isso o último comando é “diga *entrada*”. O Quadro 3.6 apresenta uma síntese dos aspectos algébricos, computacionais e de codificação no *Scratch* abordados nesse programa.

Quadro 3.6 - Síntese dos aspectos algébricos, computacional e de codificação no Scratch.

Aspectos algébricos	**Aspectos computacionais**	**Aspectos de codificação no *Scratch***
Manipular variáveis numéricas. Manipular variáveis não-numéricas. Generalizar. Ordem de operações. Relacionar variáveis. Iteração.	Estabelecer sequências lógicas. Estabelecer condições com base na ação do usuário e comparações entre números. Ciclos computacionais. Coletar e atualizar dados (variáveis *resposta*, *entrada* e *inteiro*).	Usar a estrutura de controle “repetir” para executar processos iterativos. Usar o comando “se, então” para gerar condicionais lógicas. Usar comando de blocos para modularizar processos. Usar o comando “diga” para informar o usuário. Usar o comando “pergunte” para obter dados numéricos. Utilizar o bloco “resposta” como variável numérica. Utilizar o comando “defina a __” para atualizar variáveis.

Fonte: Autores.

No Quadro 3.6 observamos que o desenvolvimento desse programa trabalha com generalizações simbólicas, exigindo do programador o domínio e entendimento sobre diferentes conceitos algébricos. Percebemos também que esse programa trabalha com todos os conceitos computacionais, além das práticas de iteração, depuração e abstração. Compreendemos que a elaboração e análise de um programa similar a este requer um alto nível de desenvolvimento do PA.

A programação pode proporcionar oportunidades para que o educando identifique, generalize e expresse padrões, além disso pode ampliar compreensões sobre objetos e conceitos matemáticos. Assim, defendemos que ensinar a Matemática via programação pode contribuir para o desenvolvimento do PA e do PC.

CAPÍTULO 4 METODOLOGIAS DE PESQUISA

Neste capítulo apresentaremos aspectos metodológicos relacionados à natureza, classificação e propósito da pesquisa, assim como a descrição dos procedimentos de coleta, tratamento e análise de dados.

4.1 DELINEAMENTO DA PESQUISA

Esta pesquisa é um estudo de caso exploratório de abordagem mista. Conforme Yin (2001) um estudo de caso pode ser entendido como um método empírico que investiga uma situação tecnicamente única. Nesse tipo de pesquisa as variáveis de interesse podem ir além dos dados coletados, ademais o fenômeno é explorado dentro de seu contexto e os limites entre o contexto e o fenômeno não estão bem definidos (YIN, 2001).

Este trabalho objetiva proporcionar maior familiaridade com o problema, por isso pode ser classificada como exploratória (GIL, 2017). Este tipo de pesquisa é adequado para desenvolver uma visão geral acerca de algum fenômeno, e muitas vezes é a primeira etapa para uma investigação mais ampla (MOREIRA; CALEFFE, 2008).

A abordagem mista parte do entendimento de que as abordagens quantitativa e qualitativa podem se complementar e serem utilizadas em conjunto no processo de pesquisa, propiciando outras compreensões sobre os fenômenos educacionais investigados (SOUZA; KERBAUY, 2017).

Creswell e Clark (2007) apontam quatro desenhos metodológicos que podem ser empregados para efetivar a abordagem mista: triangulação, explanatório, exploratório e embutido. O método de triangulação busca comparar e contrastar dados qualitativos com dados quantitativos simultaneamente. No método exploratório os resultados qualitativos são utilizados como auxiliares no desenvolvimento das análises sobre os dados quantitativos.

No método explanatório os dados quantitativos são utilizados para analisar os dados qualitativos ou vice-versa; no método embutido um conjunto de dados qualitativos é utilizado para apoiar dados quantitativos ou vice-versa (CRESWELL; CLARK, 2007). Nesse estudo adotamos o desenho metodológico explanatório.

Os sujeitos dessa pesquisa foram 17 acadêmicos dos quartos anos, integral e noturno, do curso de Licenciatura em Matemática de uma instituição de Ensino Superior do estado do Paraná. Dos 17 discentes, 10 estudavam no turno integral e 7 no noturno.

Os dados empíricos foram coletados no decorrer de uma sequência didática elaborada pelos autores e validada pelo Grupo de Pesquisa de Tecnologias Educacionais em Matemática (GTEM), essa sequência foi implementada em 2018 e será apresentada na seção 4.2. As atividades foram realizadas no horário de aula de três disciplinas: Estágio Curricular Supervisionado em Matemática II, Instrumentação para o Ensino de Matemática IV e Laboratório de Ensino de Matemática, todas relacionadas ao uso de tecnologias digitais na formação docente.

Todos os acadêmicos do quarto ano foram convidados a participar. Os discentes que optaram por não participar como sujeitos de pesquisa puderam realizar as atividades mesmo assim, nesse caso os dados gerados foram descartados. Os discentes que concordaram participar das atividades como sujeitos de pesquisa assinaram um termo de consentimento livre e esclarecido (apresentado no Apêndice A) e cederam seus dados para análise. Na seção seguinte apresentamos a sequência didática implementada.

4.2 SEQUÊNCIA DIDÁTICA

A sequência didática teve carga horária de 16 horas aula, sendo que: 6 horas aulas foram realizadas na disciplina de Estágio II, 4 horas aula na disciplina de Laboratório, 2 horas aula na disciplina de Instrumentação e 4 horas aula correspondem a atividades que os acadêmicos realizaram fora do horário de aula (via *Google Forms*).

Para que os acadêmicos pudessem utilizar o *Scratch* realizamos as atividades presenciais no laboratório de informática do curso de Licenciatura em Matemática. Algumas atividades foram realizadas em duplas e outras de forma individual, optamos por essa configuração para incentivá-los a debater estratégias, conceitos e procedimentos com seus pares nas atividades enquanto produziam algoritmos. No Quadro 4.1 apresentamos a estrutura da sequência didática.

Quadro 4.1 - Sequência didática.

Ordem	Duração	Organização	Descrição[16]
Atividade 1	2 h. aula	Individual	Aplicação do questionário pré-implementação.
Atividade 2	3 h. aula	Dupla	Produção de algoritmos em linguagem materna.
Atividade 3	2 h. aula	Dupla	Produção de um programa que desenhe um quadrado no *Scratch*.
Atividade 4	3 h. aula	Dupla	Produção de um programa que desenhe um triângulo escaleno com comprimentos variáveis no *Scratch*.
Atividade 5	2 h. aula	Dupla	Depuração de programa problemático no *Scratch*.
Atividade 6	1 h. aula	Individual	Aplicação do teste de Pensamento Computacional.
Atividade 7	3 h. aula	Individual	Análise de um programa que calcula a aproximação da raiz quadrada de um número inteiro no *Scratch*.

Fonte: Autores.

As aulas da primeira atividade, foram destinadas para a aplicação do questionário pré-implementação, apresentado no apêndice B. Essa atividade foi realizada presencialmente e os acadêmicos responderam ao questionário via *Google Forms*.

Na segunda atividade, os discentes foram convidados a produzir algoritmos que descrevessem a construção de um quadrado e de um triângulo escaleno. Os sujeitos tiveram liberdade para criar seus algoritmos como quisessem, ou seja, poderiam utilizar qualquer meio semiótico de representação que desejassem. Essa atividade foi realizada presencialmente e em duplas. Para auxiliá-los estabelecemos quatro tarefas que deveriam ser cumpridas. São elas:

Tarefa 1: Qual é a definição de quadrado?

Tarefa 2: Elabore um algoritmo que descreva a construção de um quadrado;

Tarefa 3: Qual é a definição de triângulo escaleno?

[16] Vale diferenciar nosso entendimento sobre algoritmo, programa e código. Algoritmos são sequências finitas de instruções. Programa (ou *script*) refere-se a um algoritmo construído usando a linguagem de programação do *Scratch*. Código refere-se ao conjunto de comandos que formam um programa do *Scratch*, assim, códigos podem ser entendidos como um tipo de algoritmo.

Tarefa 4: Elabore um algoritmo que descreva a construção de um triângulo escaleno com comprimentos variáveis em seus segmentos.

As tarefas 1 e 3 levam os participantes a refletir sobre as propriedades que caracterizam um quadrado e um triângulo escaleno, além de fornecer subsídios teóricos que podem auxiliá-los na elaboração dos algoritmos.

Na terceira atividade do Quadro 4.1, os acadêmicos foram instruídos a produzir um programa no *Scratch* que desenhasse um quadrado. Na quarta atividade foi pedido que elaborassem um programa que desenhe um triângulo escaleno com comprimentos variáveis, essas duas atividades foram realizadas presencialmente e em duplas.

A atividade 3 solicitou que os acadêmicos pensem no desenho de um polígono regular, enquanto na atividade 4 o polígono é irregular, essa mudança reflete na elaboração dos programas. Na atividade 3 não era preciso usar nenhum comando de variáveis, pois os lados do quadrado podem ser constantes, na atividade 4, por outro lado, era necessário que os acadêmicos utilizassem os comandos de variável.

As atividades 1, 2, 3 e 4 foram validadas em duas etapas: a primeira foi um piloto realizado com os membros do GTEM e a segunda foi um piloto realizado com acadêmicos do ano letivo de 2018. Os acadêmicos que participaram do piloto cursavam o quarto ano da licenciatura em Matemática da turma anterior à dos sujeitos desta pesquisa (CORRÊA *et al*., 2018), isto é, nenhum dos sujeitos desta pesquisa participou do piloto.

No piloto fizemos uma introdução ao *Scratch* demonstrando aos participantes como usar comandos de caneta e combiná-los com comandos de movimento. Essas atividades introdutórias aconteceram após as atividades 1 e 2 e antes de apresentarmos as atividades 3 e 4. Nesse experimento observamos que a maioria dos participantes elaboraram seus programas seguindo o mesmo raciocínio que o empregado nas atividades introdutórias.

Em vista disso optamos por não realizar uma introdução ao *Scratch,* para não direcionar os discentes enquanto estes produziam os algoritmos. Partimos do pressuposto que realizar uma introdução poderia induzi-los a seguir a mesma linha de pensamento adotada nas atividades de introdução.

Sendo assim, os acadêmicos não receberam nenhuma instrução sobre as ferramentas e o funcionamento do *Scratch*. Eles foram incentivados a explorar e pesquisar sobre o *software*

por conta própria em todas as atividades. O pesquisador forneceu informações e sugestões apenas quando os participantes não conseguiam progredir apesar de seus esforços.

A quinta atividade, do Quadro 4.1, consistiu em depurar um programa. Essa atividade foi realizada presencialmente e em duplas. O programa depurado é apresentado no apêndice C. Os acadêmicos receberam o programa, explicamos a eles a finalidade do programa e os instruímos a testá-lo com o intuito de localizar e corrigir erros que comprometiam sua execução. Essa atividade foi pensada numa perspectiva formativa, para que os discentes se familiarizassem com as ideias de depuração e análise de códigos.

Na sexta atividade, do Quadro 4.1, os discentes realizaram o teste do PC (BRENNAN, 2012 *apud* BOUCINHA, 2017) via *Google Forms*. Essa atividade foi realizada individualmente e fora do horário de aula. Na sétima e última atividade, os acadêmicos responderam 28 perguntas abertas sobre um programa que calcula a raiz quadrada de um número inteiro, similar ao apresentado no Capítulo 3 Figuras 3.3 e 3.4.

Os discentes receberam o programa e um questionário do *Google Forms*, foram instruídos a analisar o código para responder as perguntas. Tanto o programa quanto o questionário são apresentados no apêndice D. As atividades 6 e 7 foram realizadas individualmente e fora do horário de aula. Na seção seguinte apresentaremos os instrumentos de coleta de dados.

4.3 INSTRUMENTOS DE COLETA DE DADOS

Os instrumentos de coleta de dados utilizados nesta pesquisa foram questionários e produções dos participantes. Utilizamos três questionários, são eles: questionário pré-implementação, teste de PC e análise de códigos. O conjunto de registros escritos e programas do *Scratch* elaborados pelos discentes durante a sequência didática é identificado como produções dos participantes.

Optamos por excluir os dados gerados por alguns instrumentos devido ao volume de dados gerados e pelo escopo da pesquisa. Na Quadro 4.2 apresentamos uma síntese dos instrumentos de coleta de dados considerados para a análise.

Quadro 4.2 - Síntese dos instrumentos de coleta de dados analisados.

ID	Instrumento	Tipo de dado
1	Questionário pré-implementação.	Respostas às perguntas abertas e fechadas. Respostas às perguntas tipo Likert com escala de cinco pontos.
2	Algoritmo da construção de um quadrado (linguagem materna).	Perguntas abertas.
3	Algoritmo da construção de um triângulo escaleno (linguagem materna).	Perguntas abertas.
4	Código da construção de um quadrado (programa).	Projeto no *Scratch*.
5	Código da construção de um triângulo escaleno (programa).	Projeto no *Scratch*.

Fonte: Autores.

O questionário pré-implementação foi utilizado para coletar dados referentes ao perfil e preconcepções dos sujeitos sobre programação, PC e PM. Optamos por questioná-los sobre o PM porque partimos do pressuposto que esse conceito seja mais familiar aos discentes que PA.

Esse questionário é composto por perguntas abertas, fechadas e do tipo *Likert*. Nas questões tipo *Likert* apresentamos asserções seguidas de uma escala de concordância, o respondente aponta qual é seu nível de concordância para aquela afirmação. Esse questionário foi validado pelo GTEM e aplicado via *Google Forms*.

Os itens 2 e 3 do Quadro 4.2 correspondem aos algoritmos em linguagem materna produzidos na atividade 1 e 2, apresentadas na seção 4.1. Esses algoritmos foram produzidos de próprio punho usando comandos inventados pelos próprios discentes. Os itens 4 e 5 correspondem aos programas desenvolvidos pelos discentes no *Scratch* durante as atividades 3 e 4, diferem dos algoritmos em linguagem materna porque os sujeitos precisaram utilizar a linguagem do *Scratch*. Na seção serão apresentados a metodologia de análise de dados que utilizamos nessa pesquisa.

4.4 MÉTODOS DE ANÁLISE DE DADOS

Optamos por utilizar a Análise de Conteúdo de Bardin (2011) para organizar e explorar os dados qualitativos gerados pelos instrumentos de pesquisa. Os dados quantitativos foram analisados por meio de porcentagens simples geradas a partir das frequências das categorias ou

opções. No Quadro 4.3 apresentamos uma síntese dos métodos de análise para cada tipo de dado.

Quadro 4.3 - Síntese dos métodos de análise.

Método de análise	**Tipo de dado**
Análise de conteúdo. Análise sobre a frequência das categorias.	Respostas a perguntas abertas. Algoritmos em linguagem materna. Algoritmos no *Scratch.*
Análise sobre a frequência das alternativas.	Respostas a perguntas fechadas e tipo *Likert*.

Fonte: Autores.

A análise de conteúdo é um conjunto de técnicas que orientam os processos de tratamento e análise de dados, objetivando estabelecer indicadores que permitam ao pesquisador inferir convergências ou incidências nos registros analisados (BARDIN, 2011). Esse processo é composto por três etapas, a saber: pré-análise, exploração do material e tratamento dos resultados e interpretação inferencial.

Na primeira etapa (pré-análise) os objetivos da investigação são formulados e os dados a serem analisados são selecionados. Nessa etapa filtramos os dados obtidos com o intuito de descartar aqueles que não contribuem para responder as questões desta pesquisa.

A segunda etapa (exploração do material) constitui-se na análise inicial sobre os dados qualitativos com o intuito de categorizá-los. Nessa etapa analisamos as respostas dos discentes às perguntas abertas, os algoritmos em linguagem materna e aos programas do *Scratch*. Os algoritmos foram analisados com base em sua estrutura. Todas as categorias geradas neste movimento são emergentes e denominadas categorias iniciais.

Na sequência as categorias iniciais foram reanalisadas e agrupadas quando possível, as categorias geradas nesse processo são as categorias finais apresentadas nessa pesquisa. Na última etapa tratamos os resultados obtidos para facilitar sua discussão e apresentação, além disso realizamos inferências com base em nossos referenciais teóricos.

Os dados quantitativos foram analisados com base nas frequências das alternativas selecionadas nas perguntas fechadas e tipo *Likert*, e das categorias geradas pelos dados qualitativos. Os dados quantitativos serviram para auxiliar as inferências e debates realizados sobre os dados qualitativos. No capítulo seguinte apresentaremos e discutiremos os resultados obtidos nesse estudo.

CAPÍTULO 5 ANÁLISE DE DADOS

Ao todo 17 acadêmicos do curso de Licenciatura em Matemática participaram dessa pesquisa, como já mencionado. No Quadro 5.1 apresentamos quais atividades cada participante de pesquisa realizou. Essas atividades foram detalhadas no Quadro 4.1 da seção 4.1. Os acadêmicos que estudam em turno integral são identificados como *I1, I2, ..., I10*, enquanto os discentes do turno noturno são identificados como *N1*, *N2*, ..., *N7*. Utilizamos o X para indicar quais atividades o sujeito realizou e o símbolo - para apontar quais ele não realizou.

Quadro 5.1 - Relação entre os participantes e atividades da pesquisa.

	I1	***I2***	***I3***	***I4***	***I5***	***I6***	***I7***	***I8***	***I9***	***I10***	***N1***	***N2***	***N3***	***N4***	***N5***	***N6***	***N7***
At 1	X	X	X	X	X	X	X	X	X	X	X	X	X	-	X	X	X
At 2	X	X	X	X	X	X	X	X	X	X	X	X	X	X	-	X	X
At 3	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
At 4	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	-	-
At 5	X	X	X	X	X	X	X	X	X	X	X	X	X	X	-	-	-
At 6	X	X	X	X	X	-	X	-	X	X	X	X	-	X	X	X	X
At 7	X	-	X	X	X	X	X	X	X	-	-	-	X	X	-	-	-

Fonte**:** Autores.

Optamos por analisar os dados dos acadêmicos que realizaram mais do que 60% das atividades. Por esta razão, os dados dos sujeitos *N5*, *N6* e *N7* não foram considerados nesta análise.

Tendo em vista a natureza e objetivo dessa pesquisa buscamos desvelar as percepções e relações expressas pelos acadêmicos nos registros que realizaram ao longo da sequência didática. Procuramos apresentar os dados de modo a contemplar, com fidelidade, as falas e produções de todos os participantes.

Nesse sentido, as categorias apresentadas nesse capítulo emergem das respostas e programas elaborados pelos discentes. Nas seções seguintes apresentaremos o perfil dos sujeitos de pesquisa e os dados obtidos pelos diferentes instrumentos de coleta.

5.1 ANÁLISE DO QUESTIONÁRIO PRÉ-IMPLEMENTAÇÃO

O questionário pré-implementação objetivou levantar dados sobre o perfil dos participantes e preconcepções dos sujeitos sobre programação, PC, PM e relações entre esses dois tipos de pensamento.

Optamos por aplicar esse instrumento na primeira aula da sequência didática para evitar que as atividades seguintes influenciassem as respostas dos sujeitos. Dos 14 discentes participantes, 1 acadêmico do turno noturno não respondeu este instrumento. Assim, os dados apresentados nessa seção correspondem as respostas fornecidas por 13 discentes.

5.1.1 Perfil dos Sujeitos de Pesquisa

Essa pesquisa foi realizada com uma amostra de 14 discentes do quarto ano do curso de Licenciatura em Matemática. Destes 71% são do sexo feminino e 29% do sexo masculino; 71% frequentam o curso em período integral e 29% no período noturno; 58% possui entre 20 e 22 anos, 21% entre 29 e 33 anos, 14% entre 39 e 41 anos e 7% possuem 49 anos. Observamos que a maioria da amostra é composta por mulheres e estudantes do turno integral. Além disso, notamos que a faixa etária dos sujeitos é ampla, contemplando pessoas de 20 a 49 anos.

A Tabela 5.1 apresenta a relação dos sujeitos com a computação, em particular com a programação. Essa tabela foi construída com base nas respostas às seguintes questões: *Como você avalia seus conhecimentos em informática? Você já programou alguma vez? Como você avalia seus conhecimentos sobre programação? Qual é seu nível de interesse em programação?*

Tabela 5.1 - Relação dos sujeitos com a computação.

(continua)

	Alternativas	***Integral***	***Noturno***	***Total***
Conhecimento sobre informática	Básico	14%	15%	29%
	Intermediário	50%	14%	64%
	Avançado	7%	0%	7%

Tabela 5.1 - Relação dos sujeitos com a computação.

(conclusão)

	Alternativas	***Integral***	***Noturno***	***Total***
Experiência prévia com programação	Sim	36%	0%	36%
	Não	35%	29%	64%
Conhecimento sobre programação	Avançado	0%	0%	0%
	Intermediário	15%	0%	15%
	Básico	28%	0%	28%
	Não tenho nenhuma noção sobre programação	28%	29%	57%
Interesse em programação	Muito interesse	14%	11%	25%
	Algum interesse	43%	0%	43%
	Não sei dizer	14%	18%	32%
	Nenhum interesse	0%	0%	0%

Fonte**:** Autores.

Por meio do Quadro 5.2 observamos que a maioria dos participantes não têm experiência prévia (64%), enquanto 57% relatam não ter nenhuma noção sobre programação. Esses dados corroboram com nosso pressuposto de que a maioria dos licenciandos não tem experiência ou familiaridade com programação.

Os sujeitos com maior domínio sobre a informática, experiência em programar e conhecimento sobre programação se concentram na turma do turno integral, enquanto todos acadêmicos que estudam no período noturno relatam não ter experiência alguma com programação. Notamos também que uma significativa parcela dos sujeitos demonstrou interesse em programar, contudo, vale ressaltar que ao menos 32% da amostra não necessariamente se interessa pelo tema.

Perguntamos aos discentes: *Você já programou em algum ambiente de programação por blocos? Qual?* Identificamos que 23% dos sujeitos já utilizaram um ambiente de programação por blocos, sendo todos acadêmicos do turno integral, destes discentes 16% conhecem o *Scratch*, 8% o Mblock e 8% o Appinventor.

A partir dessas constatações estabelecemos três perfis de sujeitos, apresentados no Quadro 5.3. Optamos por categorizá-los de acordo com seu conhecimento prévio e interesse sobre o tema porque supomos que aqueles com interesse podem apresentar um

comprometimento maior com as atividades e, consequentemente, produzir algoritmos mais criativos ou robustos.

Quadro 5.2 - Perfil dos sujeitos de pesquisa de acordo com conhecimento e interesse por programação.

Identificação	***Descrição***	***Sujeitos***
Sem conhecimento e com interesse	Sujeitos que declaram não ter conhecimento ou experiência com programação, mas demonstram ter interesse sobre o tema.	I3, I4, I7, I8, N1.
Com conhecimento e interesse	Sujeitos que possuem conhecimento sobre programação e demonstram interesse pelo tema.	I1, I2, I5, I9.
Interesse Não Esclarecido	Sujeitos que cujo interesse por programação não está claro.	I6, I10, N2, N3, N4.

Fonte**:** Autores.

Analisando o Quadro 5.2 percebemos que a maioria dos sujeitos do noturno não tem o interesse esclarecido, isto é, podem ou não ter interesse sobre a temática. Ademais relacionamos o sujeito N4 a esse perfil, pois como esse acadêmico não respondeu o questionário pré-implementação entendemos que seu interesse não está claro. Na seção seguinte apresentamos os resultados e análises sobre as outras questões do primeiro questionário.

5.1.2 Concepções Sobre Computação e Programação

Nessa subseção apresentamos os dados relacionados às concepções dos sujeitos sobre computação e programação. Optamos por questioná-los sobre esses temas para investigar quais conhecimentos prévios os sujeitos possuem sobre programação, bem como qual é sua relação com a computação e suas opiniões sobre a abordagem desses tópicos na Educação Básica.

Destacamos que as frequências apresentadas nessa seção foram calculadas em relação ao total de ocorrência das categorias, não em relação ao total de sujeitos. Ou seja, calculamos a frequência das categorias da seguinte forma $\frac{ocorrências\ da\ categoria}{total\ de\ ocorrências}$.

Para investigar qual é a concepção dos sujeitos sobre programação perguntamos: *O que você entende por programação?* As respostas foram compiladas nas seguintes categorias: É uma forma escrita de gerar programas e aplicativos (39%); escrever comandos com uma

linguagem específica para atingir um determinado objetivo (23%); É um meio de ensinar o computador (15%) e não sei dizer (23%).

Segundo Wirth (1989) programar é a técnica ou arte de elaborar algoritmos de forma sistemática. Partindo dessa concepção percebemos que os sujeitos têm uma visão ingênua e restrita do que é programar, visto que suas concepções não mencionam algoritmos e se restringem às ações envolvendo computadores.

Isso era esperado tendo em vista que, usualmente, a programação não é abordada nos cursos de Licenciatura em Matemática. A única disciplina que trabalha um pouco com programação, Cálculo Numérico, é ofertada no último ano do curso dos sujeitos que participaram da pesquisa, e estes não tiveram contato com as noções preliminares de programação antes de responderem o questionário.

Para investigar a relação dos discentes com a computação utilizamos perguntas do tipo Likert, as asserções e respostas são apresentadas na Tabela 5.2.

Tabela 5.2 - Relação dos discentes com a computação.

Asserção	Concordo	Neutro	Discordo
Acho ciência da computação interessante.	84%	16%	0%
O desafio de resolver problemas usando conhecimentos da ciência da computação me estimula.	76%	24%	0%
Tenho facilidade para aprender a trabalhar com um novo software.	69%	23%	8%
Tenho competência para resolver problemas cotidianos usando computadores.	77%	15%	8%

Fonte**:** Autores.

A partir da Tabela 5.2 observamos que a maioria dos discentes demonstra interesse pela computação. Esses dados sugerem que uma parcela significativa pode se sentir desafiada e interessada nas atividades da sequência didática. No entanto, ao menos 16% dos discentes não opinaram sobre a primeira asserção e 24% sobre a segunda, indicando que esses acadêmicos podem não se sentir interessados, mas também não necessariamente apresentam aversão à computação.

Observamos que 69% da amostra relata ter facilidade para aprender a usar um novo *software*, além disso, também constatamos que 77% dos discentes afirmam ter competência para resolver problemas usando computadores. Apesar da autoavaliação dos discentes não

necessariamente corresponder a sua real capacidade, entendemos que ela pode ser interpretada como um indicativo da autoconfiança que o sujeito tem em relação a essas competências.

Uma parcela significativa dos sujeitos apresenta uma pré-disposição para aprender a utilizar o *Scratch*. Entretanto, para uma minoria as atividades propostas na sequência didática podem ter sido mais onerosas, considerando que 8% relatam ter dificuldade para trabalhar com *softwares* desconhecidos e 8% não acreditam ter competência para resolver problemas usando computadores.

Vale ressaltar que a confiança do sujeito em conseguir aprender a usar um novo *software* pode refletir na qualidade dos programas produzidos pelos discentes, tendo em vista que a metodologia adotada exige que os participantes explorem o *software* por conta própria.

Para investigar quais relações os discentes percebem entre computação e Matemática perguntamos: *Para você a Matemática e a computação estão relacionadas? Como?*

Dos 13 respondentes, 92% afirmaram que há relações entre esses dois campos de conhecimento e 8% não souberam dizer se essas áreas se relacionam. As relações observadas pelos discentes foram: a computação se baseia na programação e a programação utiliza raciocínio lógico e matemática (45%); o computador pode ser utilizado para ensinar matemática (45%); e estão relacionadas através de campos como robótica, programação e simulação (10%).

Para alguns discentes a programação é apresentada como um exemplo de como essas áreas estão associadas, essa concepção é coerente considerando que Kilham e Brâting (2019) apontam diversas relações entre a matemática e a programação. Contudo, para uma parcela dos participantes a relação entre as áreas se dá porque os computadores podem ser utilizados para ensinar matemática. Entendemos que esta é uma concepção simplória, pois a computação não é a finalidade para a qual um computador é utilizado, mas sim um campo de estudo com conceitos, práticas e teorias.

Questionamos também: *Você vê possibilidades para trabalhar a matemática por meio da programação? Como?* 38% dos sujeitos percebem possibilidades e 62% afirmaram que não percebem nenhum meio. As metodologias apontadas pelos discentes são apresentadas na Tabela 5.3.

Tabela 5.3 - Meios para ensinar a Matemática usando a programação.

Id.	***Categoria***	***Concordo***
(a)	Desenvolvendo a lógica por meio da programação.	27%
(b)	Fazendo os alunos explorarem programas para entender a lógica e a matemática que permeia esses programas.	9%
(c)	Por meio da criação de jogos.	9%
(d)	Estimulando os alunos a criar programas que calculem operações e expressões matemáticas.	9%
(e)	Não percebo possibilidades.	46%

Fonte**:** Autores.

As categorias (a) e (b), da Tabela 5.3, vão ao encontro da fala de Selby (2014). Segundo Ávila *et al.* (2017) a categoria (c) é uma estratégia comumente adotada. A categoria (d) pode ser considerada superficial, pois o programa não precisa ter essa limitação para que aborde a matemática, uma vez que o próprio ato de programar mobiliza conhecimentos e habilidades matemáticas. Destacamos que todos os discentes que não percebem meios de trabalhar a matemática por meio da programação também relataram não ter experiência prévia com a programação.

Perguntamos: *Você já vivenciou alguma experiência prévia relacionando conceitos de programação com a matemática em sala de aula? Se sim, em que contexto?* Obtivemos como resposta que 38% dos discentes já vivenciaram e 62% não. Dos acadêmicos que relataram ter experiência prévia observamos que 66% participaram de atividades desenvolvidas na disciplina de Estatística, estes discentes relatam ter utilizado o *software MatLab* para representar gráficos e conceitos estatísticos. Das respostas dadas, foi ainda possível observar que 17% apontaram o estudo de lógica em uma disciplina do curso como experiência prévia, apesar da programação fazer uso de conceitos e operadores lógicos entendemos que o estudo dessa área do conhecimento não é o equivalente a programar.

Além disso, 17% relatam que utilizaram programação para desenvolver projetos com o *GeoGebra* e aplicativos de realidade aumentada. Destacamos que estes discentes participaram de projetos de iniciação científica sobre o uso de realidade aumentada para o ensino de Matemática, sendo que em seus projetos os acadêmicos utilizaram aplicativos prontos e não chegaram a programar novos aplicativos.

Vale ressaltar que uma parcela dos discentes que relataram ter experiência prévia provavelmente não tem clareza sobre o que é programar, visto que alguns apontam o estudo da lógica e o uso de aplicativos de realidade aumentada como experiências com programação.

Diante dos dados apresentados nessa seção constatamos que a maioria dos discentes apresenta uma concepção superficial sobre o que é programação e não tem experiência prévia com esse tema. Além disso, observamos que uma parcela significativa relata interesse sobre a computação e apresenta uma pré-disposição para aprender a usar o *Scratch* por conta própria.

Destacamos também que uma minoria dos participantes relata ter dificuldades para lidar com computadores, além disso outra parcela minoritária declara ter conhecimento e experiência prévia com programação.

5.1.3 Concepções Sobre Pensamento Computacional

Nessa subseção apresentaremos os dados relacionados as concepções dos sujeitos sobre PC. Para levantá-los questionamos os acadêmicos quanto seu entendimento sobre PC, conceitos e práticas computacionais propostas por Brennan e Resnick (2012). Destacamos que as frequências apresentadas nessa seção foram calculadas em relação ao total de ocorrência das categorias, não em relação ao total de sujeito.

Para levantar as preconcepções dos discentes perguntamos: *O que você entende por Pensamento Computacional?* As respostas a essa questão foram compiladas em 7 categorias, classificadas em duas classes de concepções: ingênuas e coerentes. Na sequência apresentamos as concepções consideradas ingênuas.

Usar linguagens de programação para resolver problemas de computador (15%). Segundo Mannila *et al.* (2014) o PC pode ser utilizado para resolver problemas em diversos contextos que não necessariamente se relacionam a programação ou ao uso de computadores, por isso entendemos essa visão como superficial.

Pensar sobre linguagens de programação (15%). Essa perspectiva restringe o PC à programação e, segundo Wing (2006), PC trata de conceituação e não programação. *É a forma com que o computador é programado (15%),* consideramos essa concepção como rasa pelo mesmo argumento apresentado na categoria anterior.

Entender como funciona o computador (8%). Entendemos essa visão como superficial, como já mencionado, pois o PC é um processo cognitivo de resolução de problemas que pode ser desenvolvido e aplicado sem o auxílio do computador. Partindo dessa premissa um sujeito pode ter um PC desenvolvido e ainda assim não conhecer o funcionamento de um computador.

A classe das concepções coerente é composta por 3 categorias, elas são apresentadas na sequência. *Ser capaz de usar conhecimentos computacionais em situações do cotidiano (8%),* é coerente com nossa base teórica, considerando PC como um processo cognitivo associado à solução de problemas (ROMERO; LEPAGE; LILLE, 2017) e a conceitos, práticas e perspectivas computacionais (BRENNAN; RESNICK, 2012).

Pensar logicamente a partir da leitura de comandos e procedimentos (8%). O PC envolve a produção, depuração e análise de algoritmos. Essa concepção foca na depuração que, segundo Valente (1997), consiste na análise de um algoritmo em termos de coerência lógica dos comandos, efetividade das ideias e estratégias empregadas.

Sequência de etapas e execução de rotinas pré-programadas (8%). Segundo Brackmann *et al*. (2017) a produção de algoritmos é um dos pilares do PC. No entanto, ressaltamos que esses algoritmos não necessariamente precisam ser elaborados por meio linguagens de programação ou voltados exclusivamente para programas de computador. Isto é, os algoritmos podem ser expressos por vários meios semióticos e executados por humanos ou computadores (WING, 2014).

Observamos ainda que 23% dos discentes relataram não saber como definir PC. Ademais, ressaltamos que 53% das concepções foram consideradas ingênuas. Esses dados corroboram com nosso pressuposto de que a maioria dos sujeitos desconhece o que é PC. Observamos também que houve concepções coerentes com nosso referencial teórico (24%), indicando que uma minoria da amostra tem noções do que seja PC.

Para investigar a concepção dos sujeitos sobre algumas das dimensões do PC apontadas por Brennan e Resnick (2012) perguntamos: *Você conhece algum dos seguintes termos? Se sim, qual é sua compreensão sobre eles?*

Organizamos os termos apresentados aos sujeitos em dois conjuntos, conforme Quadro 5.3. O primeiro engloba aqueles que são utilizados na Computação e Matemática, são eles: *Sequência/algoritmo; Condições Lógicas; Operadores; Dados e Processos iterativos*. Além

destes, optamos por questioná-los sobre *Variável*. O segundo reúne termos mais específicos à computação, são elas: *Ciclo (loops); Execução em paralelo; Evento; Depuração e Modularização*.

Quadro 5.3 - Termos apresentados aos discentes.

Classificação	**Conceitos**
Computacionais e Matemáticos	Sequência ou algoritmo Condições Lógicas Operadores Dados e Processos iterativos Variável
Computacionais	Ciclo (loops) Execução em paralelo Evento Depuração Modularização

Fonte**:** Autores.

Por meio da classificação dos termos do Quadro 5.3, apresentamos inicialmente a análise daqueles classificados nesta pesquisa como *Computacionais e Matemáticos*. *Sequência* ou *algoritmo* pode ser entendido como um conjunto de passos ordenados que descreve uma ação ou processo de modo que possa ser executado por um agente de informação (BRENNAN; RESNICK, 2012). Observamos sete categorias nas respostas dos discentes sobre esse termo, essas categorias são apresentadas na Tabela 5.4.

Tabela 5.4 - Concepções sobre o termo *Sequência/algoritmo*.

Classificação	**Categoria**	**Frequência**
Coerente	Sequência de etapas para se atingir determinado objetivo.	29%
	Cadeia de processos lógicos que gera um determinado resultado.	7%
Parcialmente coerente	Comandos de programação.	7%
	Forma de resolver uma operação ou uma atividade.	7%
Incoerente	Sequência de números.	14%
	Operações finitas executáveis.	7%
-	Desconheço o termo.	29%

Fonte**:** Autores.

Um algoritmo consiste numa sequência finita de passos, enquanto os comandos de programação são ações individuais que podem ser utilizadas para compor um algoritmo. Deste modo, os comandos de programação estão associados ao conceito de algoritmo, porém, não o

representam. Por isso consideramos a categoria *comandos de programação* como parcialmente coerente.

Em um sentido amplo, algoritmos representam um meio de se fazer algo. Entretanto, analisando com maior rigor percebemos que um algoritmo consiste em uma sequência de ações necessária para realizar uma tarefa definidas de forma clara e precisa. Por esse motivo consideramos a categoria *forma de resolver uma operação ou uma atividade* como parcialmente coerente.

O tópico de sequência numérica é frequentemente abordado no curso de Licenciatura em Matemática, representa a ideia de ordenação entre um conjunto de números. Assim, a categoria *sequência de números* é incoerente com o entendimento de sequência assumido por Brennan e Resnick (2012).

A concepção o*perações finitas executáveis* é vaga, visto que o termo operação pode ser entendido como um cálculo matemático ou uma ação, ou seja, o sentido atribuído a essa palavra influência diretamente na coerência ou não da categoria. Devido a isso essa concepção é considerada incoerente.

O conceito do termo *Sequência/algoritmo* permeia a Matemática, sendo frequentemente abordado de forma implícita em diversas disciplinas do curso de licenciatura. Destacamos que ainda assim 29% dos discentes afirmaram desconhecê-lo, esse dado pode ser um indicativo de que alguns discentes têm dificuldade em identificar e definir conceitos matemáticos fundamentais que supomos ser familiares aos sujeitos.

Brennan e Resnick (2012) relatam que *Condição lógica* pode ser entendido como uma restrição à execução de sequências para gerar resultados diferentes com base em determinadas condições. Quanto ao termo condição lógica, constatamos quatro categorias nas falas dos sujeitos, elas são apresentadas na Tabela 5.5.

Tabela 5.5 - Concepções sobre o termo *Condição lógica.*

Classificação	**Categoria**	**Frequência**
Coerente	Condição para decisão.	15%
Parcialmente Coerente	Hipótese assumida antes de resolver uma operação.	8%
Incoerente	Estruturação lógica de um algoritmo.	8%
-	Desconheço o termo.	69%

Fonte**:** Autores.

Uma hipótese é uma afirmação que pode ou não ser verdadeira, as hipóteses são utilizadas para elaborar condições lógicas, no entanto, apesar de conectados esses conceitos não são equivalentes. Por isso, também consideramos a concepção *hipótese assumida antes de resolver uma operação* como parcialmente coerente.

O conceito do termo *Condição lógica*, representa circunstâncias que são consideradas para escolher entre diversas opções. A estrutura do algoritmo é um conceito diferente, refere-se à coerência lógica entre o conjunto de ações ordenadas utilizado para compor o algoritmo, assim a categoria e*struturação lógica de um algoritmo* é considerada como incoerente.

Esse conceito é estudado com profundidade em uma disciplina do primeiro ano do curso, Fundamentos da Matemática, e é amplamente utilizado no decorrer do curso, ainda assim, observamos que 69% dos discentes afirmaram desconhecê-lo.

Quanto ao conceito do termo *Operadores*, que é definido por Brennan e Resnick (2012) como operações matemáticas, lógicas e de *strings* (cadeias), observamos quatro categorias. A Tabela 5.6 apresenta a síntese dessas categorias.

Tabela 5.6 - Concepções sobre o termo *Operadores*.

Classificação	**Categoria**	**Frequência**
Parcialmente coerente	Símbolos utilizados para gerar condições lógicas.	8%
Incoerente	Mecanismos para alterar um processo.	8%
	Pessoas que operam um computador ou programa.	8%
-	Desconheço o termo.	76%

Fonte**:** Autores.

Um operador pode ser interpretado como um processo que altera uma informação por meio de um algoritmo pré-definido. Como exemplo citamos a soma, uma operação aritmética que reúne duas quantidades e resulta em uma terceira, nesse caso o símbolo da soma é o signo utilizado para indicar qual operação foi realizada sobre dois ou mais números.

Embora os operadores sejam representados por símbolos, o operador e o símbolo que o representa são distintos. O operador pode ser entendido como o algoritmo de manipulação das informações (numéricas ou não-numéricas), o símbolo é o signo que indica qual operação deve ser realizada entre determinadas informações. Além disso, existem diversos tipos de operadores que podem ser utilizados para atingir diversos fins, isto é, eles não estão restritos apenas à

lógica. Por essas razões consideramos que a categoria *símbolos utilizados para gerar condições lógicas* é parcialmente coerente.

Entendemos que os operadores são processos que podem ser empregados para modificar informações, assim, dizer que operadores são *mecanismos para alterar um processo* é redundante, por isso consideramos essa categoria como incoerente. A categoria *pessoas que operam um computador ou programa* é incoerente porque nela operador é entendido como um sujeito que realiza algo, ou seja, não tem relação com a ideia de operador matemático.

Constatamos que 76% dos discentes afirmaram desconhecer o termo. Supomos que isso pode ter ocorrido porque enunciamos o conceito apenas com o termo operador, o que pode ter gerado ambiguidade, uma vez que no contexto dos sujeitos esse conceito costuma ser enunciado como "operador matemático" ou "operações matemáticas".

Brennan e Resnick (2012) definem dados como o armazenamento, recuperação e atualização de informações. As definições dos discentes relativo ao termo *Dados* foram compiladas em quatro categorias que são apresentadas na Tabela 5.7.

Tabela 5.7 - Concepções sobre o termo *Dados*

Classificação	**Categoria**	**Frequência**
Coerente	Informações obtidas ou inseridas.	35%
	Informações armazenadas pelo sistema.	18%
Parcialmente coerente	Informações importantes de um problema.	23%
Incoerente	O que queremos que essa função realize.	6%
-	Desconheço o termo.	18%

Fonte**:** Autores.

Consideramos as categorias *informações obtidas ou inseridas* e *informações armazenadas pelo sistema* como coerentes porque são compatíveis com a definição de Brennan e Resnick (2012).

Na Educação Matemática existe uma metodologia de ensino baseada na Resolução de Problemas. Nesse campo usualmente o primeiro passo na abordagem de um problema é identificar na situação, informações que podem ser úteis para elaborar uma solução. Presumimos que esse é um dos motivos pelo qual os discentes associam dados a coleta de *informações importantes de um problema.*

Contudo, confrontando essa concepção com a definição de Brennan e Resnick (2012) percebemos que essa categoria é parcialmente coerente, pois o conceito de dados também pode abranger informações irrelevantes. A categoria *o que queremos que a função realize* é incoerente, pois foge da definição de dados e sugere que os dados e o objetivo de uma função são a mesma coisa.

A categoria *desconheço o termo* teve frequência de 18%. Consideramos essa porcentagem alta tendo em vista que o conceito de *dados* é abordado de forma explícita em praticamente todas as disciplinas do curso.

Processos iterativos são definidos como processos repetitivos e adaptativos usados para projetar e implementar soluções passo a passo (BRENNAN; RESNICK, 2012). Estabelecemos quatro categorias a partir das falas dos sujeitos sobre *Processos iterativos*, essas categorias são apresentadas na Tabela 5.8.

Tabela 5.8 - Concepções sobre o termo *Processos iterativos*

Classificação	**Categoria**	**Frequência**
Coerente	Cadeia de operações onde a partir da segunda, recorremos a anterior para atingir a próxima	8%
Parcialmente coerente	Processo em que há um procedimento baseado no próprio processo.	8%
Incoerente	Sequência de operações que são realizadas para resolver problemas.	8%
-	Desconheço o termo.	76%

Fonte**:** Autores.

O que diferencia um processo iterativo de uma sequência comum é a constante repetição e adequação da sequência com base em seu estado anterior. A categoria *processo em que há um procedimento baseado no próprio processo* é considera parcialmente coerente porque não ressalta o aspecto repetitivo. De forma similar, a repetição e adequação não são contemplados na categoria *sequência de operações que são realizadas para resolver problemas.*

Os processos iterativos costumam ser abordados na disciplina de Cálculo Numérico, no último ano do curso. Antes dessa disciplina dificilmente esse conceito é explorado de forma explícita. Na época em que o questionário foi implementado os discentes ainda não haviam estudado esse tópico, supomos que essa foi uma das razões para que 76% dos sujeitos afirmassem desconhecer o termo.

Brennan e Resnick (2012) não definem *variável* como um dos conceitos do PC, entretanto, Kilham e Brâting (2019) apontam que esse termo é comum ao PC e PA. Em vista disso, optamos por investigar o entendimento dos discentes sobre esse termo.

Vale ressaltar que o conceito de variável tem múltiplas facetas e aspectos. Vale relembrar que segundo Usiskin (1988) as variáveis podem ser usadas para: (i) generalizar padrões aritméticos; (ii) representar valores constantes desconhecidos; (iii) representar argumentos ou parâmetros; (iv) representar objetos arbitrários em uma estrutura de relações e (v) indicar um endereço de registro de memória.

Além disso, Kilham e Brâting (2019) apontam que no contexto da computação o conceito de variável também pode representar valores não-numéricos, como verdadeiro ou falso. As concepções dos sujeitos sobre *variável* foram compiladas em oito categorias e são apresentadas na Tabela 5.9. Quatro dessas categorias foram consideradas coerentes porque se aproximam da definição adotada em nosso referencial teórico.

Tabela 5.9 - Concepções sobre o termo *Variável.*

Classificação	**Categoria**	**Frequência**
Coerente	Grandeza que desconhecemos.	15%
	É um "valor" que pode ser diferente em determinados contextos.	15%
	Símbolo que representa elementos de um conjunto dado.	8%
	É um nome dado a uma posição de memória à qual é atribuído um valor.	8%
Incoerente	Possibilidades.	8%
	Podem ser objetos.	8%
-	Não sei como definir em palavras.	8%
	Desconheço o termo.	30%

Fonte**:** Autores.

Sobre a categoria *podem ser objetos*, destacamos que uma variável representa um objeto, não é o objeto em si, além disso, não temos como inferir o entendimento dos sujeitos devido à falta de clareza na descrição apresentada, por essas razões essa concepção foi avaliada como inconsistente. Uma variável representa um conjunto de possibilidades, entretanto, as falas que compõem a categoria *possibilidades* são vagas demais para inferir sobre o entendimento dos discentes, por isso também consideramos essa concepção como incoerente.

Encaixamos as falas de sujeitos que relatam conhecer o conceito, mas tem dificuldade em defini-lo na categoria *não sei como definir em palavras*. Além disso, destacamos que 30% dos acadêmicos afirmaram desconhecer esse termo. Esse percentual alto é preocupante,

considerando que esse conceito é fundamental na Matemática e abordado em praticamente todas as disciplinas do curso.

Apresentamos na sequência a análise dos termos *Computacionais* do Quadro 5.3. *Ciclo* (ou *loops*) é definido como um mecanismo que utilizado para executar uma mesma sequência várias vezes (BRENNAN; RESNICK, 2012). Observamos três categorias nas falas dos sujeitos, na Tabela 5.10 apresentamos todas as categorias.

Tabela 5.10 - Concepções sobre o termo *Ciclo* (ou *loops*).

Classificação	**Categoria**	**Frequência**
Coerente	Processos que se repetem	15%
	Comandos que são executados mais de uma vez	8%
-	Desconheço o termo.	77%

Fonte**:** Autores.

Duas categorias da Tabela 5.10 foram consideradas coerentes porque vão ao encontro do nosso referencial teórico. O conceito de ciclos é explorado em uma disciplina do quarto ano da Licenciatura, Cálculo Numérico, no momento dessa pesquisa os sujeitos ainda não tinham estudado esse tópico. Contudo, a ideia de ciclos permeia o senso comum. Considerando que o entendimento desse conceito no senso comum se aproxima ao entendimento da área da computação, consideramos o percentual da categoria *desconheço o termo* como alto.

Execução em paralelo é definido como coocorrência de múltiplas sequências. Na Tabela 5.11 apresentamos as três categorias observadas nas falas dos discentes.

Tabela 5.11 - Concepções sobre o termo *Execução em paralelo.*

Classificação	**Categoria**	**Frequência**
Coerente	Múltiplas funções que são executadas ao mesmo tempo.	8%
Parcialmente Coerente	Execução simultânea.	8%
-	Desconheço o termo.	84%

Fonte**:** Autores.

Consideramos a categoria *múltiplas funções que são executadas ao mesmo tempo* como coerente porque o sujeito que formulou essa resposta relatou experiência prévia com programação. Supomos que a palavra função pode ter sido empregada porque o respondente não conseguiu pensar em uma mais apropriada, ou porque dentro do campo da computação o termo função seja utilizado para indicar sequências de comandos.

Observe que a categoria *execução simultânea* não explicita os objetos que estão sendo executados simultaneamente, isto é, não deixa claro que os objetos são múltiplos algoritmos. Por essa razão consideramos essa categoria como parcialmente coerente.

Esse conceito não é comum na matemática ou no senso comum, por isso supomos que os acadêmicos tiveram certa dificuldade em defini-lo. Essa percepção é corroborada pela alta porcentagem de sujeitos que afirmaram desconhecer o termo.

Evento é definido como um gatilho para a execução de determinadas sequências (BRENNAN; RESNICK, 2012). As definições apresentadas pelos discentes foram compiladas em quatro categorias e são apresentadas na Tabela 5.12.

Tabela 5.12 - Concepções sobre o termo *Evento*.

Classificação	**Categoria**	**Frequência**
Coerente	Um acontecimento externo que resulta em uma ação pré definida.	8%
Incoerente	Momento de decisão.	8%
	Número de vezes que determinado acontecimento ocorre.	8%
-	Desconheço o termo.	76%

Fonte**:** Autores.

Um gatilho é algo que inicia um processo ou reação, a categoria *um acontecimento externo que resulta em uma ação pré-definida* lembra a ideia de gatilho por essa razão a consideremos coerente.

A categoria *momento de decisão* foi considerada inconsistente porque não temos como inferir o entendimento dos sujeitos a partir da descrição fornecida. A categoria *número de vezes que determinado acontecimento ocorre* é incoerente porque se refere a frequência do evento, não ao evento em si.

Destacamos que a frequência da categoria *desconheço o termo* é alta. Dentro da licenciatura em Matemática esse conceito costuma ser abordado na disciplina de Estatística e Probabilidade, por essa razão essa porcentagem nos surpreende. Esperávamos que os discentes relatassem conhecer o termo e ao menos atribuir a ele o sentido que assume na estatística, mesmo que esse significado seja distinto do entendimento dessa palavra na computação.

Segundo Brennan e Resnick (2012) depuração é um processo de tentativa e erro para testar e corrigir erros em um algoritmo, como já mencionado. Identificamos três categorias em relação ao termo *Depuração* elas são apresentadas na Tabela 5.13.

Tabela 5.13 - Concepções sobre o termo *Depuração*.

Classificação	**Categoria**	**Frequência**
Coerente	Fazer *debug* em um programa para encontrar possíveis erros.	8%
Incoerente	Limpar, excluir ou parar.	8%
-	Desconheço o termo.	84%

Fonte**:** Autores.

*F*azer *debug* significa executar um processo de depuração, por isso a categoria *fazer debug em um programa para encontrar possíveis erros* foi considerada coerente. A categoria *limpar, excluir ou parar* foi classificada como ingênua porque o processo de depuração não objetiva exclusão, mas sim correção.

Destacamos que a frequência do desconheço também é alta em relação a esse termo. A palavra depuração não é comum no curso de licenciatura em Matemática, costuma ser utilizada na disciplina de Cálculo Numérico no quarto ano. Supomos que, na época da coleta de dados, esse conceito ainda não tivesse sido discutido no curso, pois os discentes estavam iniciando essa disciplina.

Modularizar significa construir um sistema grande unindo conjuntos de sistemas pequenos e mais fáceis de manipular (BRENNAN; RESNICK, 2012). Observamos duas categorias nas falas dos sujeitos. A primeira é *Fracionamento de um programa em programas mais simples (8%)* que vai ao encontro da definição que adotamos. A segunda categoria é *desconheço o termo (92%)*, supomos que essa porcentagem é alta porque essa palavra não costuma ser utilizada na Matemática.

Na Tabela 5.14 apresentamos uma síntese das categorias sobre as concepções a respeito dos conceitos computacionais e matemáticos.

Tabela 5.14 - Síntese da análise das concepções dos discentes.

Classificação	Termo	Coerente	Parcialmente coerente	Incoerente	Desconhece
Computacionais e Matemáticos	Sequência / algoritmo	36%	14%	21%	29%
	Condições lógicas	15%	8%	8%	69%
	Operadores	0%	8%	16%	76%
	Dados	53%	23%	6%	18%
	Processos iterativos	8%	8%	8%	76%
	Variáveis	46%	0%	16%	38%
Computacionais	Ciclos (*loops*)	23%	0%	0%	77%
	Execução em paralelo	8%	8%	0%	84%
	Evento	8%	0%	16%	76%
	Depuração	8%	0%	8%	84%
	Modularização	8%	0%	0%	92%

Fonte**:** Autores.

A partir da Tabela 5.14 percebemos que os conceitos *Computacionais e Matemáticos*, em geral, apresentam porcentagens de coerência parcial ou total maiores que os conceitos específicos à computação. Em particular, os termos com maiores porcentagens de coerência foram *Dados* (76%), *Sequência/algoritmo* (50%) e *Variáveis* (46%).

Embora os termos *Computacionais e Matemáticos* sejam amplamente explorados ao longo do curso de Licenciatura em Matemática, com exceção do conceito de processos iterativos que costuma ser abordado no último ano do curso, apenas no conceito *Dados* as concepções com coerência parcial ou total foram mais frequentes que as incoerentes.

Além disso, observamos que uma significativa parcela dos discentes afirmou desconhecer conceitos fundamentais da Matemática. Supomos que os discentes que relataram desconhecer os conceitos *Computacionais e Matemáticos* o fizeram porque tem dificuldades em reconhecer esses conceitos em seu cotidiano, ou têm dificuldade em definir esses conceitos.

Em relação aos conceitos *Computacionais* observamos que a maioria dos discentes relatou desconhecer ou apresentou concepções incoerentes. Contudo, notamos que uma minoria dos sujeitos apresenta um entendimento coerente em relação a esses termos, constatamos que esses são os discentes que relataram ter experiência prévia com programação.

Apesar de alguns discentes apresentarem noções coerentes sobre PC e sobre os conceitos apontados por Brennan e Resnick (2012), a maioria dos acadêmicos demonstra desconhecer ou ter diversas lacunas conceituais sobre o PC. Esse resultado era esperado, uma vez que esse assunto não costuma ser abordado no curso de Licenciatura em Matemática.

5.1.4 Concepções sobre Pensamento Matemático

Nessa subseção apresentamos os dados relacionados as concepções dos sujeitos sobre PM. A coleta foi realizada por meio de questões abertas, fechadas e escalas Likert. Destacamos que as frequências apresentadas nessa seção foram calculadas em relação ao total de ocorrência das categorias, não em relação ao total de sujeito.

Na primeira questão aberta perguntamos aos discentes: *O que você entende por Pensamento Matemático?* As respostas foram compiladas nas seis categorias apresentadas na Tabela 5.15.

Tabela 5.15 - Concepções sobre *Pensamento Matemático* – questão aberta.

Classificação	Categoria	Frequência
Coerente	Ser capaz de usar conceitos matemáticos no cotidiano.	21%
	Pensar sobre a matemática, suas operações, lógica e sua linguagem.	14%
	Organizar, desenvolver e compreender um raciocínio.	7%
	Fazer uso de hipóteses e argumentos logicamente válidos para comprovar determinadas teses.	7%
Parcialmente coerente	Pensar de forma lógica.	29%
	Saber resolver problemas que exigem cálculos.	22%

Fonte**:** Autores.

A categoria *ser capaz de usar conceitos matemáticos no cotidiano* foi considerada coerente, pois o PM tem uma íntima relação com a resolução de problemas (MELLO, 2015). No entanto, vale ressaltar que como o PM está relacionado a processos cognitivos ele não é limitado aos conhecimentos matemáticos.

Segundo Stenberg e Ben-Zeev (1996) a abstração e linguagem são elementos essenciais do PM (STENBERG; BEN-ZEEV, 1996), assim como a própria matemática, essas ideias corroboram com a categoria *pensar sobre a matemática, suas operações, lógica e sua linguagem* é coerente.

Uma das características do PM é elaborar ideias com base em questionamentos, explicações, hipóteses, teses e inferências, ou seja, desenvolver raciocínios (MONTELEONE; WHITE; GEIGER, 2018), por esta razão as categorias *organizar, desenvolver e compreender um raciocínio* e *fazer uso de hipóteses e argumentos logicamente válidos para comprovar determinadas teses* são coerentes.

Segundo Krantz (2017) o carro-chefe do PM é a lógica, no entanto esse pensamento não se resume à lógica, em vista disso consideramos a concepção *pensar de forma lógica* como parcialmente coerente.

Ser capaz de resolver problemas usando cálculos certamente está relacionado ao PM, contudo, esse pensamento não se restringe a essa habilidade. Portanto, consideramos a categoria *saber resolver problemas que exigem cálculos* como parcialmente coerente.

Constatamos que uma significativa parcela das concepções são coerentes, entretanto, ressaltamos que as respostas individuais apontam apenas aspectos particulares do PM. Assim, podemos inferir que os sujeitos têm noções coerentes sobre esse conceito, porém, elas ainda são limitadas a aspectos específicos.

Na elaboração dos questionários pressupomos que uma parcela significativa dos sujeitos poderia desconhecer o termo PM ou ter dificuldades em expressar seu entendimento. Assim, optamos por complementar a questão aberta anterior com perguntas tipo *Likert*. Deste modo, os sujeitos que não souberam defini-lo poderiam fornecer dados sobre o que imaginavam ser PM. As respostas para as questões tipo *Likert* são apresentadas na Tabela 5.16.

Tabela 5.16 - Concepções sobre *Pensamento Matemático* – questões tipo Likert.

(continua)

Id.	Asserção	Concordo	Neutro	Discordo
1	*Pensamento Matemático não está relacionado a intuição ou imaginação.*	23%	8%	69%
2	*Pensamento Matemático se limita a procedimentos e manipulações simbólicas.*	31%	8%	61%
3	*Pensar matematicamente é lembrar e aplicar a regra correta na ordem correta.*	23%	8%	69%

Tabela 5.16 - Concepções sobre Pensamento Matemático – questões tipo Likert.

(conclusão)

Id.	Asserção	Concordo	Neutro	Discordo
4	*Pensamento Matemático é um tipo de pensamento fechado, exato e livre de incertezas.*	16%	15%	69%
5	*Pensamento Matemático envolve abstração e linguagem*	61%	16%	23%
6	*Pensamento Matemático está relacionado a imaginação e a criatividade*	84%	16%	0%
7	*Pensamento Matemático envolve a habilidade de pensar de forma flexível*	100%	0%	0%
8	*Pensamento Matemático envolve utilizar conhecimentos e práticas da matemática para resolver problemas em diversos contextos*	92%	8%	0%

Fonte**:** Autores.

Em nosso entendimento, as asserções 1, 2, 3 e 4 representam uma visão "procedimental", isto é, a concepção de que o PM é limitado aos procedimentos matemáticos e não está relacionado a criatividade. Constatamos que a maioria dos discentes discordou dessa visão, enquanto uma parcela razoável demonstrou concordância.

As asserções 5, 6, 7 e 8 representam uma visão "criativa", ou seja, a concepção de que o PM é um processo criativo que vai além dos procedimentos e conhecimentos matemáticos. Observamos que a maioria dos discentes concorda com essa visão.

Percebemos que 23% dos discentes concordaram com a asserção 1, porém, não houve discordância em relação a asserção 6. Considerando que as asserções 1 e 6 são opostas, esses dados apontam uma contradição na concepção de alguns discentes. Destacamos também que houve consenso em relação a asserção 7.

Em vista dos resultados discutidos nessa subseção percebemos que a maioria dos discentes apresenta noções sobre PM coerentes com nosso referencial teórico. No entanto, a partir das questões tipo *Likert* observamos algumas inconsistências. Uma minoria dos discentes aparenta ter concepção que ora observa o PM sob uma ótica procedimental, ora sob uma visão criativa.

5.1.5 Relações Entre PC e PM

Nessa subseção apresentamos as relações entre PC e PM estabelecidas pelos discentes. Antes de perguntar aos discentes sobre quais relações percebem entre esses tipos de pensamento foi apresentado uma definição de PC e de PM.

A definição de PC apresentada foi: processo de resolução de problemas que abrange análise e sistematização de dados, raciocínio lógico e elaboração de soluções que utilizam algoritmos, além da capacidade de lidar com problemas em aberto (WING, 2014).

A definição de PM apresentada foi: forma específica de pensar que abrange o pensamento lógico, crítico, abstrato, algorítmico, algébrico, geométrico; que envolve a imaginação, a criatividade e a resolução de problemas, possui uma linguagem específica e extrapola a própria matemática, refutando assim a concepção de um pensamento fechado, exato e livre de incertezas.

Para levantar os dados perguntamos: *Você consegue estabelecer/ver alguma relação entre estes dois tipos de pensamento? Quais?* As respostas dos discentes foram compiladas em dez categorias que são apresentadas na Tabela 5.17.

Tabela 5.17 - Relações observadas pelos discentes entre PC e PM.

Classificação	**Categoria**	**Frequência**
Coerente	Ambos se baseiam na lógica	23%
	Ambos são utilizados para solucionar problemas	18%
	Ambos são criativos e imaginativos	4%
	Ambos envolvem a tomada de decisões	4%
	Muitos conceitos matemáticos podem ser expressos de forma computacional	4%
Incoerente	O Pensamento Computacional precisa do Pensamento Matemático	23%
	O Pensamento Matemático é criativo e o Pensamento Computacional é limitado	4%
	Ambos são exatos	4%
	Ambos são lineares	4%
-	Não observa nenhuma relação	12%

Fonte**:** Autores.

A categoria *ambos se baseiam na lógica* está alinhada com a fala de Barcelos e Silveira (2012), os autores apontam que a lógica é um elemento comum entre o PC e o conhecimento matemático.

Mestre *et al*. (2015) estabelece relações entre o PC e a metodologia de resolução de problemas. Mello (2015) discute o PM a partir da solução de problemas. Os apontamentos destes autores corroboram com a categoria *ambos são utilizados para solucionar problemas (18%).*

Consideramos a categoria *ambos são criativos e imaginativos* coerente. Essa relação vai ao encontro de Mannila e Settle (2014), os autores consideram a programação como um meio de estimular a criatividade, pois permite a criação de produtos. Além disso, Balanskat e Engelhart (2014) defendem que o PC estimula o pensamento criativo. O PM tem relação com a imaginação e criatividade, tendo em vista que de acordo com Monteleone, White e Geiger (2018) ele envolve a criação de novas ideias e de novas abordagens sobre problemas complexos.

O PM e PC tem relação com a resolução de problemas complexos, onde o sujeito precisa tomar série de decisões, por essa razão consideramos a categoria *ambos envolvem a tomada de decisões* coerente.

Barcelos e Silveira (2012) apontam que *softwares* podem ser utilizados para explorar e validar conceitos e modelos matemáticos, esses autores corroboram com a categoria *muitos conceitos matemáticos podem ser expressos de forma computacional*.

Entendemos que não existe uma dicotomia ou uma relação de dependência entre o PC e PM. Concordamos com Gadanidis *et al*. (2016) que sugerem a existência de uma conexão natural entre esses dois tipos de pensamento. Por essas razões consideramos as categorias *o Pensamento Computacional precisa do Pensamento Matemático* e, *o Pensamento Matemático é criativo e o Pensamento Computacional é limitado* como incoerentes.

As categorias *ambos são exatos* e *ambos são lineares* são semelhantes, as duas refletem uma visão procedimental sobre o PC e PM. Consideramos essa visão incoerente, pois esses pensamentos têm uma natureza criativa.

Constatamos que 53% das relações observadas pelos discentes foram coerentes, 35% incoerentes e 12% não conseguiram estabelecer nenhuma relação. Apesar dos acadêmicos não

terem conhecimentos prévios sobre PC ou conhecimentos formais sobre PM uma parcela significativa conseguiu perceber conexões consistentes entre esses tipos de pensamento.

5.2 ANÁLISE DOS ALGORITMOS

Nesta seção analisamos os algoritmos elaborados pelos discentes durante a sequência didática. Ao total houve quatro tipos de produção: quadrado em linguagem materna, triângulo escaleno em linguagem materna, quadrado em linguagem do *Scratch* e triângulo em linguagem do *Scratch*.

Os participantes foram organizados em 7 duplas e os discentes com maior domínio sobre programação realizaram as atividades individualmente. No entanto, no segundo dia da atividade uma parcela dos acadêmicos faltou e optamos por dissolver 3 duplas. Assim, outros 3 discentes realizaram a atividade de forma individual.

Desse modo, nessa seção analisamos os dados dos 14 sujeitos de pesquisa cujo perfil e concepções foram apresentados nas seções anteriores, no Quadro 5.4 apresentamos a organização dos discentes em relação ao seu perfil.

Quadro 5.4 - Relação entre perfil e organização dos participantes.

Identificação	**Sujeito**	**Interesse por programação?**	**Conhecimento prévio sobre programação?**
Individual 1	I1	Sim	Sim
Individual 2	I2	Sim	Sim
Individual 3	N1	Sim	Não
Individual 4	N2	Não opinou	Não
Individual 5	N3	Não opinou	Não
Individual 6	N4	Não informou	Não informou
Dupla 1 (D1)	I3 I4	Sim Sim	Não Não
Dupla 2 (D2)	I5 I6	Sim Não opinou	Sim Sim
Dupla 3 (D3)	I7 I8	Sim Sim	Não Não
Dupla 4 (D4)	I9 I10	Sim Não opinou	Sim Não

Fonte**:** Autores.

Em cada produção analisamos dois aspectos: algébricos e computacionais. Nos aspectos algébricos avaliamos as produções a partir dos seguintes focos: objetificação, simbolização e generalização (RADFORD, 2006). Sendo que na objetificação observamos quais aspectos dos objetos matemáticos foram evidenciados no algoritmo, na simbolização analisamos os meios que os discentes usaram para representar esses aspectos e na generalização avaliamos o grau de generalidade dos algoritmos.

Nos aspectos computacionais avaliamos os algoritmos em relação a sua estrutura e erros lógicos (ou a ausência destes). No critério de estrutura observamos a organização do algoritmo, enquanto a análise de erros visamos inferir se os sujeitos realizaram ou não práticas de depuração para aprimorar seus algoritmos. Ambos os critérios englobam os seguintes conceitos computacionais: sequência, ciclo, evento, condicional, operador e dados (BRENNAN; RESNICK, 2012). No Quadro 5.5 apresentamos uma síntese das características analisadas.

Quadro 5.5 - Aspectos Algébricos e Computacionais analisados nos algoritmos em linguagem materna.

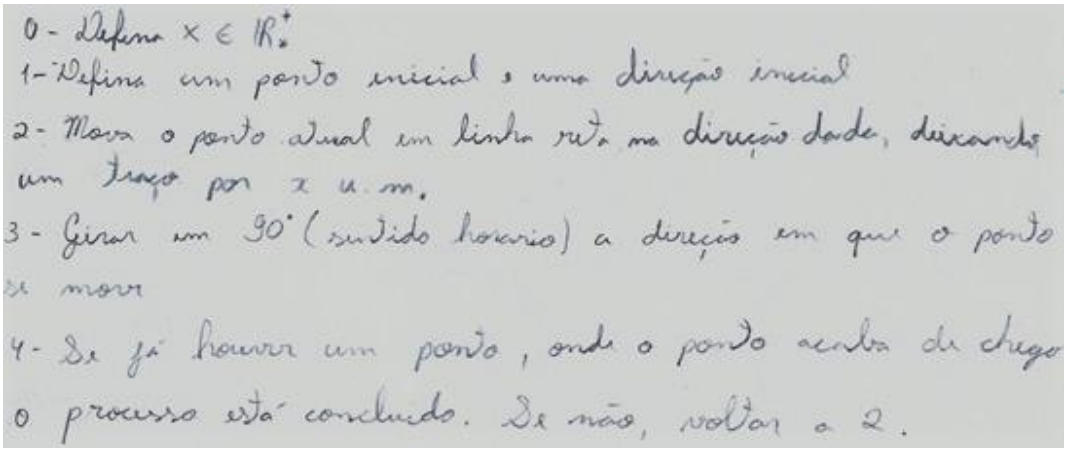
Aspecto	**Foco**	**Indicadores**
Algébrico	Objetificação	Características do objeto matemático que são evidenciadas pelo algoritmo.
	Simbolização	Recursos semióticos utilizados pelos discentes para representar as características do objeto.
	Generalização	Grau de generalização segundo Radford (2006).
Computacional	Estrutura	Forma com que os comandos são organizados e apresentados.
	Depuração	Erros lógicos observados no algoritmo.

Fonte**:** Autores.

Os aspectos, foco e indicadores foram estabelecidos com base em dois fatores: referencial teórico e capacidade de observação do pesquisador. Os aspectos algébricos foram estabelecidos com base em Radford (2006), os aspectos computacionais com base em Brennan e Resnick (2012). Os indicadores para esses aspectos surgiram de forma emergente e foram refinados durante o processo de análise conforme a percepção do pesquisador sobre os dados amadurecia.

Os algoritmos em linguagem materna foram produzidos pelos discentes de próprio punho em folhas A4, na figura 5.1 apresentamos um exemplo, posteriormente esses dados foram tabulados e analisados.

Figura 5.1 - Quadrado em linguagem materna produzido pelo Integral 2 (*I2*).



Fonte**:** Autores.

Para identificar os sujeitos que elaboraram cada algoritmo associamos as produções aos códigos de identificação apresentados no Quadro 5.5. No apêndice E apresentamos a relação de todos os algoritmos em linguagem materna produzidos pelos discentes.

Entendemos que o método utilizado para calcular as porcentagens de frequência adotado anteriormente não seria significativo para os dados dessa seção. Em vista, nessa seção as frequências foram calculadas como a razão entre o número de ocorrências da categoria e o total de algoritmos, ou seja $\frac{ocorrências\ da\ categoria}{total\ de\ algoritmos}$.

5.2.1 Quadrado em Linguagem Materna

Nessa subseção discutimos os algoritmos em linguagem materna do processo de construção de um quadrado, apresentados no Apêndice E, considerando os aspectos algébricos e computacionais conforme apresentado no Quadro 5.5. Na sequência apresentamos as análises sobre os aspectos algébricos, sintetizamos esses resultados na Tabela 5.18.

Tabela 5.18 - Síntese sobre aspectos algébricos – quadrado em linguagem materna.

(continua)

Aspecto	**Categorias**	**Frequência**
Objetificação	Vértices, ângulos internos e arestas.	60%
	Ângulos internos e arestas.	20%
	Vértices e ângulos internos.	20%

Tabela 5.18 - Síntese sobre aspectos algébricos – quadrado em linguagem materna.

(conclusão)

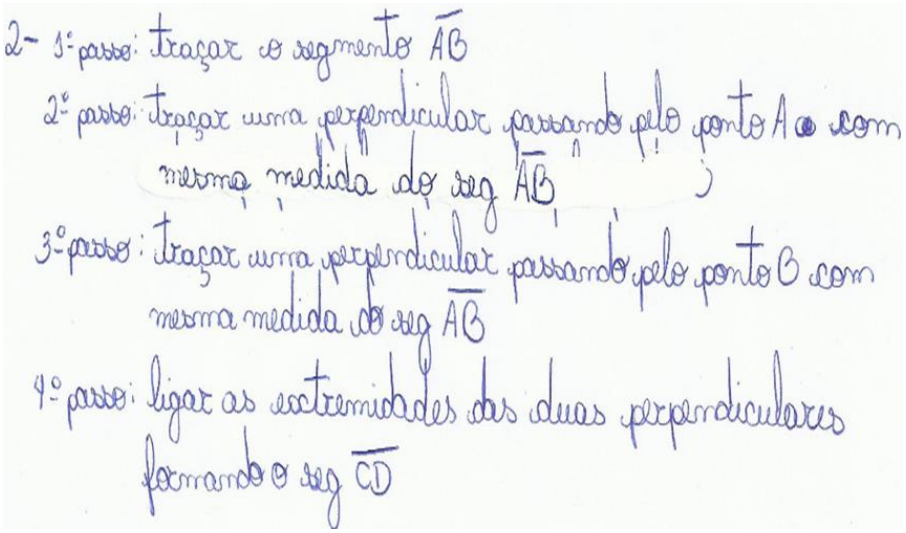
Aspecto	Categorias	Frequência
Simbolização	Simbolização indireta informal ou semiformal.	90%
	Simbolização direta semiformal.	10%
Generalização	Contextual.	90%
	Simbólica.	10%

Fonte**:** Autores.

Observamos três categorias sobre a *objetificação*. Na primeira os aspectos objetificados são: vértices, ângulos internos e arestas do quadrado. Observamos que 60% dos algoritmos estão associados a essa categoria (*I1*, *I2*, *D1*, *D2*, *D3* e *D4*). O algoritmo do sujeito *I2*, apresentado na figura 5.1, é um exemplo dessa categoria.

Na segunda os discentes objetificaram os ângulos internos e arestas do quadrado, ao todo 20% dos algoritmos compõem essa categoria (*N3* e *N4*). O algoritmo do sujeito *N3*, apresentado na figura 5.2, é um exemplo dessa categoria.

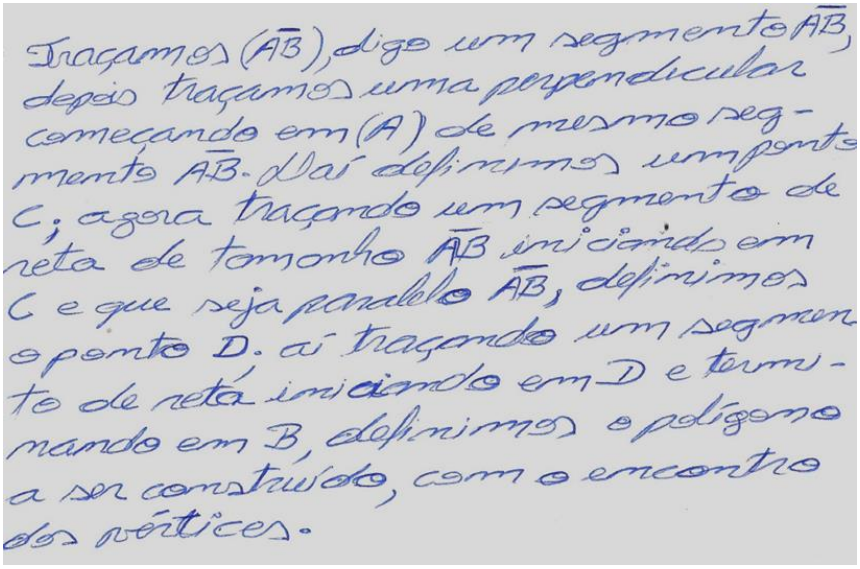
Figura 5.2 - Quadrado em linguagem materna produzido pelo Noturno 4 *(N4)*.



Fonte**:** Autores.

A terceira abrange as produções onde os vértices e ângulos internos do quadrado foram objetificados, 20% dos algoritmos estão nessa categoria (*N1* e *N2*). O algoritmo do sujeito *N1*, apresentado na figura 5.3, é um exemplo dessa categoria.

Figura 5.3 - Quadrado em linguagem materna produzido pelo Noturno 1 *(N1).*



Fonte**:** Autores.

Assim, a diferença das categorias de objetificação está na forma com que diferentes características são combinadas, pois de forma geral apenas três aspectos do quadrado foram evidenciados.

No âmbito da *simbolização* percebemos duas categorias. Na primeira os vértices foram representados como pontos, as arestas foram expressas pelos segmentos de reta formados ao conectar os vértices, e os ângulos internos foram representados pelos ângulos retos formados pelas interseções entre extremidades dos segmentos de reta.

Além disso, a maioria dos aspectos objetificados nesses algoritmos foram caracterizados de forma indireta e com meios semióticos informais ou semiformais. Ao todo 90% dos algoritmos foram relacionados a essa categoria (*I1*, *N1*, *N2*, *N3*, *N4*, *D1*, *D2*, *D3* e *D4*). As figuras 5.2 e 5.3 apresentam exemplos de algoritmos dessa categoria.

A segunda categoria engloba o algoritmo no qual a variável $x \in \mathbb{R}_+^*$ foi utilizada para simbolizar o comprimento das arestas do quadrado, o vértice inicial é representado por um ponto inicial e os ângulos internos são enunciados de forma explícita (gire 90º graus). O algoritmo do sujeito I2, apresentado na figura 5.1, foi relacionado a essa categoria (10%).

As simbolizações observadas se diferenciam pelo grau de formalidade e pela forma com que os objetos são apresentados (direta e indireta). A maioria dos discentes utilizou meios

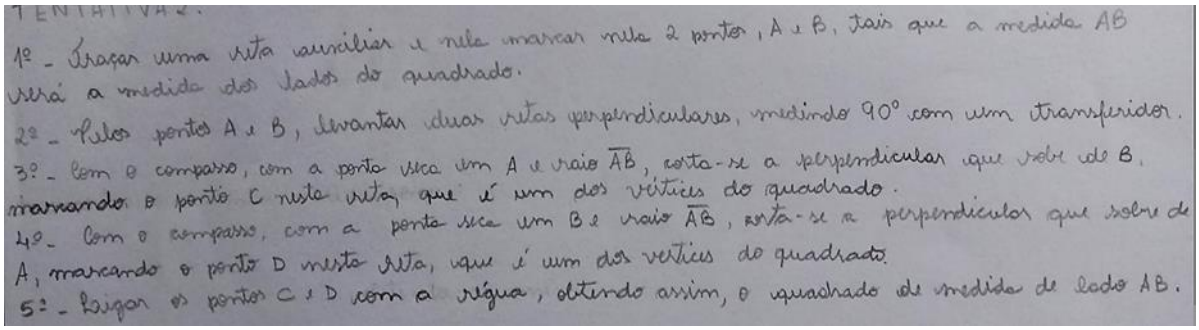
semióticos próprios, isto é, não se restringiram a nenhum sistema formal de simbolização para expressar os comandos do algoritmo, além disso esses sujeitos não enunciaram as variáveis ou constantes envolvidas de forma explicita. Apenas o sujeito *I2* explicitou todos os objetos, é interessante ressaltar que esse discente declarou ter interesse e conhecimento prévio sobre programação.

Em relação a *generalização* constatamos duas categorias. A primeira agrupa os sujeitos que estabeleceram generalizações contextuais, consistindo em 90% das produções (*I1*, *N1*, *N2*, *N3*, *N4*, *D1*, *D2*, *D3* e *D4*). As figuras 5.2 e 5.3 apresentam exemplos de algoritmos dessa categoria.

A generalização é entendida como contextual porque os algoritmos permitem a construção de quadrados com comprimentos variáveis, porém, a medida da aresta não é enunciada de forma explicita. Ademais, destacamos que nesses algoritmos a simbolização depende de elementos contextuais, isto é, da percepção do sujeito sobre a posição dos vértices, retas e semirretas.

Isso pode ser visualizado no primeiro passo do algoritmo produzido pela dupla *D1*, apresentado na Figura 5.4: “Traçar uma reta auxiliar e nela marcar 2 pontos, *A* e *B*, tais que a medida *AB* será a medida dos lados do quadrado”. Observamos que a medida da aresta é definida pela distância entre os pontos *A* e *B*, variando conforme esses pontos são reposicionados. Assim, o comprimento dos lados é definido a partir da visão do sujeito sobre essa reta auxiliar. Segundo Radford (2006) na generalização contextual a representação não é totalmente abstrata e está associada a percepção do sujeito sobre o objeto.

Figura 5.4 - Quadrado em linguagem materna produzido pela Dupla 1 *(D1)*.



Fonte**:** Autores.

Na segunda categoria a generalização estabelecida foi considerada como simbólica, visto que a aresta do quadrado não tem tamanho fixo, e a variável de comprimento é enunciada e definida de forma explícita e formal. O algoritmo do sujeito *I2*, apresentado na figura 5.1, foi relacionado a essa categoria (10%).

No primeiro passo o sujeito *I2* escreveu: "Defina $x \in R_+^*$", posteriormente essa variável é utilizada como comprimento das arestas do quadrado. De acordo com Radford (2006) na generalização simbólica a regularidade é expressa por meio de símbolos alfanuméricos.

Observamos também que os meios semióticos utilizados pelos discentes refletiram nas camadas de generalização estabelecidas. Na Tabela 5.19 apresentamos uma síntese das análises sobre os aspectos computacionais.

Tabela 5.19 - Síntese sobre aspectos computacionais – quadrado em linguagem materna.

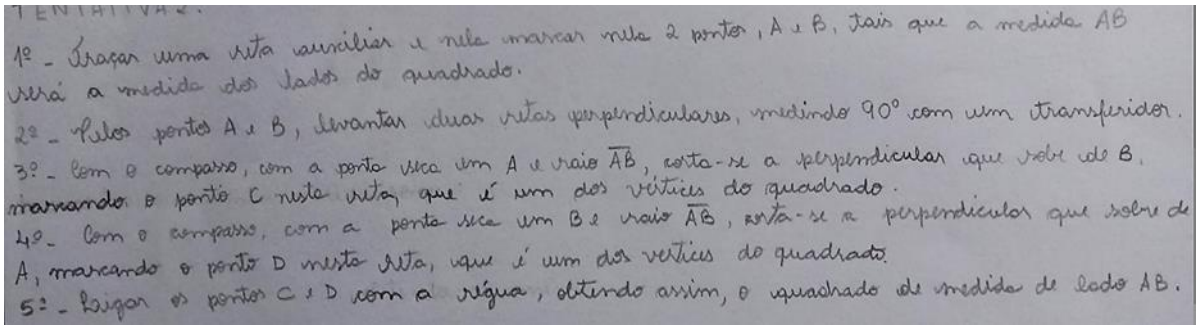
Aspecto	**Categorias**	**Frequência**
Estrutura	Passos são indicados e organizados de forma sequencial.	70%
	A sequência de passos é sugerida de forma implícita.	20%
	Há um único passo composto por múltiplas etapas distintas.	10%
	O algoritmo é estruturado como um ciclo iterativo.	10%
Depuração	Descrição de duas ou mais etapas em um mesmo passo.	60%
	Há uma imprecisão nos comandos que impossibilita a construção do quadrado.	20%
	Ausência de restrições sobre a posição dos pontos que originam o tamanho da aresta do quadrado.	20%
	Erros graves que comprometem a construção do quadrado.	10%

Fonte**:** Autores.

A primeira categoria reúne as produções onde os passos foram organizados em uma ordem lógica de execução, 70% das produções foram relacionadas a essa categoria (*I1*, *I2*, *N4*, *D1*, *D2*, *D3* e *D4*). Nesses algoritmos cada passo é indicado usando numerais, um exemplo é o algoritmo do sujeito *I2* apresentado na Figura 5.1.

A segunda categoria engloba os algoritmos onde a sequência dos passos é sugerida pela disposição do texto, mas, não é explicitada. 20% dos algoritmos se encaixam nessa categoria (*N2* e *N3*). O algoritmo do sujeito *N2*, apresentado na Figura 5.5, é um exemplo dessa categoria.

Figura 5.5 - Quadrado em linguagem materna produzido pelo Noturno 2 *(N2)*.



Fonte**:** Autores.

Na terceira reunimos os algoritmos onde não há separação nenhuma entre os passos, isto é, todas as ações são apresentadas num único parágrafo. Apenas o algoritmo do sujeito *N1*, apresentado na Figura 5.3, foi relacionado a essa categoria (10%).

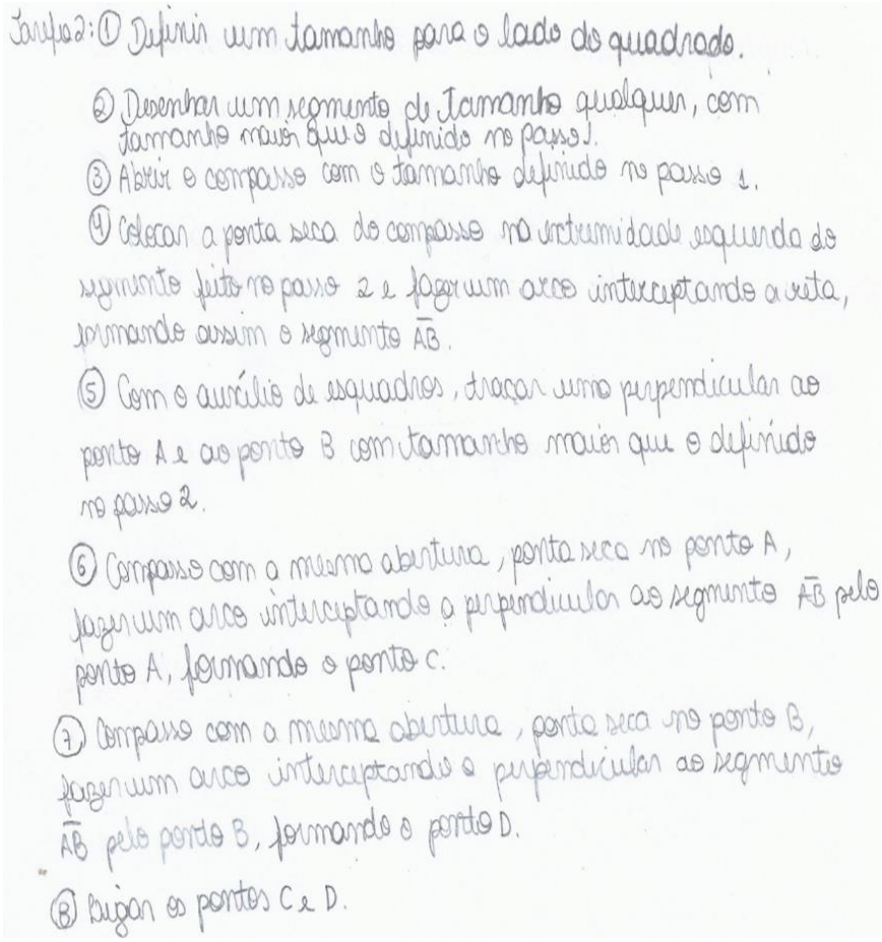
Na quarta agrupamos os algoritmos que estabeleceram um ciclo de passos que se repete até que o quadrado esteja construído, para exemplificar observe o passo 4 do algoritmo *I2*: “Se já houver um ponto, onde o ponto acaba de chegar o processo está concluído. Se não voltar a 2”. Apenas o algoritmo do sujeito *I2*, apresentado na Figura 5.1, foi associado a essa categoria representando 10% da amostra.

Quanto a *depuração* percebemos quatro tipos de problemas lógicos nos algoritmos, conforme apresentados na Tabela 5.19. No primeiro os sujeitos estabelecem múltiplas ações dentro de um mesmo passo, ou seja, descrevem duas ou mais etapas em um único passo. 60% dos algoritmos se encaixam nessa categoria (*I1*, *N1*, *N3*, *D1*, *D3* e *D4*). As figuras 5.3 e 5.4 apresentam exemplos de algoritmos dessa categoria.

Como exemplo dessa situação citamos o passo 1 da dupla *D1*: “Traçar uma reta auxiliar e nela marcar 2 pontos, *A* e *B*, tais que a medida *AB* será a medida dos lados do quadrado”. Avaliamos que “traçar uma reta auxiliar” é um comando e “marcar nela dois pontos ...” é uma outra ação, essas duas instruções deveriam ser decompostas em dois passos sequenciais.

O segundo problema engloba as produções que descrevem a construção do quadrado usando régua e compasso, essas produções representam 20% dos algoritmos (*I1* e *D3*). Em todas essas produções observamos uma brecha que impossibilita a construção do quadrado. A Figura 5.6 apresenta o algoritmo *I1* para exemplificar essa categoria.

Figura 5.6 - Quadrado em linguagem materna produzido pelo Indivíduo 1

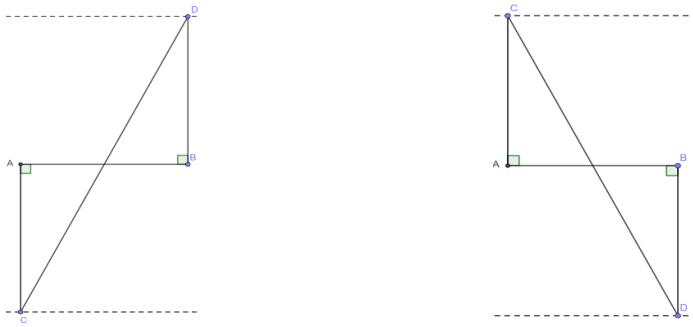


Fonte**:** Autores.

No algoritmo do sujeito *I1* é possível que ao menos dois segmentos sejam direcionados para sentidos diferentes gerando, ao final, dois triângulos retângulos ao invés de um quadrado. Observe os passos 6, 7 e 8 desse algoritmo: “6 - Compasso com a mesma abertura, ponta seca no ponto *A*, fazer um arco interceptando a perpendicular ao segmento $\overline{AB}$ pelo ponto *A*, formando o ponto *C*”; “7 - Compasso com a mesma abertura, ponta seca no ponto *B*, fazer um arco interceptando a perpendicular ao segmento $\overline{AB}$ pelo ponto *B*, formando o ponto *D*”; “8 – Ligar os pontos C e D”.

Para fechar o quadrado é necessário que os pontos *C* e *D* sejam colineares e pertençam a uma única reta paralela ao segmento $\overline{AB}$. Entendemos que, devido à falta de precisão nessas instruções, o ponto *C* pode ser feito acima do segmento $\overline{AB}$ e o ponto *D* abaixo desse segmento, ou vice-versa. Nesse caso os pontos *C* e *D* pertenceriam retas paralelas perpendiculares em relação ao segmento $\overline{AB}$ como apresentado na figura 5.7.

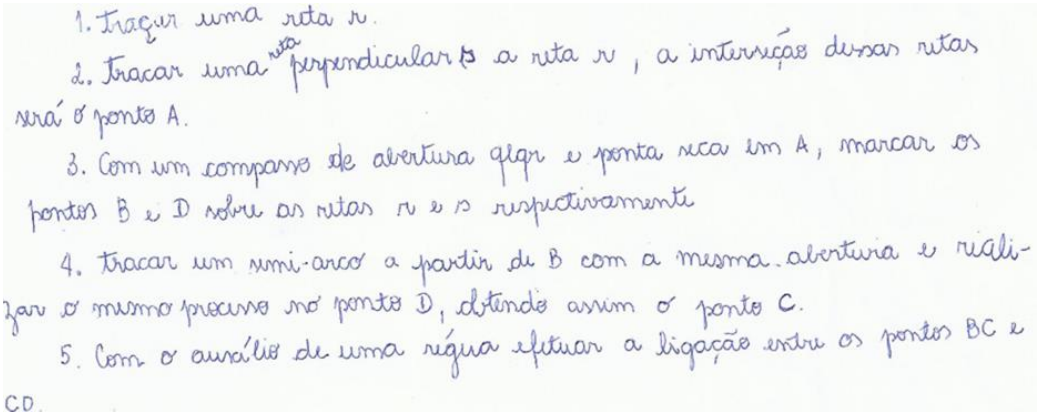
Figura 5.7 - Possível erro na execução do algoritmo *I1*.



Fonte**:** Autores.

No algoritmo da dupla *D3*, apresentado na Figura 5.8, é possível que os semiarcos não gerem uma intersecção. Observe os passos 4 e 5 desse algoritmo: "4. Traçar um semiarco a partir de *B* com a mesma abertura e realizar o mesmo processo no ponto *D*, obtendo assim o ponto *C*" e "5. Com o auxílio de uma régua efetuar a ligação entre os pontos *BC* e *CD*".

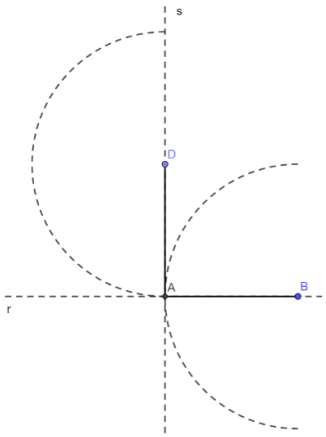
Figura 5.8 - Quadrado em linguagem materna produzido pela Dupla 3 *(D3)*.


1. Traçar uma reta r.
2. Tracar uma reta perpendicular a reta r, a intersição dessas retas será o ponto A.
3. Com um compasso de abertura qlqr e ponta seca em A, marcar os pontos B e D sobre as retas r e s respectivamente
4. Tracar um semi-arco a partir de B com a mesma abertura e realizar o mesmo procesu no ponto D, obtendo assim o ponto C.
5. Com o auxílio de uma régua efetuar a ligação entre os pontos BC e CD.

Fonte**:** Autores.

Seguindo o passo 4 é possível desenhar os semiarcos de modo que estes não se interceptem ou se encontrem apenas no ponto *A*, nesse caso o ponto *C* não é demarcado conforme mostra a figura 5.9.

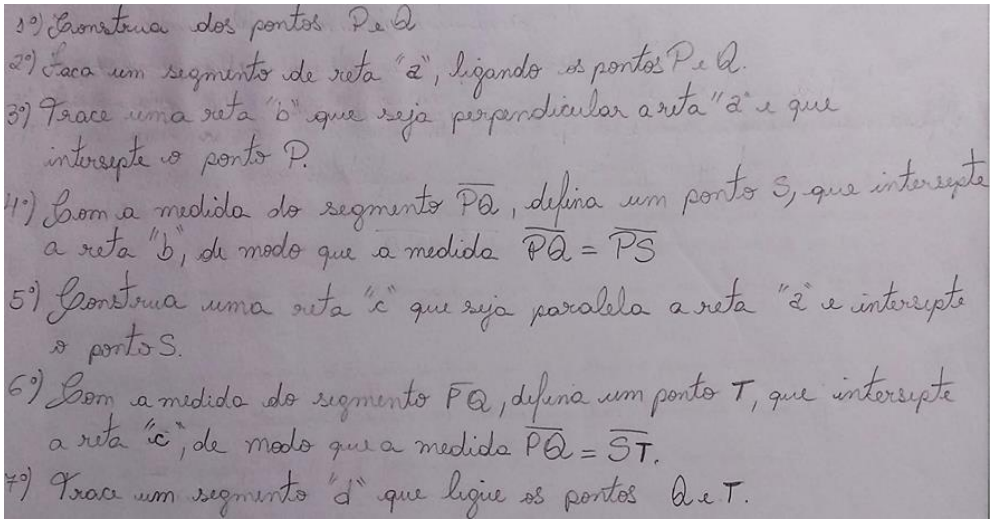
Figura 5.9 - Possível erro na execução do algoritmo *D3*.



Fonte**:** Autores.

O terceiro problema elencado quanto a depuração, Tabela 5.19, é a ausência de restrições sobre a posição dos pontos que originam o tamanho da aresta do quadrado. Em 20% dos algoritmos esses pontos podem ser coincidentes, ou seja, o comprimento da aresta pode ser nulo (*D1* e *D2*). As figuras 5.4 e 5.10 apresentam exemplos de algoritmos dessa categoria.

Figura 5.10 - Quadrado em linguagem materna produzido pela Dupla 2 *(D2)*.


1°) Construa dos pontos P e Q
2°) Faça um segmento de reta "a", ligando os pontos P e Q.
3°) Trace uma reta "b" que seja perpendicular a reta "a" e que intersepte o ponto P.
4°) Com a medida do segmento $\overline{PQ}$, defina um ponto S, que intersepte a reta "b", de modo que a medida $\overline{PQ} = \overline{PS}$
5°) Construa uma reta "c" que seja paralela a reta "a" e intersepte o ponto S.
6°) Com a medida do segmento $\overline{PQ}$, defina um ponto T, que intersepte a reta "c", de modo que a medida $\overline{PQ} = \overline{ST}$.
7°) Trace um segmento "d" que ligue os pontos Q e T.

Fonte**:** Autores.

Ilustramos essa situação com o passo 1 da dupla *D2*: “Construa dois pontos P e Q”[17], os pontos P e Q podem ser coincidentes tendo em vista que não é enunciada nenhuma restrição nesse sentido.

Vale destacar que apenas o algoritmo *I2* (10%) apresentou restrições explícitas sobre o comprimento do quadrado, esse algoritmo é apresentado na Figura 5.1. Como exemplo citamos o passo 0 do algoritmo produzido individualmente pelo sujeito *I2*: “Defina $x \in R_+^*$”.

A quarta categoria é composta pelo algoritmo da dupla *D1* (10%), apresentado na figura 5.4. Esse algoritmo não desenha o quadrado devido a um erro grave. Observe os passos 2 e 3: "2º - Pelos pontos *A* e *B*, levantar duas retas perpendiculares, medindo 90º com um transferidor"; "3º - Com o compasso, com a ponta seca em *A* e raio $\overline{AB}$, corta-se a perpendicular que sobe de *B*, marcando o ponto *C* nesta reta, que é um dos vértices do quadrado".

É impossível cruzar a perpendicular que passa pelo ponto *B* seguindo essas instruções, consequentemente o vértice *C* não é delimitado e o quadrado não pode ser concluído. O passo 4 comete um erro similar em relação a perpendicular de *A*, impossibilitando a demarcação do ponto *D*.

Assim, observamos que os discentes demonstram algumas dificuldades como: estabelecer sequências, usar a linguagem materna para expressar sua lógica de pensamento, e perceber que os passos do algoritmo devem ser formulados com precisão para evitar ambiguidades. Segundo Selby (2014) estas dificuldades são comuns para principiantes em programação e esse perfil corresponde à maioria dos discentes.

5.2.2 Triângulo Escaleno Em Linguagem Materna

Nessa seção analisamos os algoritmos em linguagem materna que descrevem a construção de triângulos escalenos em relação aos aspectos apresentados no Quadro 5.5. Assim como na subseção 5.2.1, inicialmente apresentamos a análise sobre os aspectos algébricos. Na sequência apresentamos as análises sobre os aspectos algébricos, sintetizamos esses resultados na Tabela 5.20. Já os algoritmos produzidos pelos sujeitos se encontram no Apêndice E.

[17] Vale pontuar que não se constrói pontos, eles podem ser marcados, determinados ou definidos por coordenadas. Logo, é equivocado dizer “Construa dois pontos P e Q”, seria mais apropriado dizer “Defina dois pontos P e Q”.

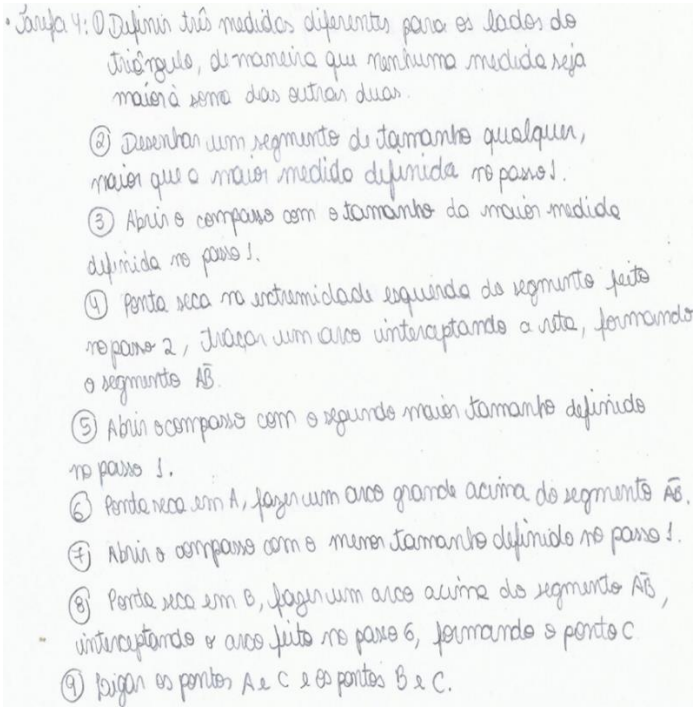
Tabela 5.20 - Síntese sobre aspectos algébricos – triângulo em linguagem materna.

Aspecto	Categorias	Frequência
Objetificação	Vértices e arestas.	80%
	Ângulo suplementar a um dos ângulos internos.	10%
	Ao menos uma das condições de existência de um triângulo escaleno.	10%
	Nenhum aspecto foi objetificado.	20%
Simbolização	Simbolização indireta informal ou semiformal.	70%
	Simbolização direta semiformal.	10%
	Não simbolizou nenhum objeto.	20%
Generalização	Contextual.	90%
	Simbólica.	10%
	Não estabeleceu generalizações.	20%

Fonte**:** Autores.

Percebemos quatro categorias no tocante a *objetificação*. Na primeira os aspectos evidenciados são: vértices e arestas do triângulo. Observamos que 80% dos algoritmos estão associados a essa categoria (*I1*, *I2*, *N1*, *N2*, *N3*, *D1*, *D2* e *D3*). Na figura 5.11 apresentamos o algoritmo *I1* para ilustrar essa categoria.

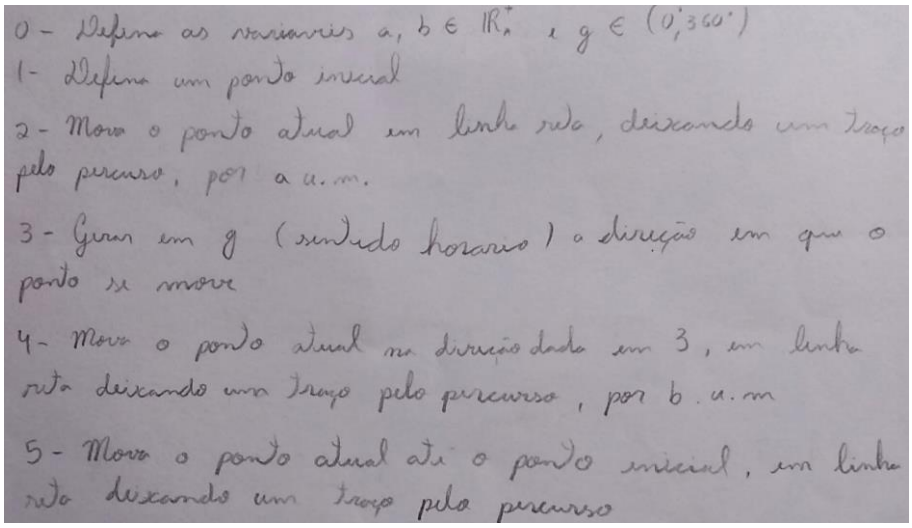
Figura 5.11 - Triângulo escaleno em linguagem materna produzido pelo *I1*.



Fonte**:** Autores.

Na segunda um ângulo suplementar a um dos ângulos internos do triângulo foi objetificado, apenas o algoritmo do sujeito *I2* compõe essa categoria (10%). Na figura 5.12 apresentamos o algoritmo *I2* para ilustrar essa categoria.

Figura 5.12 - Triângulo escaleno em linguagem materna produzido pelo *I2*.



Fonte**:** Autores.

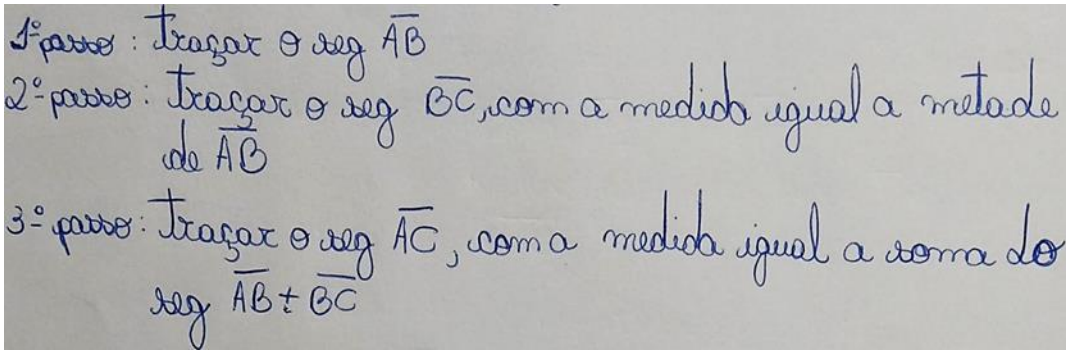
A terceira representa o algoritmo que objetifica ao menos uma das condições de existência de um triângulo, é composta pelo algoritmo do sujeito *I1* (10%), esse algoritmo é apresentado na Figura 5.11. Restringir o tamanho dos segmentos do triângulo escaleno é fundamental por duas razões: garantir que todos os lados terão comprimentos diferentes e que os lados satisfaçam as condições de existência de um triângulo.

Considere um triângulo cujos lados medem $a, b, c \in \mathbb{R}_+^*$. Para que ele exista é necessário que a medida de qualquer um dos lados seja menor que a soma dos outros dois, por exemplo: $a < b + c$. Além disso, é preciso que a medida de qualquer um dos lados seja maior que o valor absoluto da diferença entre os outros dois, por exemplo: $a > |b - c|$. Para que os algoritmos sempre gerem triângulos é necessário que essas duas condições sejam atendidas.

Exemplificamos essa categoria com o passo 1 do algoritmo elaborado pelo sujeito *I1*: "Definir três medidas diferentes para os lados do triângulo, de maneira que nenhuma medida seja maior que a soma das outras duas". Vale destacar que esse algoritmo impõe apenas uma das condições, ou seja, ainda é possível selecionar comprimentos que não geram um triângulo.

A quarta é composta pelos algoritmos que não objetificam nenhuma característica do triângulo escaleno, essas produções apresentam graves erros que impossibilitam a construção de um triângulo. Essa categoria é composta por 20% dos algoritmos (*N4* e *D4*). Na figura 5.13 apresentamos o algoritmo *N4* para ilustrar essa categoria.

Figura 5.13 - Triângulo escaleno em linguagem materna produzido pelo *N4*.



Fonte**:** Autores.

Analisando a Tabela 5.20 percebemos que diferentemente do que ocorreu no quadrado em linguagem materna, neste caso as categorias de objetificação se distinguem em relação aos aspectos objetificados.

Quanto ao aspecto da *simbolização,* observamos duas categorias. Na primeira os vértices foram simbolizados por meio de pontos e as arestas pelos segmentos de reta formados pelos vértices. A maioria dos aspectos objetificados nesses algoritmos foram caracterizados de forma indireta e com meios semióticos informais ou semiformais. Ao todo 70% dos algoritmos foram relacionados a essa categoria (*I1*, *N1*, *N2*, *N3*, *D1*, *D2* e *D3*). A Figura 5.11 apresenta um algoritmo como exemplo dessa categoria.

A segunda é composta pelo algoritmo do sujeito *I2* (10%), esse algoritmo simboliza o comprimento das arestas por meio das variáveis $a, b \in \mathbb{R}_+^*$ e o ângulo suplementar é representado pela variável $g \in (0°, 360°)$. A Figura 5.12 apresenta um algoritmo como exemplo dessa categoria.

Não avaliamos as produções *N4* e *D4* quanto a simbolização porque seus algoritmos não objetificaram nenhum aspecto do triângulo, ou seja, não há nenhum símbolo representando uma característica do triângulo.

Em relação a *generalização* constatamos três categorias. A primeira agrupa os sujeitos que estabeleceram generalizações contextuais, consistindo em 70% das produções (*I1*, *N1*, *N2*, *N3*, *D1*, *D2* e *D3*). Esses algoritmos permitem a construção de triângulos escalenos com comprimentos variáveis, contudo, a medida das arestas não é enunciada de forma explicita, e depende de elementos contextuais como a posição de um ponto acima de uma reta.

Para ilustrar essa situação considere os seguintes passos do algoritmo produzido pelo sujeito *I1*, apresentado na figura 5.11: "4 – Ponta seca na extremidade esquerda do segmento feito no passo 2, traçar um arco interceptando a reta, formando o segmento $\overline{AB}$"; "5 – Abrir o compasso com o segundo maior tamanho definido no passo 1"; "6 – Ponta seca em A, fazer um arco grande acima do segmento $\overline{AB}$".

No passo 5 há dois elementos contextuais "arco grande" e "acima do segmento", essas duas orientações refletem diretamente a percepção do sujeito sobre o objeto. Além disso, as medidas de duas arestas do triângulo também dependem do contexto, como pode ser observado no passo 8 desse algoritmo: "8 – Ponta seca em B, fazer um arco acima do segmento AB, interceptando o arco feito no passo 6, formando o ponto C".

Além disso, destacamos que as características do triângulo escaleno não foram percebidas em sua totalidade, visto que os algoritmos dessa categoria não impõem todas as condições de existência de um triângulo escaleno.

A segunda categoria é composta pelo algoritmo do sujeito *I2*, apresentado na figura 5.12, representando 10% do total. A generalização estabelecida foi considerada como simbólica, pois as arestas do triângulo têm comprimento variável e duas delas são enunciadas por meio de símbolos formais.

Vale ressaltar que definir as três medidas pode comprometer a construção do triângulo escaleno, uma vez que os comprimentos escolhidos poderiam violar as condições de existência. Desse modo, fixar o comprimento de duas arestas e gerar a terceira pela união do vértice inicial com o vértice final é a melhor alternativa para garantir que a figura resultante sempre será um triângulo escaleno.

Na terceira categoria agrupamos as produções onde nenhum tipo de generalização foi percebido, 20% dos algoritmos compõem essa categoria (*N4* e *D4*). A Figura 5.13 apresenta o algoritmo *N4* como exemplo dessa categoria.

Percebemos que as categorias e frequências observadas em relação a simbolização e contextualização são similares às estabelecidas no algoritmo do quadrado. Apenas o sujeito *I2* simbolizou de forma direta e semiformal os aspectos das figuras geométricas e estabeleceu generalizações simbólica nos dois casos. A diferença entre os resultados está nos 20% dos algoritmos que não geram o triângulo.

No tocante aos aspectos computacionais as discussões estão relacionadas à estrutura e erros lógicos (ou a ausência destes), conforme Quadro 5.5. As análises sobre esses aspectos são sintetizadas na Tabela 5.21.

Tabela 5.21 - Síntese sobre aspectos computacionais – triângulo em linguagem materna.

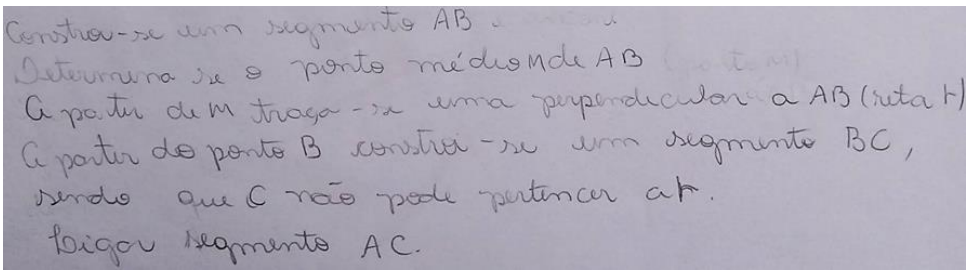
Aspecto	Categorias	Frequência
Estrutura	Passos são indicados e organizados de forma sequencial.	60%
	A sequência de passos é sugerida de forma implícita.	20%
	Há um único passo composto por múltiplas etapas distintas.	20%
Depuração	Descrição de duas ou mais etapas em um mesmo passo.	70%
	O sentido do giro do compasso não é especificado.	20%
	Ausência de restrições sobre os dados de entrada.	70%
	Erros graves que comprometem a construção do triângulo.	20%

Fonte: Autores.

No tocante à *estrutura* dos algoritmos observamos três categorias. A primeira reúne as produções onde os passos foram organizados em uma ordem lógica de execução, 60% dos algoritmos foram relacionados a essa categoria (*I1*, *I2*, *N4*, *D1*, *D2* e *D4*). Nesses algoritmos cada passo é indicado usando numerais. As figuras 5.11, 5.12 e 5.13 apresentam exemplos dessa categoria.

Na segunda reunimos os algoritmos onde a sequência dos passos é sugerida pela disposição do texto, mas, não é explicitada. 20% dos algoritmos se encaixam nessa categoria (*N3* e *D3*). A Figura 5.14 apresenta o algoritmo *N3* como exemplo dessa categoria.

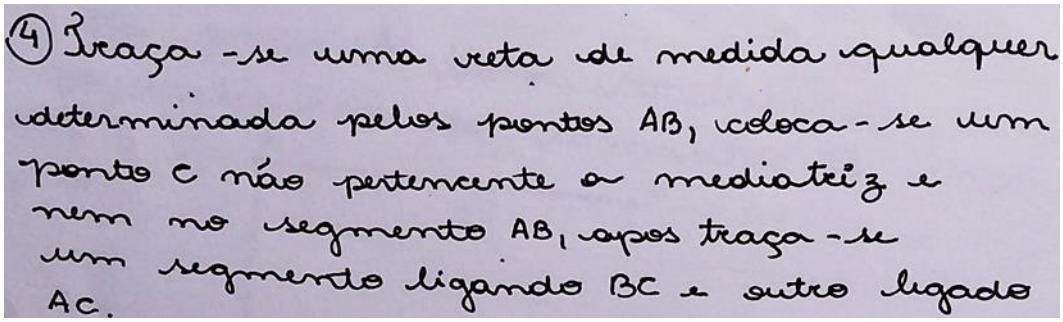
Figura 5.14 - Triângulo escaleno em linguagem materna produzido pelo *N3*.



Fonte**:** Autores.

A terceira categoria engloba os algoritmos onde não há nenhuma separação entre os passos, isto é, todas as ações são apresentadas num único parágrafo. 20% dos algoritmos foram relacionados a essa categoria (*N1* e *N2*). A Figura 5.15 apresenta o algoritmo *N2* como exemplo dessa categoria.

Figura 5.15 - Triângulo escaleno em linguagem materna produzido pelo *N2*.

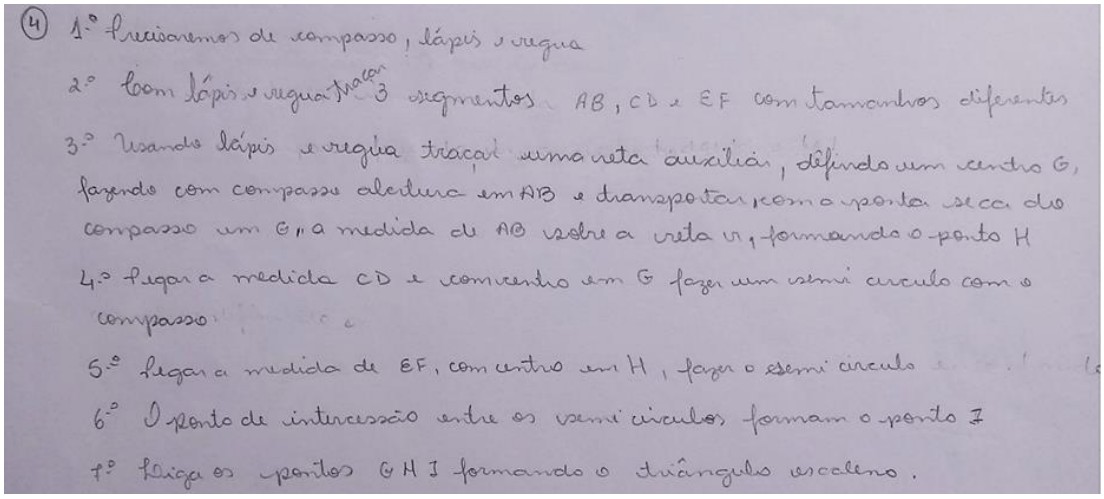


Fonte**:** Autores.

Quanto a *depuração* percebemos quatro tipos problemas lógicos nos algoritmos. No primeiro os sujeitos estabelecem duas ações distintas dentro de um mesmo passo, ou seja, descrevem duas ou mais etapas em um único passo. 70% dos algoritmos se encaixam nessa categoria (*I1*, *N1*, *N2*, *N3*, *D1*, *D2* e *D3*). As figuras 5.11, 5.14 e 5.15 apresentam exemplos dessa categoria.

Observamos um problema específico às produções que descrevem a construção do triângulo usando régua e compasso, sendo este o segundo problema. O sentido para o qual o compasso deve ser girado não é especificado, podendo acarretar que o terceiro vértice do triângulo não seja gerado, 20% dos algoritmos se encaixam nessa categoria de erro (*D2* e *D3*). Para ilustrar essa situação observe o algoritmo da dupla *D2* apresentado na figura 5.16.

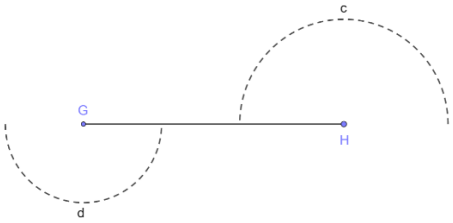
Figura 5.16 - Algoritmo de construção de triângulo escaleno da dupla *D2*.



Fonte**:** Autores.

Na figura 5.17 ilustramos um caso em que o algoritmo da dupla D2 pode dar errado, o primeiro semicírculo (denominado c) pode ser feito acima do segmento GH, enquanto o segundo (denominado d) pode ser construído abaixo desse segmento.

Figura 5.17 - Exemplo de caso em que não há interseção entre os semicírculos.



Fonte**:** Autores.

O terceiro problema é a ausência de restrições sobre os comprimentos dos segmentos que originam os vértices do triângulo. Em 70% dos algoritmos esses comprimentos podem não satisfazer as condições de existência do triângulo (*I1*, *N1*, *N2*, *N3*, *D1*, *D2* e *D3*). As figuras 5.11, 5.14, 5.15 e 5.16 apresentam exemplos dessa categoria.

Ademais, nos algoritmos que geram os lados do triângulo a partir de vértices, os pontos iniciais podem ser coincidentes impossibilitando a construção do triângulo. O passo 1 do algoritmo *I2* exemplifica essa situação: "0 – Defina as variáveis $a, b \in \mathbb{R}_+^*$". Note que nessas condições $a = b$, uma vez que não há restrições nesse sentido.

Observamos que alguns algoritmos não geram nenhum tipo de triângulo, sendo este o quarto tipo de erro, 20% das produções apresentam esse problema (*N4* e *D4*). É importante destacar que esta categoria se diferencia da anterior porque os algoritmos produzidos apresentam erros graves que comprometem a construção do triângulo.

O algoritmo produzido pelo sujeito *N4*, apresentado na figura 5.13, prevê deliberadamente que as condições de existência de um triângulo sejam violadas. Para que um triângulo exista é necessário que a medida de qualquer um dos lados seja menor que a soma dos outros dois. No entanto o passo 3 do algoritmo produzido pelo sujeito *N4* viola essa condição ao orientar: "3º passo: traçar o segmento $\overline{AC}$, com a medida igual à soma dos segmentos $\overline{AB} + \overline{BC}$".

No algoritmo produzido pela dupla *D4* observamos um erro conceitual que impossibilita a construção do triângulo. Os passos 1 e 2 dessa produção ilustram essa situação: "1 - Construir uma reta com uma medida qualquer" e "2 - A partir da extremidade desta reta, fazer uma nova, também com medida qualquer e uma abertura angular entre elas.

Uma reta é formada por um conjunto infinito de pontos alinhados, sendo geometricamente representada por uma linha reta. Desta forma, os dois passos citados apresentam erros conceituais, como do passo 1 ao atribuir ideia de dimensão à reta e do passo 2 ao citar que uma reta tem extremidade; além da falta de formalismo matemático, pois não é possível construir ou fazer uma reta, mas sim traçar uma reta.

Percebemos um aumento na frequência da categoria "descrição de duas ou mais etapas em um mesmo passo" em relação aos dados obtidos sobre o quadrado. Essa categoria teve 60% de frequência no algoritmo do quadrado e 70% no algoritmo do triângulo.

Corrêa *et al.* (2018) relatam três dificuldades, apontadas por licenciandos em matemática, sobre a produção de um algoritmo que desenhe um triângulo escaleno em linguagem materna, são elas: determinar o sentido ou medida dos ângulos; elaborar, ordenar ou descrever o passo a passo e obedecer às propriedades matemáticas das formas geométricas.

Neste estudo não observamos nenhum dado que nos permita inferir se os discentes tiveram ou não dificuldade em determinar o sentido ou medida dos ângulos. Contudo, verificamos que os discentes tiveram dificuldade em expressar esses aspectos, posto que tanto no quadrado quanto no triângulo o sentido de giro do compasso não foi especificado.

Os dados apresentados nas Tabelas 5.20 e 5.21 mostram que os sujeitos desta pesquisa também apresentaram dificuldade na elaboração, ordenação e descrição dos passos do algoritmo, tendo em vista que uma significativa parte das categorias encontradas no critério de depuração se refere a ordenação e descrição.

Outras dificuldades observadas, referem-se ao formalismo dos conceitos matemáticos e a relevância das condições de existência do ente geométrico, visto que uma significativa parcela dos participantes ignorou as propriedades do triângulo escaleno.

Segundo Brandt *et al.* (2018) um objeto matemático pode ser representado de diversas maneiras. Entendemos que os algoritmos em linguagem materna ilustram essa situação, uma vez que os algoritmos se referem ao mesmo objeto, porém, as formas de representação são diversificadas.

Por fim, destacamos que nos algoritmos em linguagem materna a maioria dos sujeitos utilizaram comandos e meios de representação próprios, ou seja, não recorreram nenhum sistema formal de programação ou simbolização para formular seus comandos. Radford (2006) relata que num primeiro momento a simbolização acontece de forma idiossincrática, entendemos que esses resultados corroboram com essa visão.

Além disso, observamos que apenas um dos sujeitos que possui conhecimento prévio baseou seu algoritmo em recursos semióticos formais, e buscou estruturá-lo de modo semelhante à organização sintática de uma linguagem formal de programação.

5.2.3 Quadrados Produzidos no *Scratch*

Nesta subseção discutiremos os algoritmos dos quadrados feitos no *Scratch*, todas as produções são apresentadas no apêndice F. Os algoritmos foram analisados em relação aos aspectos algébricos e computacionais, conforme apresentado no Quadro 5.5.

As análises dessa seção foram organizadas de forma diferente das anteriores. Primeiramente apresentamos nossas considerações sobre a *estrutura* dos algoritmos, na sequência a *objetificação, simbolização, generalização* e por último a *depuração*.

O critério de estrutura foi empregado de forma diferente da utilizada na análise dos algoritmos em linguagem materna. Enquanto no primeiro observamos como os discentes organizaram os comandos, neste reparamos na forma com que o quadrado foi construído. As análises sobre a estrutura e objetificação foram sintetizadas na Tabela 5.22.

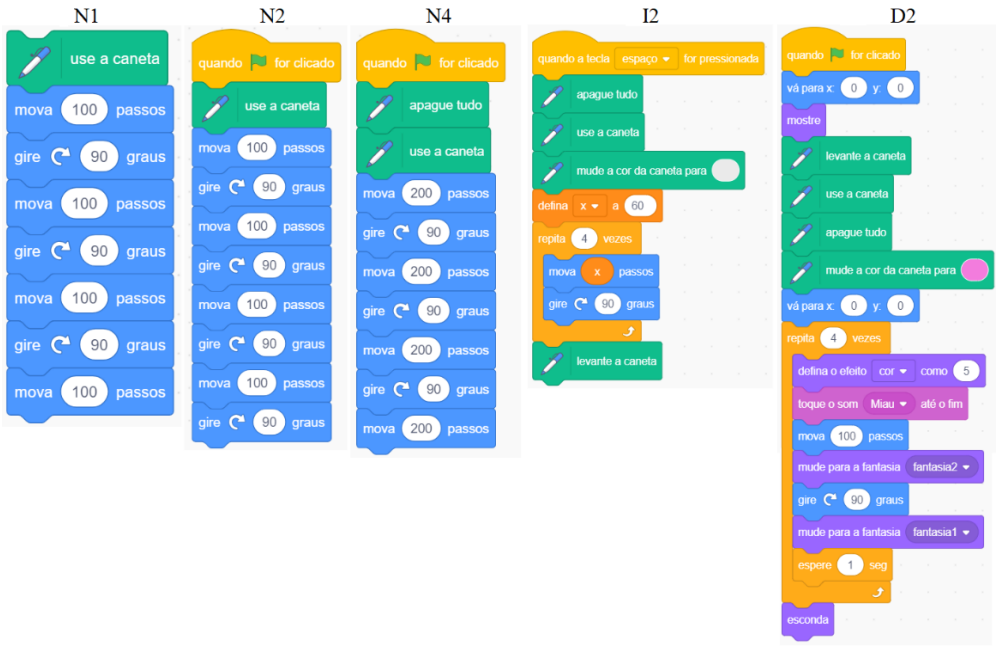
Tabela 5.22 - Categorias de estrutura e objetificação – quadrado no Scratch.

Aspecto	Categorias	Frequência
Estrutura	Desenho do quadrado baseado em segmentos de reta e ângulos internos.	50%
	Desenho do quadrado baseado nas coordenadas cartesianas.	30%
	Algoritmos que não desenham um quadrado.	20%
Objetificação	Ângulos internos do quadrado.	50%
	Arestas do quadrado.	60%

Fonte**:** Autores.

No tocante a *estrutura* observamos três categorias. A primeira reúne algoritmos onde o quadrado foi desenhado usando ângulos internos e segmentos de reta. Essa categoria abrange 50% das produções (*I2*, *N1*, *N2*, *N3* e *D2*), eles são apresentados na Figura 5.18. Os programas foram dispostos em ordem de complexidade, sendo o *N1* o algoritmo mais simples dessa categoria e o programa *D2* o mais complexo.

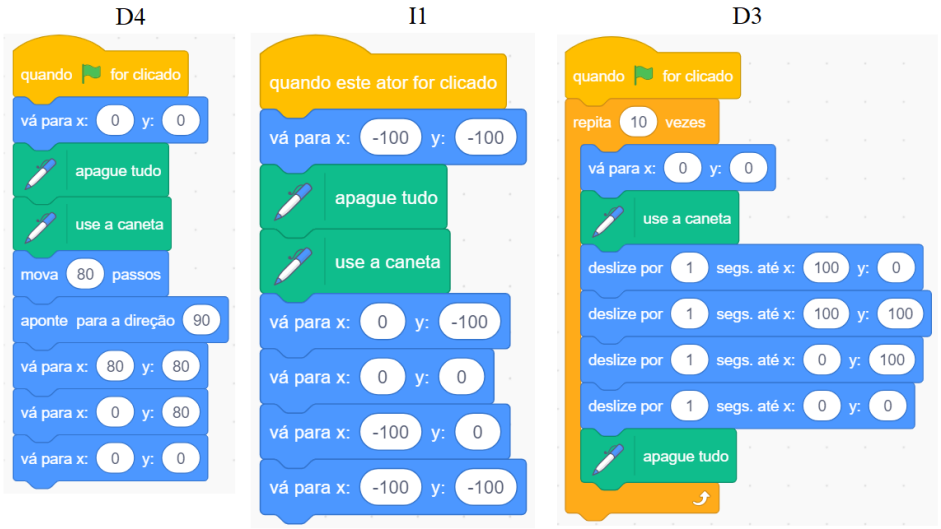
Figura 5.18 - Construção de quadrado usando ângulos internos e segmentos de reta.



Fonte: Autores.

Na segunda categoria agrupamos os algoritmos que utilizaram coordenadas cartesianas, apresentados na Figura 5.19, 30% das produções foram relacionadas a essa categoria (*I1*, *D3* e *D4*).

Figura 5.19 - Construção do quadrado usando coordenadas cartesianas.

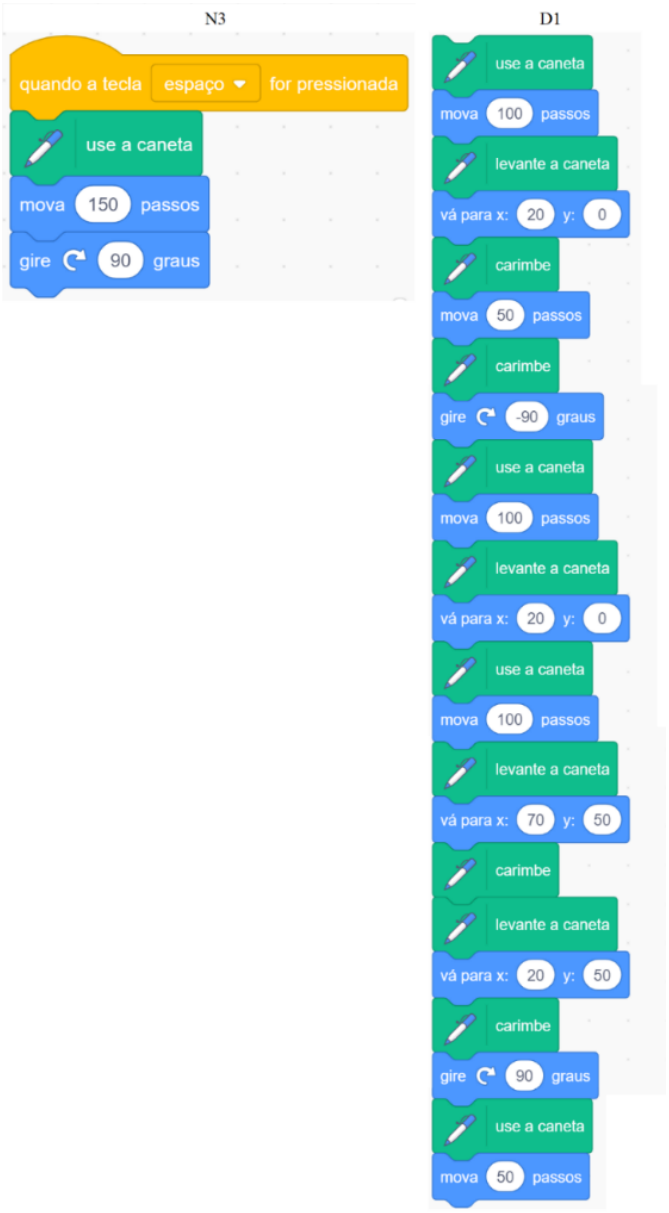


Fonte**:** Autores.

Os programas *D4* e *I1* da Figura 5.19 são similares. O programa *D3* desenha o quadrado simulando o movimento de uma caneta, isto é, o usuário vê as arestas deslizando como se estivessem sendo desenhadas em uma folha.

A terceira engloba os algoritmos que não desenham um quadrado, 20% das produções se encaixam nessa categoria (*N3* e *D1*). Na Figura 5.20 apresentamos esses algoritmos.

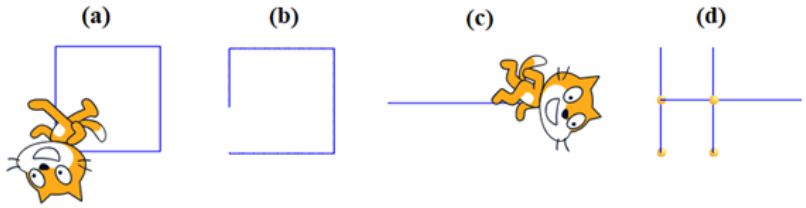
Figura 5.20 - Programas que não desenham um quadrado.



Fonte**:** Autores.

Na Figura 5.21 são apresentados os desenhos gerados pelos programas, sendo que: (a) corresponde a todos os programas da primeira categoria e os algoritmos *I1* e *D4* da segunda; (b) ao programa *D3* da segunda categoria; (c) e (d) aos programas *N3* e *D1* respectivamente, ambos da terceira categoria.

Figura 5.21 - Palco dos algoritmos do quadrado



Fonte**:** Autores.

Os programas relacionados à parte (a) da Figura 5.21 desenham o quadrado de forma direta, as arestas simplesmente aparecem na tela sem nenhum tipo de animação. Na parte (b) o programa desenha o quadrado simulando o movimento de uma caneta, ou seja, o usuário vê o segmento se movimentar entre dois vértices. O programa da parte (c) desenha apenas uma aresta e o programa da parte (d) produz os segmentos de reta e pontos, ambos de forma direta.

Em relação a *objetificação* observamos duas categorias. Na primeira os aspectos objetificados são: ângulos internos e arestas do quadrado. Observamos que 60% dos algoritmos estão associados a essa categoria (*I2*, *N1*, *N2*, *N3*, *N4* e *D2*). Na segunda os discentes objetificaram os vértices do quadrado, ao todo 50% dos algoritmos compõem essa categoria (*I1*, *D1*, *D2*, *D3* e *D4*).

Destacamos que os aspectos objetificados nesses algoritmos são os mesmos que os apontados nas produções em linguagem materna. Supomos que isso ocorreu porque os discentes basearam seus algoritmos do *Scratch* nos elaborados em linguagem materna.

Constatamos sete categorias em relação a *simbolização*, para facilitar a discussão organizamos os dados no Quadro 5.6.

Quadro 5.6 - Categorias de simbolização – quadrado no *Scratch*.

(continua)

Aspecto objetificado	**Simbolização**	**Frequência/Sujeitos**
Arestas do quadrado	Combinação dos comandos “mova __ passos” e “use a caneta”.	80% (*I1*, *I2*, *N1*, *N2*, *N3*, *N4*, *D2* e *D4*).
	Combinação dos comandos “vá para x:__ e y:__” e “use a caneta”.	60% (N1, N2, N3, N4, I2 e D2).
	Combinação dos comandos “deslize por __ segs. até x:__ e y:__” e “use a caneta”.	10% (*D3*).

Quadro 5.6 - Categorias de simbolização – quadrado no Scratch.

(conclusão)

Aspecto objetificado	**Simbolização**	**Frequência/Sujeitos**
Vértices do quadrado	Comando “vá para x:__ e y:__”.	40% (*I1*, *D2*, *D3* e *D4*).
	Combinação dos comandos “vá para x:__ e y:__” e “carimbe”.	10% (D1).
Ângulos internos	Comando “gire __ graus”.	60% (*I2*, *N1*, *N2*, *N3*, *N4* e *D2*).
Regularidades na estrutura do código	Comando “repita __ vezes”.	30% (*I2*, *D2* e *D3*).

Fonte**:** Autores.

Diferentemente dos algoritmos em linguagem materna, neste caso os discentes foram forçados a utilizar um sistema simbólico formal para representar os aspectos objetificados. Como todos utilizaram o mesmo conjunto de comandos a classificação dos diferentes tipos de simbolização foi facilitada, assim conseguimos estabelecer mais subcategorias do que nos algoritmos de linguagem materna.

Em relação a *generalização* estabelecemos três categorias que são apresentadas na Tabela 5.23.

Tabela 5.23 - Categorias de generalização – quadrado no *Scratch*.

Aspecto	**Categorias**	**Frequência**
Generalização	Factual.	80%
	Indução ingênua.	10%
	Não estabeleceu generalizações.	10%

Fonte**:** Autores.

A primeira reúne os sujeitos que estabeleceram generalizações factuais, consistindo em 80% das produções (*I1*, *I2*, *N1*, *N2*, *N4*, *D2*, *D3* e *D4*). Consideramos as generalizações como factuais (RADFORD, 2006) porque os programas podem gerar um conjunto limitado de quadrados, contanto que o valor nos comandos “mova __ passos” seja alterado manualmente. Os discentes aparentam reconhecer o comprimento como uma variável, porém, essa variável não é nomeada e permanece implícita.

O programa do sujeito *N3* produz apenas uma aresta do quadrado. Durante a realização da atividade, este acadêmico acidentalmente percebeu que podia executar seu algoritmo quatro vezes para desenhar o quadrado. Assim, o código não representa um quadrado, mas sim uma

aresta. Nesse caso a regularidade foi estabelecida com base na tentativa e erro, em vista disso classificamos essa abstração como indução ingênua (RADFORD, 2006), essa categoria é composta apenas pelo programa *N3* (10%).

O programa da dupla *D1* desenha os vértices e duas arestas do quadrado, além disso também desenha alguns segmentos externos ao quadrado pretendido. Apesar de o algoritmo demarcar os vértices, ele não desenha um quadrado como requisitado. Em vista disso, entendemos que a generalização não foi estabelecida, somente o algoritmo *D1* foi associado a essa categoria (10%).

Em relação à *depuração* observamos cinco categorias de problemas lógico, esses dados são apresentados na Tabela 5.24.

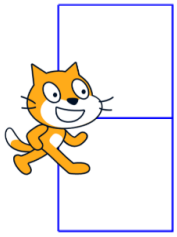
Tabela 5.24 - Categorias de depuração – quadrado no *Scratch*.

Aspecto	Categorias	Frequência
Depuração	Ausência de comandos para apagar os rastros da caneta.	40%
	Ausência de comandos de evento para iniciar o programa.	20%
	Ausência de mensagens que informem ao usuário qual comando inicia a execução do programa.	30%
	Presença de comandos irrelevantes.	20%
	Erros graves que comprometem a construção do quadrado.	20%

Fonte**:** Autores.

O primeiro erro consiste na ausência de comandos para apagar os traços da caneta. Isso compromete a construção do quadrado caso o programa seja executado mais de uma vez, pois o desenho anterior permanecerá na tela. Na Figura 5.22 apresentamos um exemplo. 40% dos algoritmos apresentam esse problema (*N1*, *N2*, *N3* e *D1*).

Figura 5.22 - Problema 1: Ausência de comandos para apagar traços.



Fonte**:** Autores.

Destacamos que mesmo sem ter domínio sobre o *software* uma considerável parcela dos sujeitos foi capaz de perceber que precisavam usar o comando “apague tudo” para limpar a tela antes de construir um novo quadrado.

O segundo problema é a ausência de comandos de evento para iniciar o programa. Isso significa que o usuário precisa acessar a interface de programação e executar o algoritmo manualmente com um duplo clique. Ao todo 20% das produções apresentam esse problema (*N1* e *D1*).

A ausência de comandos de evento para iniciar o programa é um indicativo de que alguns discentes não testaram o programa como usuário, apenas como programadores. Apesar de qualquer pessoa conseguir ter acesso à interface de programação, a interface de usuário do *Scratch* consiste no palco. Ou seja, os comandos de evento são importantes para que um usuário consiga dar início ao programa sem acessar seu código base.

Na terceira categoria agrupamos os algoritmos que tem comandos de evento para iniciar o programa, mas não utilizam a bandeira verde como gatilho. Além disso, não informam ao usuário como iniciar o programa. Observamos esse problema em 30% das produções (*I1*, *I2* e *N3*).

Intuitivamente a bandeira verde é utilizada para iniciar o código, assim a mudança da bandeira para “quando este ator for clicado” ou “quando a barra de espaço for clicada” precisa ser comunicada ao usuário de algum modo para que este saiba como executar o programa.

A quarta categoria reúne algoritmos onde há comandos que não desempenham nenhuma função prática. Observamos que 20% dos algoritmos apresentam esse problema (*D1* e *D2*).

O programa *D1* apresenta comandos de caneta excessivos que poderiam ser sintetizados para economizar linhas de código. Esse dado indica que uma parcela dos discentes teve dificuldade para refinar os algoritmos que produziram, isto é, perceber quais comandos eram desnecessários e poderiam ser eliminados.

O programa *D2* executa o comando “use a caneta” imediatamente após o comando “levante a caneta” anulando seu efeito prático. Ademais, não há nenhum tempo de espera entre os comandos “mude para a fantasia 2” e “mude para a fantasia 1”, fazendo com que a animação pretendida não se concretize, pois o *software* faz a troca de fantasias em milésimos de segundo.

Observamos também que o comando para determinar a posição inicial do ator é repetido sem necessidade.

Além disso, o programa conta com um comando para alterar a cor da caneta, no entanto o usuário não tem como fazer essa alteração (exceto se o fizer manualmente). Provavelmente isso ocorreu porque os discentes não sabiam como inserir comandos de interação com o usuário, ou não perceberam que precisavam fazer isso, uma vez que alteravam a cor da caneta diretamente no código.

A quinta categoria é formada pelos programas que não desenham um quadrado, 20% dos algoritmos compõem essa categoria (*N3* e *D1*). O algoritmo *N3* desenha apenas uma das arestas do quadrado. A produção *D1* demarca os vértices e desenha três arestas do quadrado, contudo, esse algoritmo também desenha segmentos externos ao quadrado.

A dupla *D1* tentou replicar o algoritmo produzido em linguagem materna usando os comandos do *Scratch*. No entanto, o primeiro algoritmo foi pensado usando régua e compasso e o *software* não tem nenhuma função que se assemelhe a um compasso. Apesar de terem sucesso em marcar os pontos dos vértices o algoritmo que elaboraram não chegou a desenhar todos os lados do quadrado.

Observamos que 20% dos discentes não conseguiram produzir um algoritmo que desenhasse o quadrado. É interessante pontuar que isso não ocorreu no caso da linguagem materna, sugerindo que a dificuldade dos discentes pode ter sido utilizar a linguagem do *Scratch*.

Kilham e Brâting (2019) apontam que a diferença entre sistemas simbólicos pode gerar dificuldades aos alunos, entendemos que esse resultado corrobora com essa fala. Por outro lado, observamos também que uma parcela considerável dos sujeitos conseguiu fazer uso da linguagem simbólica para reproduzir seus algoritmos em linguagem materna ou similares.

5.2.4 Triângulo Escaleno no *Scratch*

Nessa subseção apresentamos e discutimos os algoritmos do triângulo escaleno produzidos no *Scratch*. Os algoritmos foram analisados em relação aos aspectos algébricos e computacionais, conforme apresentado no Quadro 5.5.

Nessa subseção aplicamos o critério de estrutura de forma diferente da utilizada na análise dos algoritmos do quadrado no *Scratch*. No primeiro observamos o modo com que o quadrado foi construído, nesse caso analisamos como os discentes coletaram os dados sobre os comprimentos das arestas. As análises sobre a estrutura e objetificação foram sintetizadas na Tabela 5.25.

Tabela 5.25 - Categorias de estrutura e objetificação – triângulo escaleno no *Scratch*.

Aspecto	Categorias	Frequência
Estrutura	Coleta de dados baseada em controles deslizantes.	60%
	Coleta de dados baseada em comandos do tipo pergunte.	40%
	Uso de condicionais lógicas para restringir os dados de entrada.	20%
Objetificação	Ângulo suplementar a um ângulo interno, vértice inicial e duas arestas do triângulo.	80%
	Vértices do triângulo.	10%
	Dois vértices e uma aresta do triângulo.	20%

Fonte**:** Autores.

Em relação à *estrutura* observamos três categorias. A primeira reúne os algoritmos onde os comprimentos das arestas são modificados por meio de controles deslizantes. Na Figura 5.23 apresentamos exemplos desse recurso.

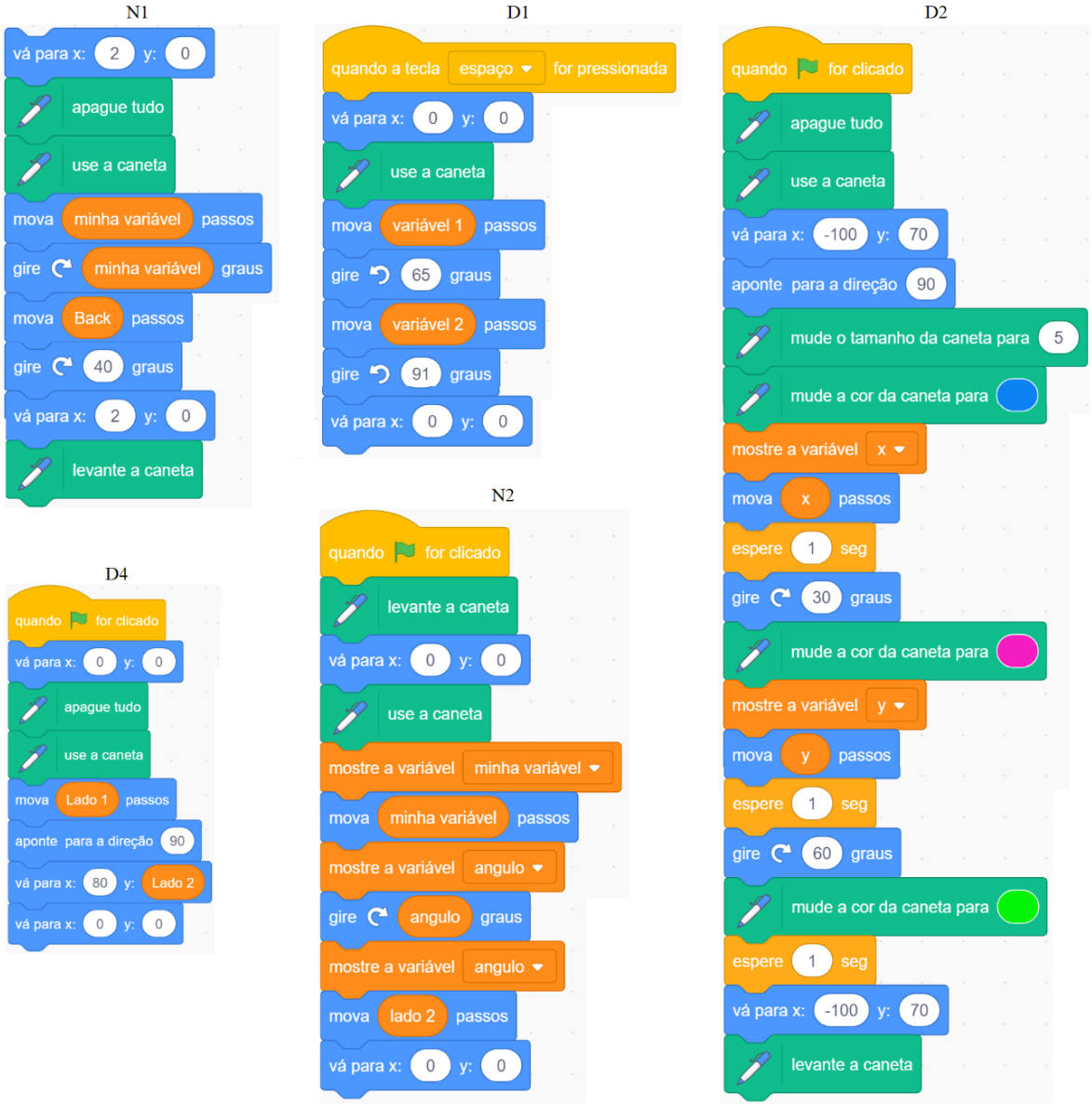
Figura 5.23 - Controles deslizantes utilizados pelos discentes.

Fonte**:** Autores.

Quando uma variável é criada o *software* automaticamente apresenta um controle deslizante vinculado a ela, supomos que é por essa razão que esse foi o meio de coleta mais utilizado.

Esses controles foram fixados no palco e permitem que o usuário altere o valor de determinadas variáveis. Todas os algoritmos relacionados a essa categoria são apresentados na Figura 5.24, essas produções representam 50% (*N1*, *N2*, *D1*, *D2* e *D4*) da amostra.

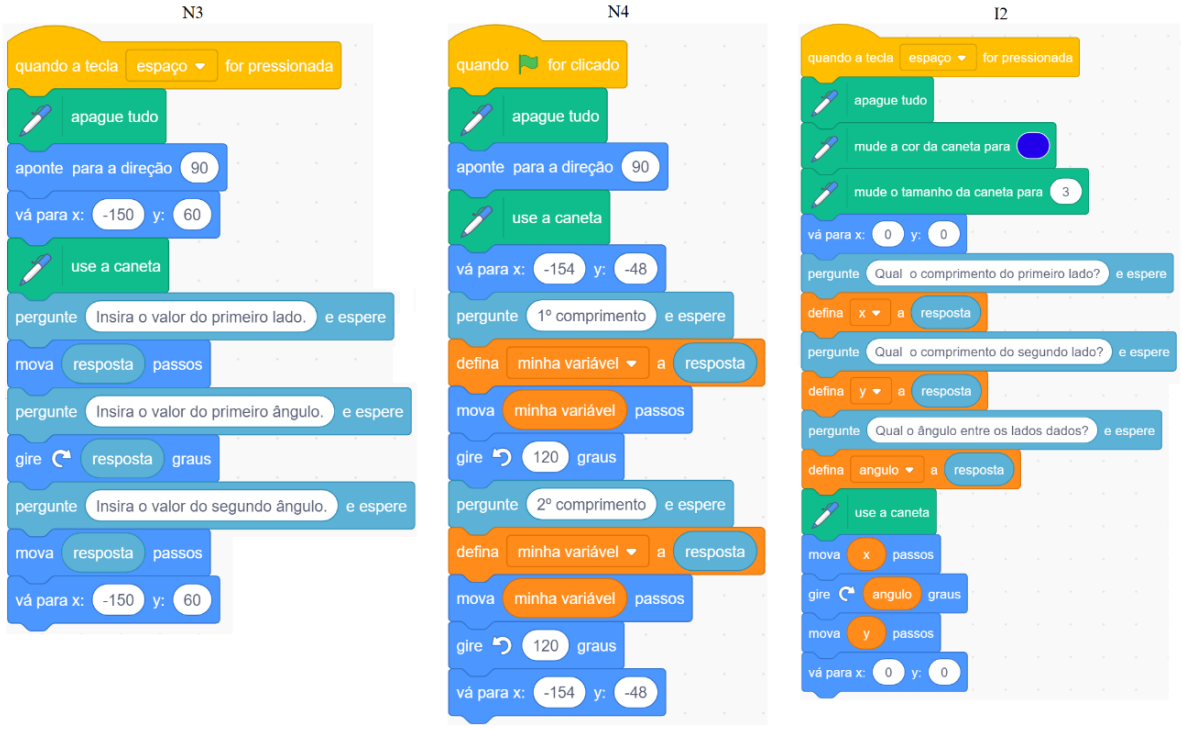
Figura 5.24 - Construção de triângulo escaleno usando controles deslizantes.



Fonte: Autores.

Na segunda categoria agrupamos os algoritmos na qual os comprimentos do triângulo escaleno são definidos usando comandos do tipo *pergunte*. Nesses programas o usuário modifica o tamanho dos segmentos respondendo a perguntas, assim o valor da variável é atualizado para o número inserido como resposta. Na Figura 5.25 apresentamos os programas que foram relacionados a essa categoria.

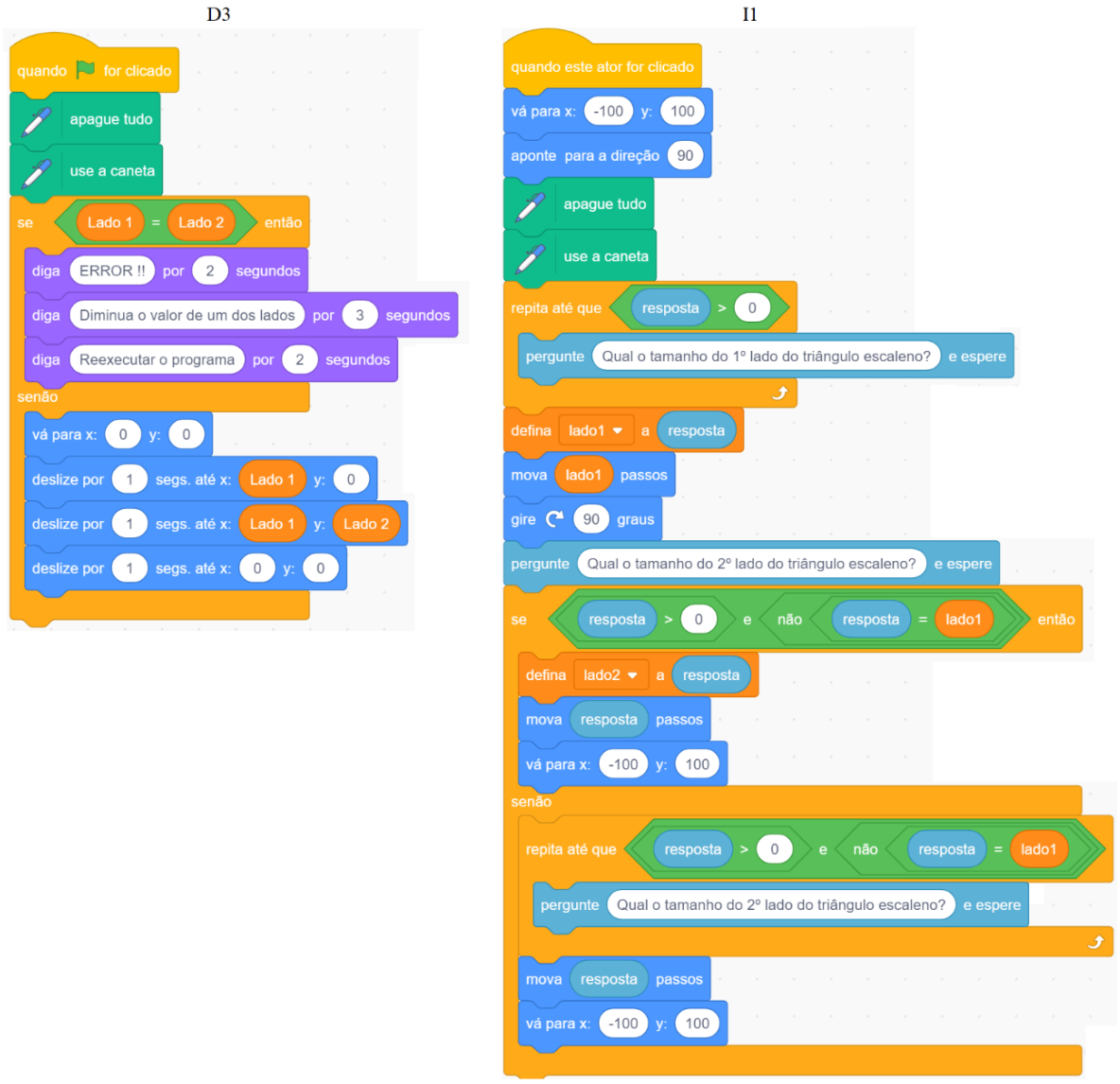
Figura 5.25 - Construção de triângulo escaleno usando comandos do tipo pergunte.



Fonte**:** Autores.

A terceira categoria engloba algoritmos com restrições sobre o comprimento dos lados do triângulo, 20% das produções se encaixam nessa categoria (*I1* e *D3*). Na Figura 5.26 apresentamos esses algoritmos.

Figura 5.26 - Programas com restrições sobre os valores de entrada.



Fonte: Autores.

Por meio da Tabela 5.25 e a sequência de figuras de 5.24 até 5.26, notamos que a primeira categoria, que apresenta um controle deslizante vinculado à variável foi o meio de coleta mais utilizado.

A coleta por meio de perguntas não é tão intuitiva, pois exige que o sujeito perceba a relação entre o valor informado pelo usuário e o bloco “resposta”, e que esse valor pode ser atualizado conforme novas perguntas sejam realizadas.

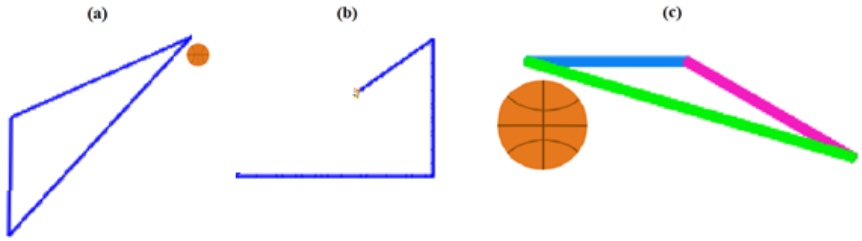
Apesar de menos intuitivo, a coleta por meio de perguntas permite que o usuário insira um conjunto de valores maior que o conjunto gerado pelo controle deslizante. Visto que, os

controles deslizantes só aceitam valores inteiros e o comando pergunte permite que o usuário insira números racionais.

Destacamos também que todos os sujeitos da primeira categoria mantiveram os controles deslizantes com o intervalo numérico padrão, isto é, o mínimo aceito era 0 e o máximo 100. Inferimos que isso ocorreu porque os discentes não perceberam que podiam alterar esse intervalo.

Na sequência apresentamos três palcos das produções dos sujeitos, é importante ressaltar que os exemplos tomados não são representantes de cada categoria apresentada anteriormente. Na Figura 5.27 apresentamos exemplos dos desenhos gerados pelos programas analisados nessa seção, sendo que: (a) corresponde aos programas *I1*, *I2*, *N1*, *N2*, *N3*, *N4*, *D1* e *D4*; (b) ao programa *D3*; e (c) ao programa *D2*.

Figura 5.27 - Palco dos programas que desenham um triângulo escaleno.



Fonte**:** Autores.

Os programas da parte (a) da Figura 5.27 desenham o triângulo de forma direta, as arestas simplesmente aparecem na tela sem nenhum tipo de animação. Na parte (b) o programa desenha o triângulo simulando o movimento de uma caneta, ou seja, o usuário vê o segmento se movimento entre dois vértices. O programa da parte (c) desenha o triângulo sem animações e de forma direta, porém com arestas coloridas.

No tocante a *objetificação* observamos três categorias. Na primeira os aspectos objetificados são: ângulo suplementar a um dos ângulos internos, vértice inicial e duas arestas do triângulo. 80% dos algoritmos estão associados a essa categoria (*I1*, *I2*, *N1*, *N2*, *N3*, *N4*, *D1* e *D2*). A segunda engloba produções que objetificaram os vértices do triângulo, ao todo 10% dos algoritmos compõem essa categoria (*D3*).

Na terceira categoria reunimos as produções que objetificaram dois vértices e uma aresta do triângulo, essa categoria corresponde a 10% dos algoritmos (*D4*). A quarta categoria agrupa

os algoritmos que objetificam a propriedade que caracteriza o triângulo escaleno: ter três lados diferentes. Essa categoria representa 20% das produções (*I1* e *D3*).

Os aspectos evidenciados nesses algoritmos são semelhantes aos apontados nos algoritmos em linguagem materna, contudo, nesse caso nenhum dos discentes objetificou todas as arestas do triângulo. Supomos que isso ocorreu porque as produções em linguagem materna foram pensadas sobre instrumentos como régua e compasso, e embora o *Scratch* possa ser usado para simular uma régua o mesmo não ocorre com o compasso.

Em relação as análises sobre simbolização e generalização, apresentaremos na sequência uma síntese das categorias encontradas. Constatamos cinco categorias em relação a *simbolização*, os resultados são apresentados no Quadro 5.7.

Quadro 5.7 - Categorias de simbolização – triângulo escaleno no *Scratch*.

Aspecto objetificado	**Simbolização**	**Frequência**
Arestas do triângulo	Comando “mova __ passos”.	90% (*I1*, *I2*, *N1*, *N2*, *N3*, *N4*, *D1*, *D2* e *D4*).
Comprimento da aresta	Bloco de variável.	90% (I1, I2, N1, N2, N4, D1, D2, D3 e D4).
	Bloco de resposta.	40% (I1, I2, N3 e N4).
Vértices do triângulo	Comando “vá para x:__ y:__”.	90% (*I1*, *I2*, *N1*, *N2*, *N3*, *N4*, *D1*, *D2* e *D4*).
	Comando “deslize por __ segs. até x:__ y:__”.	10% (*D3*).
Ângulos suplementar	Comando “gire __ graus”.	80% (*I1*, *I2*, *N1*, *N2*, *N3*, *N4*, *D1* e *D2*).
Comprimento do ângulo suplementar	Bloco de variável.	30% (I2, N1 e N2).
Diferença no comprimento dos lados	Combinações de comandos de condicional, controle e operadores.	20% (*I1* e *D3*).

Fonte**:** Autores.

Como todos os discentes utilizaram o mesmo conjunto de comandos a classificação dos diferentes tipos de simbolização foi facilitada, assim conseguimos estabelecer mais

subcategorias do que nos algoritmos de linguagem materna. Além disso, conseguimos identificar os meios de coleta de dados utilizados pelos acadêmicos.

Nesse caso todos os algoritmos precisavam incorporar ao menos uma variável, tornando-os mais complexos que os produzidos para desenhar o quadrado. Um reflexo dessa complexidade pode ser percebido nas categorias de simbolização, Quadro 5.7. Diferentemente do quadrado, nessa situação há categorias específicas para os comprimentos das arestas e de ângulos suplementares.

No tocante a *generalização* observamos uma única categoria, 100% dos algoritmos estabeleceram generalizações simbólicas, pois as variáveis foram enunciadas por meio de símbolos alfanuméricos (RADFORD, 2006).

Vale destacar que em 80% dos algoritmos dessa categoria (*I2*, *N1*, *N2*, *N3*, *N4*, *D1*, *D2* e *D4*) as características do triângulo escaleno não foram percebidas em sua totalidade, visto que os programas permitem a geração de triângulos que não são isósceles ou violações nas condições de existência de um triângulo.

Em relação a *depuração*, observamos seis categorias de problemas lógicos. Esses dados são apresentados na Tabela 5.26.

Tabela 5.26 - Categorias de depuração – triângulo escaleno no Scratch.

Aspecto	**Categorias**	**Frequência**
Depuração	Ausência de comandos para apagar os rastros da caneta.	20%
	Ausência de comandos de evento para iniciar o programa.	10%
	Ausência de mensagens que informem ao usuário qual comando inicia a execução do programa.	40%
	Presença de comandos irrelevantes.	40%
	Controles deslizantes só são apresentados ao usuário após a primeira execução do programa.	30%
	Arestas distintas podem ter comprimentos iguais, ou ser iguais ou menores que 0.	90%

Fonte**:** Autores.

O primeiro erro consiste na ausência de comandos para apagar os traços da caneta, 20% dos algoritmos apresentam esse problema (*N2* e *D1*). A segunda, representando 10% dos

algoritmos, é composta pelos algoritmos que não contam com comandos de evento para iniciar o programa (*N1* e *D1*).

Durante a atividade instruímos os acadêmicos a se certificarem que o usuário tenha como alterar os comprimentos do triângulo sem acessar a programação, ou seja, que criassem meios de coletar os dados usando a interface de usuário. Essa condição pode ter levado os participantes a testar seus programas como usuários, não somente como programadores, fazendo com que percebessem a necessidade dos comandos de evento.

Na terceira categoria agrupamos os algoritmos que possuem comandos de evento, mas não utilizam a bandeira verde como gatilho e não informam ao usuário como iniciar o programa. 40% dos programas se encaixam nessa categoria (*I1*, *I2*, *N3* e *D1*).

A quarta categoria reúne algoritmos onde há comandos que não desempenham nenhuma função prática no programa. Observamos que 40% dos algoritmos apresentam esse problema (*N1*, *N4*, *D2* e *D4*).

O último comando “gire __ graus” nos programas *N1*, *N4* e *D2* não tem nenhuma influência sobre a construção do triângulo e pode ser removido sem prejuízos. O mesmo acontece com o comando “aponte para a direção __” do programa *D4*.

Apesar do programa *N4* enunciar duas variáveis, “*minha variável”* e “*resposta”*, na prática apenas uma é utilizada, pois as duas são igualadas a cada atualização de valor. Assim, a variável “*minha variável*” e os comandos “defina *minha variável* a *resposta*” podem ser removidos sem prejuízos.

Para elaborar os algoritmos do triângulo escaleno no *Scratch*, os discentes tiveram que elaborar mecanismos de coleta de dados, além disso, desta vez estavam trabalhando com uma figura geométrica cujos ângulos e lados tem medidas diferentes. Supomos que esses dois aspectos podem ter contribuído para o aumento da porcentagem de programas com comandos irrelevantes.

A quinta categoria é composta pelos algoritmos onde os controles deslizantes só são apresentados ao usuário após a primeira execução do programa, ela representa 30% da amostra (I1, N2 e D2). Isso significa que as arestas do primeiro triângulo têm comprimento fixo, o comprimento das arestas é variável somente a partir da segunda execução do programa.

Na sexta categoria agrupamos as produções onde as arestas podem ser iguais ou menores que zero. Além disso, os programas permitem que arestas distintas tenham o mesmo comprimento, violando a definição de triângulo escaleno. 90% das produções foram associadas a essa categoria (*I2*, *N1*, *N2*, *N3*, *N4*, *D1*, *D2*, *D3* e *D4*).

Os últimos dois tipos de erro não ocorreram anteriormente porque estão relacionados à coleta de dados. O problema com os controles deslizantes provavelmente foi ocasionado pela falta de familiaridade dos discentes com o *software*. As violações da condição de existência podem ser resultado do pouco domínio dos sujeitos sobre a linguagem do *Scratch*, contudo, o mesmo problema foi observado nos algoritmos em linguagem materna indicando que também estão relacionadas à percepção dos acadêmicos sobre o triângulo escaleno.

É interessante pontuar que todos os discentes conseguiram produzir algoritmos que desenham triângulos escalenos no *Scratch*, o mesmo não ocorreu quando utilizaram a linguagem materna. Supomos que a linguagem do *software* evitou que os sujeitos cometessem erros conceituais como os observados no algoritmo de linguagem materna.

CONSIDERAÇÕES FINAIS

Este estudo buscou responder a seguinte pergunta: *quais aspectos do PC e PA são evidenciados por licenciandos em Matemática na realização de atividades com o Scratch?* Para isso realizamos um estudo de caso exploratório de abordagem mista. Os dados foram coletados no decorrer de uma sequência didática, o método de análise foi a Análise de Conteúdo (BARDIN, 2011) e as discussões foram alicerçadas em referenciais teóricos que exploram esses dois tipos de pensamento.

Por meio do questionário pré-implementação constatamos que a maioria dos discentes apresentou concepções ingênuas sobre PC ou afirmou desconhecer esse termo, o mesmo pode ser dito sobre os conceitos fundamentais do PC apontados por Brennan e Resnick (2012). Ademais, uma significativa parcela dos discentes não percebeu relações entre o PC e o PM, ou estabeleceu relações incoerentes entre esses tipos de pensamento.

Esses dados corroboram com nosso pressuposto de que *a maioria dos sujeitos desconhece o que é PC e sua relação com a Matemática*. Esses resultados eram esperados, uma vez que esses tópicos não costumam serem abordados no curso de Licenciatura em Matemática.

Uma significativa parcela dos sujeitos afirmou desconhecer os termos: sequência/algoritmo, condições lógicas, operadores, dados, processos iterativos e variáveis. Esses conceitos, com exceção de processos iterativos, são amplamente trabalhados de forma implícita ou explícita na licenciatura em Matemática. Em vista disso, entendemos esse dado como um indicativo de que alguns discentes tem dificuldade em identificar e definir conceitos matemáticos fundamentais, uma vez que todos os sujeitos estão familiarizados com esses conceitos. Outra perspectiva que podemos inferir, refere-se ao problema em relacionar conceitos que são vistos na matemática atrelados a outra área de conhecimento.

Vale ressaltar que, apesar dos acadêmicos não terem conhecimentos prévios sobre PC ou conhecimentos formais sobre PM, uma parcela dos sujeitos apresentou concepções coerentes sobre o PC e conseguiu perceber conexões consistentes entre esses tipos de pensamento.

Em relação ao PM, observamos que a maioria dos discentes apresentou noções coerentes sobre esse pensamento, contudo, as questões tipo Likert evidenciaram algumas inconsistências nas concepções observadas nas questões abertas. Uma parcela minoritária dos sujeitos ora caracteriza o PM como imutável e procedimental, ora o como criativo e flexível.

Supomos que isso ocorreu porque esses discentes não distinguem o conhecimento matemático formal do PM, isto é, atribuem ao PM a rigidez e formalismo da Matemática formal enquanto reconhecem a necessidade de ser criativo e versátil na abordagem de problemas.

Apesar dessas constatações não temos dados que nos permitam inferir o entendimento dos discentes sobre PA, uma vez que optamos por questioná-los sobre PM. Assim, não temos como confirmar se *os licenciandos desconhecem o conceito de PA e sua relação com a programação*, um de nossos pressupostos.

Quanto à relação dos sujeitos com a programação, identificamos que 64% da amostra não tem experiência com programação e 57% não tem nenhuma noção de programação. Ademais, considerando os resultados obtidos na análise dos códigos do *Scratch*, constatamos que a maioria dos sujeitos é principiante em programação, mesmo que alguns tenham relatado possuir conhecimento prévio sobre o tema. Esses dados corroboram com nosso pressuposto de que *a maioria dos licenciandos não tem familiaridade ou experiência com programação*. Todavia salientamos que uma minoria dos discentes demonstra ter domínio intermediário ou avançado nesse campo.

Vale destacar que o perfil dos acadêmicos do turno integral e noturno são significativamente diferentes em relação a programação. Os discentes com maior domínio sobre informática e familiaridade com programação se concentram na turma do integral, enquanto todos os licenciandos do turno noturno relatam não ter experiência alguma com programação.

Recorremos novamente à literatura para responder à pergunta auxiliar*: quais relações podem ser estabelecidas entre o PC e PA?* O estudo de Kilhamn e Brâting (2019) apontou os seguintes pontos comuns entre PC e PA: estrutura, simbolização, generalização e abstração, funções e variáveis e algoritmos. Apesar desses termos serem comuns entre os dois pensamentos, não está claro se representam as mesmas ideias nas duas áreas.

Além destes, consideramos que a depuração e modularização também possam ser pontos comuns. Esses conceitos são fundamentais para o PC, porém não são enunciados de forma explícita no PA.

Essas considerações e resultados nos permitem afirmar que o objetivo específico *apontar relações entre PC e PA* foi atingido. Contudo, ressaltamos que nossa pesquisa não nos permite inferir relações entre PA, depuração e modularização, para refutar ou confirmar essas possibilidades é necessário investigá-las em outro estudo.

Recorremos à literatura para responder à pergunta auxiliar*: Como o PA pode ser analisado em projetos do Scratch?* Com base nas pesquisas de Radford (2006) e Kaput (2008)

observamos três aspectos que guiaram nossas análises: objetificação, simbolização e generalização.

Na objetificação analisamos quais aspectos dos objetos matemáticos foram evidenciados nos algoritmos. Nessa pesquisa analisamos programas que abordam os objetos de forma explícita, entretanto, esse aspecto também pode ser avaliado em programas que abordem a matemática de forma implícita, uma vez que o usuário ainda recorrerá a algum conhecimento matemático para elaborá-los.

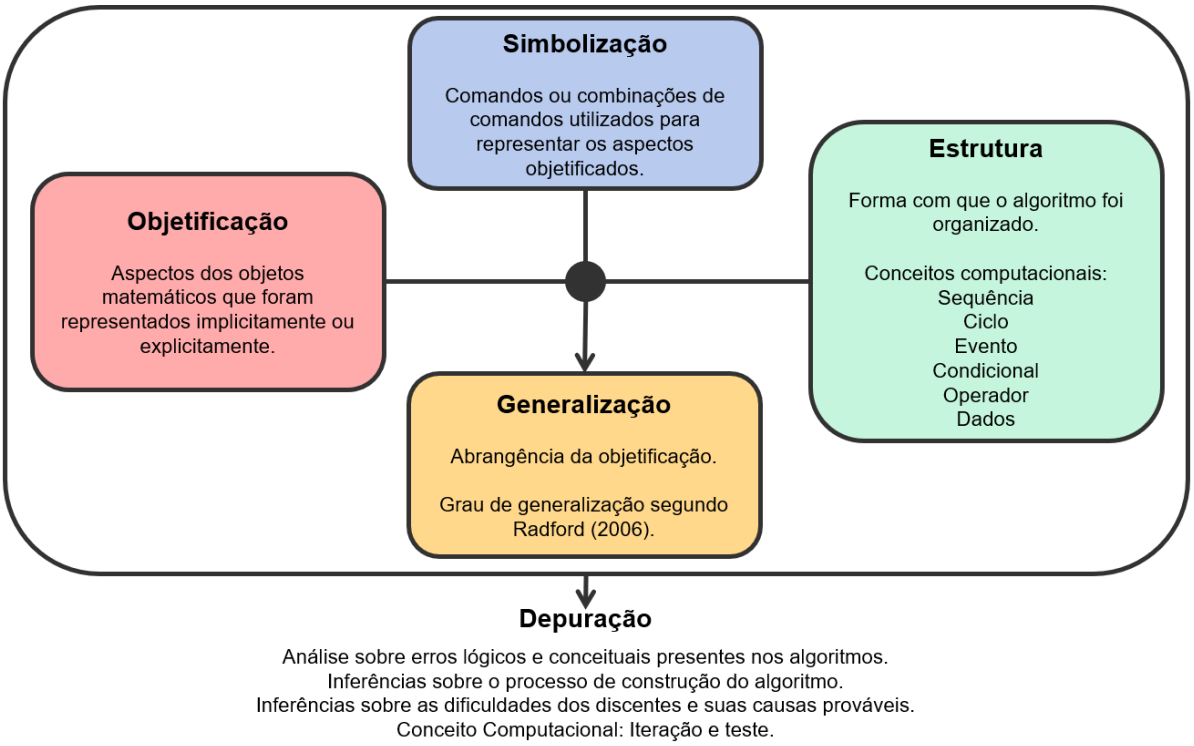
Os blocos do *Scratch* podem ser usados para representar relações algébricas, como evidenciado nesse estudo. Assim, na simbolização observamos quais comandos ou combinações de comandos os discentes utilizaram como meio de representação. Ressaltamos que a objetificação e a simbolização tem uma relação intrínseca, uma vez que os objetos matemáticos só podem ser apreendidos por meio da representação, isto é, a simbolização precisa evidenciar a característica objetificada de forma explícita ou implícita.

O terceiro aspecto do PA analisado foi a generalização. Analisamos os algoritmos em relação ao grau de generalização estabelecido pelo sujeito sobre determinado aspecto matemático. Vale destacar que essa análise também pode ser feita sobre algoritmos que trabalham com a matemática de forma indireta. Nessa pesquisa recorremos às camadas de generalização de Radford (2006) para avaliar os algoritmos.

Em relação à pergunta norteadora *quais aspectos do PC e PA são evidenciados por licenciandos em Matemática na realização de atividades com o Scratch?* Observamos dois aspectos gerais nos algoritmos: algébricos e computacionais.

Os aspectos algébricos analisados foram objetificação, simbolização e generalização, enquanto os computacionais foram estrutura e depuração. Cada um desses aspectos foi analisado segundo os indicadores apresentados no Quadro 5.5., vale destacar que esses indicadores foram estabelecidos com base no referencial teórico e na capacidade de observação do pesquisador. Na Figura 6.1 apresentamos um esquema que representa as relações observadas entre os aspectos algébricos e computacionais..

Figura 6.1 - Aspectos algébricos e computacionais observados nos algoritmos



Fonte**:** Autores.

É interessante destacar que a objetificação, simbolização e estrutura refletem diretamente no tipo de generalização estabelecida. Além disso todos esses aspectos são contemplados pela depuração, uma vez que para identificar erros, entender suas causas e corrigi-los é necessário pensar no que foi objetificado, como isso foi simbolizado usando a estrutura e se o código contempla o nível de generalização que se objetiva atingir.

Na sequência apresentaremos uma síntese de cada um dos aspectos em relação aos quatro algoritmos produzidos, a construção do quadrado em linguagem materna e no *Scratch* e a construção do triângulo em linguagem materna e no *Scratch*.

Em relação a *objetificação*, nos *quadrados em linguagem materna* as categorias diferenciaram-se em relação a forma com que diferentes características desse objeto foram combinadas para gerar o desenho. Por exemplo, algumas produções se basearam nos vértices e ângulos internos, enquanto outras utilizaram arestas e ângulos internos.

Constatamos também que os *quadrados feitos no Scratch* refletiram os mesmos aspectos observados nas produções em linguagem materna, supomos que isso aconteceu porque os discentes basearam seus programas nos algoritmos feitos a mão.

Nos *triângulos em linguagem materna* as categorias se distinguiram em relação aos aspectos, ou seja, algumas produções objetificaram somente vértices e arestas, enquanto outras representaram vértices, arestas e ângulos suplementares. Os aspectos evidenciados nos *triângulos feitos no Scratch* são semelhantes aos dos algoritmos feitos a mão, contudo, nesse caso nenhum dos sujeitos objetificou todas as arestas do triângulo.

As produções em linguagem materna foram pensadas sobre instrumentos como régua e compasso, e embora o *Scratch* possa simular uma régua o mesmo não ocorre com o compasso. Assim, os discentes não puderam gerar os vértices a partir da intersecção de semicírculos (gerados com o compasso), sendo forçados a recorrer apenas à comprimentos de arestas e ângulos. Em vista disso supomos que, diferentemente do caso dos quadrados, os discentes não conseguiram adaptar seus algoritmos feitos a mão para a linguagem do *software*.

No tocante à *simbolização*, destacamos que nos *algoritmos em linguagem materna* a maioria dos sujeitos utilizou meios semióticos próprios, enquanto nas produções feitas no *Scratch* os discentes foram forçados a utilizar um sistema simbólico formal. Como todos utilizaram o mesmo conjunto de comandos a classificação dos diferentes tipos de simbolização foi facilitada nos programas do *Scratch*, assim conseguimos estabelecer mais subcategorias do que nas produções feitas à mão.

Observamos que as categorias dos *algoritmos em linguagem materna* se diferenciam pelo grau de formalidade e pela forma (direta ou indireta) com que os objetos são apresentados. Já nos *algoritmos do Scratch* as categorias foram estabelecidas em relação às combinações de comandos que os discentes utilizaram.

Em relação à *generalização*, nos *algoritmos em linguagem materna* as categorias se diferenciaram pelo tipo de simbologia utilizado (formal ou semiformal), pela forma com que as características do objeto foram abordadas (direta ou indireta) e, pela coerência lógica entre os comandos e a definição do objeto.

Nas *produções feitas no Scratch* observamos se a variável foi enunciada, se o algoritmo respeitou as condições de existência do objeto e, se os sujeitos estabeleceram a generalização por adivinhação.

O aspecto *estrutura* está associado aos conceitos computacionais sequência, ciclo, evento, condicional, operador e dados, uma vez que eles são utilizados para dar corpo ao algoritmo. A forma com que empregamos esse critério variou conforme as peculiaridades de cada produção.

Nos *algoritmos em linguagem materna* os discentes utilizaram meios de representação semiótica diversos, isso dificultou uma análise comparativa entre os comandos. Em vista disso, no critério *estrutura* observamos como os sujeitos organizaram seus algoritmos.

Os *quadrados feitos no Scratch* foram analisados em relação à forma com que essa figura geométrica foi desenhada, optamos por esse critério porque observamos regularidades sobre essa característica. Contudo, esse foco não nos pareceu apropriado para avaliar os *triângulos no Scratch*, uma vez que 90% das produções se baseou no mesmo método. Assim, nesse caso optamos por observar como os discentes coletaram os dados sobre as variáveis do problema. Vale ressaltar que a objetificação, simbolização e estrutura estão intrinsecamente relacionadas, pois a ordem e a combinação dos blocos de comando influenciam a simbolização.

Além disso, observamos que os algoritmos com estruturas mais complexas foram produzidos pelos sujeitos *I1*, *I2* e pela dupla *D3*. Os indivíduos *I1* e *I2* declaram ter conhecimento prévio e interesse sobre programação. A dupla *D3* é composta por discentes que relataram ter interesse sobre programação, porém não possuem conhecimento prévio sobre o assunto.

A *depuração* contempla todos os conceitos computacionais relacionados a estrutura, e, além desses abrange as práticas de iteração e teste. Nesse critério utilizamos os erros lógicos e conceituais presentes nos algoritmos para inferir sobre o processo de construção deles.

As análises sobre esse aspecto nos permitiram identificar dificuldades dos sujeitos em relação à: *compreensão e simbolização dos objetos matemáticos*, e *elaboração dos algoritmos*. Além disso, a depuração nos auxiliou a inferir sobre as razões que levaram os discentes a cometer esses erros.

Em relação às *dificuldades de simbolização* destacamos duas situações. A primeira se refere à uma dupla que conseguiu produzir o quadrado usando a linguagem materna, contudo, ao utilizar a linguagem do *Scratch* não concluiu essa tarefa. A segunda remete a duas duplas cujos algoritmos em linguagem materna não desenharam triângulos escalenos, mas, quando utilizaram o *software* conseguiram.

Em nossa visão produzir um algoritmo sobre um polígono irregular é mais complexo do que sobre uma figura regular, independentemente do tipo de linguagem utilizada. Um reflexo dessa complexidade pode ser percebido nas categorias de simbolização dos *triângulos produzidos no Scratch*, diferentemente do quadrado, nessa situação há categorias específicas para os comprimentos das arestas e dos ângulos suplementares.

Destacamos também que a programação do quadrado foi o primeiro contato desses sujeitos com o *software*. Diante disso, supomos que a primeira situação ocorreu porque a dupla desconhecia o *Scratch* e não estava familiarizada com essa linguagem de programação.

Um algoritmo em linguagem materna é precisa ser testado manualmente, seja pelo sujeito que o fez ou alguma outra pessoa. Ademais, nesse tipo de produção o sujeito pode cometer vários erros sintáticos e lógicos, uma vez que ele é livre para elaborar qualquer tipo de comando e organizá-los como preferir.

O *Scratch* evita que o usuário cometa erros sintáticos. Além disso, o indivíduo pode testar o programa a qualquer momento e ver o resultado no palco de forma imediata, isso permite que erros sejam detectados e corrigidos com mais facilidade. Como os algoritmos tinham por objetivo desenhar figuras geométricas, esse recurso se torna ainda mais prático e intuitivo.

Assim, presumimos que a segunda situação ocorreu porque o *software* impediu que os acadêmicos cometessem erros sintáticos e, os ajudou a corrigir erros que passaram despercebidos quando usaram linguagem materna. Ademais, o triângulo foi o segundo projetos dos discentes no *Scratch*, ou seja, os sujeitos já estavam mais familiarizados com a linguagem do programa.

Em relação as *dificuldades de compreensão*, observamos tanto nos *algoritmos do triângulo feitos à mão* quanto nos *produzidos no Scratch* que a maioria dos discentes não considerou as condições de existência desse objeto. Esse dado indica que os sujeitos não

perceberam o papel fundamental que as condições de existência têm sobre o processo de construção.

Para além disso, partindo dessa observação podemos refletir se há ou não alguma lacuna conceitual no entendimento dos discentes sobre esse objeto. Isto é, se eles percebem que não podemos selecionar três comprimentos distintos de forma arbitrária para obter um triângulo escaleno, é preciso certificar-se que essas medidas cumpram com determinadas condições, caso contrário será impossível formar um triângulo.

Em relação às dificuldades de *elaboração dos algoritmos*, destacamos três situações. A primeira se refere a dificuldade que os sujeitos tiveram em organizar seus *algoritmos em linguagem materna* de modo que cada comando executasse uma única ação. Vale destacar que provavelmente isso ocorreu porque no curso de licenciatura em Matemática a produção de algoritmos não é comum.

A segunda situação remete os erros relacionados aos comandos de evento e caneta observados em alguns programas. Uma significativa parcela das categorias de depuração dos programas do *Scratch* está relacionada a esse tipo de comando. Isso era esperado uma vez que os discentes tiveram que explorar o *software* por conta própria, e a maioria deles não tiveram experiência prévia com o *Scratch*.

A terceira situação se refere aos comandos desnecessários. No *quadrado do Scratch* 20% das produções apresentaram esse problema, supomos que ocorreu porque os discentes desconheciam o *software*. No *triângulo do Scratch*, segundo projeto dos discentes no *software*, observamos que esse percentual subiu para 40%.

Para elaborar esse algoritmo os discentes tiveram que estabelecer mecanismos de coleta de dados, pois exigimos que os comprimentos fossem variáveis. Ademais, essa figura geométrica tem ângulos e lados com medidas diferentes, isto significa que os discentes precisaram considerar mais de uma variável. Diante disso, presumimos que o aumento percentual se deve a essas razões.

Esses resultados nos permitem afirmar que os objetivos *delinear indicadores de PC e PA em algoritmos produzidos em linguagem Materna e no Scratch*, e *apontar aspectos do PC e PA mobilizados por licenciandos em Matemática ao realizar atividades com o Scratch* foram alcançados.

Em relação a *metodologia adotada nessa pesquisa*, vale destacar que a ausência de uma definição universal de PC na literatura, e de um estudo que defina indicadores de PA em programas do *Scratch* dificultou nosso processo de análise. Em razão disso elaboramos e aplicamos procedimentos próprios para avaliar o PC e o PA nos algoritmos analisados, contudo, isso dificulta a comparação dos resultados desse estudo com os de outras pesquisas na área.

Dentre os *desafios enfrentados* destacamos a dificuldade em não direcionar os sujeitos para uma linha de raciocínio específica, e conseguir a participação de todos os discentes nas atividades da sequência. Mesmo coletando a maioria dos dados nas aulas de algumas disciplinas do curso não conseguimos garantir a frequência dos sujeitos, nas atividades realizadas de forma remota tivemos um retorno ainda menor.

Em relação aos nossos *instrumentos de coleta*, ressaltamos que gostaríamos de ter feito algumas perguntas tipo Likert de forma diferente, com intuito de evitar ambiguidade. Como exemplo remover a opção "não sei dizer" da pergunta q*ual é o seu nível de interesse em programar?* Outra mudança que faríamos é acrescentar perguntas tipo Likert sobre PC de forma semelhante ao que fizemos em relação ao PM. Além disso, perguntaríamos aos discentes sobre PA uma vez que investigamos apenas suas concepções sobre PM.

Usamos outros instrumentos que podem contribuir para a discussão de nossa pergunta de pesquisa como o teste de PC (BRENNAN, 2012 *apud* BOUCINHA, 2017) e a atividade de análise de códigos (apêndice D). Contudo, nessa pesquisa não conseguimos contemplar todos os dados, assim esses instrumentos serão explorados em artigos posteriores a essa pesquisa.

Observamos uma escassez de artigos sobre indicadores de PA em algoritmos, em vista disso sugerimos que esse tema seja explorado em pesquisas futuras. Além disso, apontamos também as seguintes temáticas: relação da modularização e depuração com o PA; relação entre metacognição, PM e PC; aproximação entre a teoria de representação semiótica e a programação.

Considerando o contexto escolar atual, transformado pela pandemia do Sars-CoV-2, ressaltamos a importância de se desenvolver habilidades e competências digitais na formação inicial de professores. Reforçarmos também a importância de abordar *softwares* de forma teórica e prática nos cursos de licenciatura em Matemática.

Nesse estudo indicamos algumas relações do PC com o PA, mostrando que é possível abordar esse tema nos cursos de licenciatura em Matemática. Além disso, evidenciamos contribuições que o uso do *Scratch* pode trazer para a formação de professores. Por fim, destacamos também que não é preciso realizar projetos de programação longos e complexos para abordar o PC, uma vez que mesmo projetos curtos como o desenho de um quadrado trabalham com uma variedade de conceitos e práticas desse tipo de pensamento.

REFERÊNCIAS

ACKERMANN, E. **Piaget's Constructivism, Papert's Constructionism: What's the difference?**, v. 5, 2001. Disponível em: https://learning.media.mit.edu/content/publications/EA.Piaget%20_%20Papert.pdf. Acesso em 24 jul 2019.

ALMEIDA, J. R. de; SANTOS, M. C. dos. Pensamento Algébrico: Em busca de uma definição. **Revista Paranaense de Matemática**. Paraná, v. 6, n. 10, p. 34–60, 2017.

ANGELI, C; JAIPAL-JAMANI, K. **Computational Thinking in the STEM Disciplines: Foundations and Research Highlights**. Springer International Publishing, p. 127–150, 2018.

ARRUDA, E. P. Ensino e aprendizagem na sociedade do entretenimento: desafios para a formação docente. **Educação**, v. 36, n. 2, p. 232–239, 2013.

AVILA, C; CAVALHEIRO, S; BORDINI, A; *et al*. Metodologias de Avaliação do Pensamento Computacional: uma revisão sistemática. *In*: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 28, 2017, Recife**. Anais...**[...] Recife: SBIE, 2017. p. 113.

AVILA, C; BORDINI, A; MARQUES, M; *et al*. Desdobramentos do Pensamento Computacional no Brasil. *In*: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 27, 2016, Uberlândia **Anais...**[...]. Uberlandia: SBIE. 2016. p. 200.

BALANSKAT, A; ENGELHARDT, K. **Computing our future Computer programming and coding - Priorities, school curricula and initiatives across Europe**. 2014. Disponível em: https://www.researchgate.net/publication/284139559_Computing_our_future_Computer_programming_and_coding_-_Priorities_school_curricula_and_initiatives_across_Europe. Acesso em: 20 dez. 2019.

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. Disponível em: http://dl.acm.org/citation.cfm?doid=1929887.1929905. Acesso em: 19 fev. 2020.

BARCELOS, T; BORTOLETTO, R; ANDRIOLI, M. Formação online para o desenvolvimento do Pensamento Computacional em professores de Matemática. *In*: CONGRESSO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO**. Workshops...**[...], 5, 2016, Uberlândia: SBC, 2016. p. 1228.

BARCELOS, T. S; SILVEIRA, I. F. Pensamento Computacional e Educação Matemática: Relações para o Ensino de Computação na Educação Básica. *In*: WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO, 2012, Curitiba **Anais...**[...]. Curitiba: CBIE, 2012. p. 23.

BARCELOS, T. S.; MUNOZ, R; VILLARROEL, R; *et al.* Mathematics learning through computational thinking activities: A systematic literature review. **Journal of Universal Computer Science**, v. 24, n. 7, p. 815–845, 2018.

BARCELOS, T. S. *et al.* A Systematic Literature Review on relationships between Computational Thinking and Mathematics. **Journal on Computational Thinking** (JCThink), v. 2, n. 1, p. 23, 2018b.

BARDIN, L. **Análise de Conteúdo**. São Paulo: Edições 70 Ltda/Almedina Brasil. 279p. 2011.

BELINE, W.; COSTA, N. M. L. (Orgs.). **Educação matemática, tecnologia e formação de professores: algumas reflexões.** Campo Mourão: Editora da Fecilcam, 2010.

BENTON, L; HOYLES, C; KALAS, I; *et al.* Building mathematical knowledge with programming: insights from the *Scratch*Maths project. *In*: CONSTRUCTIONISM IN ACTION 2016. **Anais...**[...] Bangkok: 2016, p. 26–33.

BORDINI, A; AVILA, C; MARQUES, M; *et al.* Pensamento Computacional nos Ensinos Fundamental e Médio: uma revisão sistemática. *In*: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 27, 2017, Recife**. Anais...**[...]. Recife: SBIE 2017, p. 123.

BOMBASAR, J; RAABE, A; MIRANDA, E. M de; *et al.* Ferramentas para o Ensino-Aprendizagem do Pensamento Computacional: onde está Alan Turing? *In*: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO,26, 2015, Balneário Camburiú. **Anais...**[...] Balneário Camboriú: SBIE, 2015. p. 81.

BOUCINHA, R.M. **Aprendizagem do pensamento computacional e desenvolvimento do raciocínio**. 2017, Tese (Doutorado no Programa de Pós-Graduação em Informática na Educação). Universidade Federal do Rio Grande do Sul, Porto Alegre, 2017.

BRACKMANN, C; BOUCINHA, R.M; ROMÁN-GONZÁLEZ, M; *et al.* Pensamento Computacional Desplugado: Ensino e Avaliação na Educação Primária Espanhola. *In*: CONGRESSO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 6, 2017, Recife. **Anais...**[...]. Recife: CBIE, 2017. p. 982.

BRANDT, C F; MORETTI, M. T; SCHEIFER, C; *et al.* A importância da função discursiva de designações de relações algébricas para o desenvolvimento do pensamento algébrico. **Educação Matemática Pesquisa: Revista do Programa de Estudos Pós-Graduados em Educação Matemática**, v. 20, n. 1, 2018.

BRENNAN, K; RESNICK, M. New frameworks for studying and assessing the development of computational thinking. *In:* AMERICAN EDUCATIONAL RESEARCH ASSOCIATION MEETING, Vancouver, 2012, Vancouver, **Anais...**[...] Canada, 2012. p. 1–25.

BRENNAN, K; RESNICK, M. Imagining, creating, playing, sharing, reflecting: How online community supports young people as designers of interactive media. *In*: EMERGING TECHNOLOGIES FOR THE CLASSROOM: A LEARNING SCIENCES PERSPECTIVE, 2013, New York **Anais...**[...]. New York: 2013. p. 253–268.

BRENNAN, K. Beyond technocentrism: Supporting constructionism in the classroom. **Constructivist Foundations**, v. 10, n. 3, p. 289–296, 2015.

BRESSAN, M. L. Q; AMARAL, M. A. Avaliando a contribuição do *Scratch* para a aprendizagem pela solução de problemas e o desenvolvimento do pensamento criativo. **Revista Intersaberes**, v. 10, n. 21, p. 509–526, 2015.

BRESSAN, M. L. Q. ***Scratch*! um estudo de caso**. 2016. Dissertação (Mestrado em Tecnologia e Sociedade). Universidade Tecnológica Federal do Paraná. Curitiba, 2016.

BUSTILLO, J; GARAIZAR, P. *Scratch*ing the surface of digital literacy... but we need to go deeper. *In*: FRONTIERS IN EDUCATION CONFERENCE, 2015, El Paso, **Anais...**[...]El Paso: FIE 2015. p. 1–4.

CALDER, N. Using *Scratch*: An integrated problem-solving approach to mathematical thinking. **Australian Primary Mathematics Classroom**, v. 15, n. 4, p. 9–14, 2010.

COLOMBIA, Ministerio de Educación Nacional. **Estándares básicos de competencias en Lenguaje, Matemáticas, Ciencias y Ciudadanas.** Editora: Ministerio de Educación Nacional 2006.

CORRÊA, E. B.; BARBOSA, D. A. O.; TREML, H.; MENDES, L. O. R.; OLIVEIRA, F.; GROSSI, L. Pensamento Computacional na Formação Inicial de Professores de Matemática: uma experiência com *Scratch*. *In*: ENCONTRO PARANAENSE DE TECNOLOGIA NA EDUCAÇÃO MATEMÁTICA,1, 2018, Apucarana. **Anais...**[...] Apucarana: EPTEM, 2018. p. 1-13.

CORRÊA, E. B.; GROSSI, L.; PEREIRA, A. L. Pensamento Computacional na Educação Básica: Um panorama sobre as teses e dissertações produzidas no Brasil. **Revista Tecnologias na Educação**, 2018.

CURCI, A. P. de F.. **O software de programação *Scratch* na formação inicial de professores de matemática por meio da criação de objetos de aprendizagem**.. 2017.Tese (Mestrado em Ensino de Matemática) Universidade Tecnológica Federal do Paraná. Curitiba, 2017.

CURZON, P., PECKHAM, J., TAYLOR, H., SETTLE, A. & ROBERTS, E. 2009. Computational thinking (CT): on weaving it in. **SIGCSE Bull**., 41, 201-202.

DEVLIN, Keith J. **Introduction to mathematical thinking**. Palo Alto: editora Keith Devlin. 2012.

DENNING, P. J. Beyond computational thinking. **Communications of the ACM**, v. 52, n. 6, p. 28-30, 2009.

DENNING. P. J. The profession of it: Beyond computational thinking. **Communications of the ACM**, v. 52, n. 6, 2009.

DO CARMO PAZ, L. A. S. O pensamento computacional e a formação continuada de professores: uma experiência com as TICs. **Revista online de Política e Gestão Educacional**, p. 1655-16677, 2017.

DR. SCRATCH**, Dr. Scratch -** Analyze your Scratch projects here!. Disponível em http://drscratch.org. Acesso em 20 de agosto de 2019.

ELOY, A.; LOPES, R.; ANGELO, I. *Scratch* in Brazil for educational purposes: a systematic review. **Novas Tecnologias na Educação**, v. 15, p. 1–10, 2017.

FIGUEIREDO, J. A.Q. How to improve computational thinking: A case study. **Education in the Knowledge Society**, v. 18, n. 4, p. 35-51, 2017.

FLOS, J. A.; VILAHUR, J. V.. ESeeCode: Creating a computer language from teaching experiences. **Olympiads in Informatics**, v. 10, p. 3–18, 2016.

FRANÇA, R. S de; AMARAL, H. J. C. do. Proposta Metodológica de Ensino e Avaliação para o Desenvolvimento do Pensamento Computacional com o Uso do *Scratch*. *In*: CONGRESSO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 2, 2013, Campinas. **Anais...**[...] Campinas: CBIE, 2013. p. 179–188.

FREIRE, P. **Pedagogia do oprimido**. 50. ed. São Paulo: Paz e Terra, 2011.

FURBER, S. **Shut down or restart? The way forward for computing in UK schools.** Editora Royal Society, Londres 2012.

FUTSCHEK, G. **Algorithmic thinking: The key for understanding computer science.** Springer Verlag, 2006, v. 4226 LNCS, p. 159–168. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics).

GADANIDIS, G; HUGHES, J. M.; MINNITI, L; *et al*. Computational Thinking, Grade 1 Students and the Binomial Theorem. **Digital Experiences in Mathematics Education**, v. 3, n. 2, p. 77–96, 2017.

GATTI, B.A.; NUNES, M.M.R. (Org.). **Formação de professores para o ensino fundamental: estudo de currículos das licenciaturas em Pedagogia, Língua Português, Matemática e Ciências Biológicas.** Textos Fundação Carlos Chagas, São Paulo, v. 29, 160 p., 2009.

GRETTER, S., YADAV, A. Computational thinking and media & information literacy: An integrated approach to teaching twenty-first-century skills. **TechTrends**, p. 1–7., 2016.

GROVER, S., PEA, R. Computational Thinking in K–12: A Review of the State of the Field. **Educational Researcher**, v. 42, n. 1, p 38–43, 2013.

HOYLES, C., NOSS, R.. Revisiting programming to enhance mathematics learning. **In Math + Coding Symposium**. Western University: Londres ,2015.

HEMMENDINGER D. A plea for modesty. **ACM Inroads**, v.1 n. 2, p.4-7, 2010.

KALELIOGLU, F.; GULBAHAR, Y. The Effects of Teaching Programming via *Scratch* on Problem Solving Skills: A Discussion from Learners' Perspective. **Informatics in Education**, v. 13, p. 33–50, 2014.

KALELIOGLU, F.; GÜLBAHAR, Y.; KUKUL, V. A Framework for Computational Thinking Based on a Systematic Research Review. Baltic Journal of Modern Computing, Riga, v. 4, n. 3, p. 583, 2016.

KALELIOGLU, F. Characteristics of Studies Conducted on Computational Thinking: A Content Analysis: Foundations and Research Highlights. *In*: COMPUTATIONAL THINKING IN THE STEM DISCIPLINES: FOUNDATIONS AND RESEARCH HIGHLIGHTS, 2018. **Anais**, 2018, p. 11–29.

KAPUT, J. J. **Teaching and Learning a New Algebra with Understanding.**, 2000. . Disponível em: <https://eric.ed.gov/?id=ED441662>. Acesso em: 26j ul. 2019.

KAPUT, J. J. What is algebra? What is algebraic reasoning? In: KAPUT, J.; CARRAHER, D.; BLANTON, M. (Eds.), **Algebra in the Early Grades. Lawrence Erlbaum Associates.** Nova York, 2008.

KAPUT, J. BLANTON, M. L.; MORENO, Algebra from a symbolization point of view. In: KAPUT, J.; CARRAHER, D.; BLANTON, M. (Eds.), **Algebra in the Early Grades. Lawrence Erlbaum Associates**. New York, 2008.

KAPUT, J J; BLANTON, M L. Characterizing a classroom practice that promotes algebraic reasoning. **Journal for research in mathematics education**, v. 36, n. 5, p. 412, 2005.

KILHAMN, Cecilia; BRÅTING, Kajsa. Algebraic thinking in the shadow of programming. *In*: ongress of the European Society for Research in Mathematics Education, 11, 2019, Suiça. **Anais...**. Suiça, 2019.

KRANTZ, S. G. **Essentials of mathematical thinking.** 1ª Edição. Boca Raton: CRC Press, 2017.

LAI, A F; YANG, S M. The learning effect of visualized programming learning on 6th graders' problem solving and logical reasoning abilities. *In*: INTERNATIONAL CONFERENCE ON ELECTRICAL AND CONTROL ENGINEERING, 2011, Yichangp. - **Anais....** Yichangp, 2011, p 6940–6944.

LAMPERT, M. When the problem is not the question and the solution is not the answer: Mathematical knowing and teaching. **American educational research journal**, v. 27, n. 1, p. 29-63, 1990.

LOGO FOUNDATION. **Logo History**. 2015. Disponível em: <https://el.media.mit.edu/logo-foundation/what_is_logo/history.html>. Acesso em 31 ago. 2019.

LU, J J.; FLETCHER, G H.L. Thinking about computational thinking. **SIGCSE Bulletin Inroads**, v. 41, n. 1, p. 260–264, 2009.

LUMMERTZ, R. dos S. **As potencialidades do uso do software *Scratch* para a construção**

da literacia digital. 2016. Tese (Mestrado em Ensino de Ciências e Matemática) . Universidade Luterana do Brasil, Canoas, 2016.

MACHADO, R. N.; GAUTÉRIO, V. L. B.; PIÑEIRO, M. O.; CRIZEL, R. T. O *Scratch* na sala de aula: o uso da programação com vista à resolução de problemas. **RELACult - Revista Latino-Americana de Estudos em Cultura e Sociedade**, v. 5, n. 4, 5 maio 2019.

MALONEY, J; RUSK, N; BURD,L; *et al. Scratch*: A sneak preview. *In*: PROCEEDINGS - SECOND INTERNATIONAL CONFERENCE ON CREATING, Connecting and Collaborating Through Computing. **Anais...** Kyoto, p. 104–109, 2004.

MALONEY, J.; EASTMOND, E.; SILVERMAN, B.; RUSK, N.; RESNICK, M. The *Scratch* Programming Language and Environment. **ACM Transactions on Computing Education**, v. 10, n. 4, p. 1–15, 2010.

MANDAJI, M *et al.* O PROGRAMAÊ! E a formação de professores para a integração do pensamento computacional ao currículo. **CIET:EnPED**, [S.l.], maio 2018. ISSN 2316-8722.

MANNILA, L; DAGIENE, V ; DEMO, B; *et al.* Computational thinking in K-9 education. *In*: GROUP REPORTS OF THE 2014 INNOVATION AND TECHNOLOGY IN COMPUTER SCIENCE EDUCATION CONFERENCE, 2014. **Anais...**, 2014, p. 1–29.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY. **Scratch** - Imagine, Program, Share. Disponível em: https://scratch.mit.edu. Acesso em: 27 de agosto de 2019.

MASON, John. How Early Is Too Early for Thinking Algebraically? *In*: TEACHING AND LEARNING ALGEBRAIC THINKING WITH 5- TO 12-YEAR-OLDS, 2018.. **Anais...** Springer, Cham, 2018. p. 329–350.

MELLO, T A**. Estratégias de pensamento, atitudes em relação à matemática e desempenho na Prova Brasil**. 2015. Tese (doutorado) - Universidade Estadual de Campinas, Faculdade de Educação, Campinas, 2015.

MESTRE, P; ANDRADE, W; GUERRERO, D; *et al.* Pensamento Computacional: Um estudo empírico sobre as questões de matemática do PISA. *In*: CONGRESSO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 4, 2015, Santa Catarina. **Anais...** Santa Catarina: 2015, v. 1, p. 1281.

MILLER, J; LARKIN, K. **Using Coding to Promote Mathematical Thinking with Year 2 Students: Alignment with the Australian Curriculum**.. Australia, MERGA, 2017.

MONTELEONE, C; WHITE, P; GEIGER, V. Defining the Characteristics of Critical Mathematical Thinking. *In*: ANNUAL MEETING OF THE MATHEMATICS EDUCATION RESEARCH GROUP OF AUSTRALASIA, 2018, Australia. **Anais...** Australia: 2018.

MORENO-LEÓN, J; ROBLES, G. Dr. *Scratch*: A web tool to automatically evaluate *Scratch* projects. *In*: ACM INTERNATIONAL CONFERENCE PROCEEDING SERIES. **Anais...**, v. 09, 2015, p. 132–133.

MORENO-LEÓN, J.; ROBLES, G. Code to learn with *Scratch*? A systematic literature review. *In:* GLOBAL ENGINEERING EDUCATION CONFERENCE (EDUCON), 2016 **Anais**...2016.

MORENO-LEON, J; ROMAN-GONZALEZ, M; ROBLES, G. On computational thinking as a universal skill: A review of the latest research on this ability. *In*: IEEE GLOBAL ENGINEERING EDUCATION CONFERENCE,. **Anais...** 2018, p. 1684–1689.

MUGAYITOGLU, Bekir. **Attitudes of Pre-service Teachers Toward Computational Thinking in Education**. 2016. Tese (Doutorado em Educação). Duquesne University 2016.

NAITO, L; BEZERRA, M; SILVEIRA, I. F. Licenciatura em Computação no Estado de São Paulo: uma Análise Contextualizada e um Estudo de Caso. *In*: WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO, 9, 2011. **Anais...** São Paulo: 2011.

NATIONAL RESEARCH COUNCIL**. Report of a workshop of pedagogical aspects of computational thinking**. Washington, DC: National Academies Press., 2011.

ORO, N; PAZINATO, A; TEIXEIRA, A; *et al.* Olimpíada de Programação de Computadores para Estudantes do Ensino Fundamental: A interdisciplinaridade por meio do Software *Scratch*. *In:* WORKSHOP DE INFORMÁTICA NA ESCOLA, 21, 2015. **Anais...** Sociedade Brasileira de Computação - SBC, v. 1, 2015, p. 102.

PAPERT, S. **Children, computers, and powerful ideas**. Basic Books, Inc., 1980.

PAPERT, S. **Mindstorms: Children, computers, and powerful ideas.** Basic Books, Inc., 1980.

PAPERT, S. **Logo: Computadores e Educação**. Brasiliense, 1985.

PAPERT, S. **An exploration in the space of mathematics educations. International Journal of Computers for Mathematical Learning.,** Kluwer Academic Publishers. v. 1, n. 1, p. 95-123, 1996.

PAPERT, S. **A máquina das crianças**. Porto Alegre: Artemed, 2008.

PINTO, A; ESCUDEIRO, P. The use of *Scratch* for the development of 21st century learning skills in ICT. *In*: IBERIAN CONFERENCE ON INFORMATION SYSTEMS AND TECHNOLOGIES, 2014. **Anais...**. IEEE Computer Society, 2014.

PRADO, M. E. N. B.; DA COSTA, N. M. L. O papel da atividade de programação no processo de construção de conhecimentos para a docência. **Revista e-Curriculum**, v. 14, n. 3, p. 898-918, 2016.

RADER, C; BRAND, C; LEWIS, C. Degrees of comprehension: Children's Understanding of a Visual Programming Environment. *In*: CONFERENCE: HUMAN FACTORS IN COMPUTING SYSTEMS, 1997. **Anais...**. Atlanta: Association for Computing Machinery (ACM), 1997, p. 351–358..

RADFORD, L. Algebraic thinking and the generalization of patterns: a semiotic perspective. *In*: NORTH AMERICA CONFERENCE OF THE INTERNATIONAL GROUP OF

PSYCHOLOGY OF MATHEMATICS EDUCATION – PME, 2006, Bergen. **Anais...**. Bergen University College. v. 1, 2006

RADFORD, L. Grade 2 students' non-symbolic algebraic thinking. In: CAI, J.; KNUTH, E. (Eds). **A global dialogue from multiple perspectives**. Editora Springer. Berlin, 2011.

RADFORD, L. Signs, gestures, meanings: Algebraic thinking from a cultural semiotic perspective. *In*: CONGRESS OF THE EUROPEAN SOCIETY FOR RESEARCH IN MATHEMATICS EDUCATION, 6, 2009, Lyon. **Anais...**Lyon – França, 2009.

RAMOS, J. L.; ESPADEIRO, R. G. Os futuros professores e os professores do futuro. Os desafios da introdução ao pensamento computacional na escola, no currículo e na aprendizagem. **Educação, Formação & Tecnologias - ISSN 1646-933X**, v. 7, n. 2, p. 4–25, 2014.

RESNICK, M. Technologies for Lifelong Kindergarten. **Educational Technology Research and Development**, v. 46, p. 43–55, 1998.

RESNICK, M. All I really need to know (about creative thinking) I learned (by studying how children learn) in kindergarten. *In*: CREATIVITY AND COGNITION 2007 - SEEDING CREATIVITY: TOOLS, MEDIA, AND ENVIRONMENTS, 2007. **Anais...** 2007, p. 1–6.

RESNICK, M.; MALONEY, J.; MONROY-, A.; RUSK, N; *et al. Scratch* Programing For All. **Communications of the ACM**, v. 52, n. 11, p. 60–67, 2009.

RESNICK, M. **Learn to Code , Code to Learn**. 2013 Disponível em: <https://www.edsurge.com/n/2013-05-08-learn-to-code-code-to-learn>.. Acesso em 28 de jul 2019.

RESNICK, M. Give P's a chance: Projects, Peers, Passion, Play. **Constructionism and Creativity conference, opening keynote, Vienna**, p. 13–20, 2014.

REYES-SANTANDER, P.; ACEITUNO, D.; CÁCERES, P. Estilos de pensamiento matemático de estudiantes con talento académico. **Revista de Psicología (PUCP)**, v. 36, n. 1, p. 49-73, 2018.

ROCHA, A. K. D. O. **A programação de computadores como meio para integrar diferentes conhecimentos: uma experiência com professores de matemática.** 2015, Tese (Doutorado em Educação Matemática) Universidade Anhanguera de São Paulo, São Paulo, 2015.

ROCHA, H. R. P. **Programação com o *Scratch* para aprender ciências: uma proposta para a formação de professores da educação básica**. Manaus, AM: Instituto Federal Amazonas, 2017.

ROMÁN-GONZÁLEZ, M; MORENO-LEÓN, J; ROBLES, G. Combining Assessment Tools for a Comprehensive Evaluation of Computational Thinking Interventions. *In*: COMPUTATIONAL THINKING EDUCATION, 2019. **Anais...** Springer Singapore, 2019, p. 79–98.

ROMÁN-GONZÁLEZ, M; PÉREZ-GONZÁLEZ, J.C; JIMÉNEZ-FERNÁNDEZ, C. Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. **Computers in Human Behavior**, v. 72, p. 678-691, 2017.

ROMERO, M; LEPAGE, A; LILLE, B. Computational thinking development through creative programming in higher education. International **Journal of Educational Technology in Higher Education**, v. 14, n. 1, p. 42, 2017.

SELBY, C.; WOOLLARD, J. Refining an understanding of computational thinking. **Technology, Pedagogy and Education. Sage.** 2014.

SELBY, C. C. **How can the teaching of programming be used to enhance computational thinking skills?**2014**,** Tese (Pós-doutrado), University of Southampton, Southampton, UK, 2014.

SHUTE, V J.; SUN, C; ASBELL-CLARKE, J.. Demystifying computational thinking. **Educational Research Review**, v. 22, p. 142-158, 2017.

SOCIEDADE BRASILEIRA DE COMPUTAÇÂO (SBC). **Diretrizes para ensino de Computação na Educação Básica.** 2019. Disponível em: https://www.sbc.org.br/documentos-da-sbc/summary/203-educacao-basica/1220-bncc-em-itinerario-informativo-computacao-2. Acesso em 20 de março de 2020.

SOUSA, M. J.; FINO, C. N. As TIC abrindo caminho a um novo paradigma educacional. **Revista Educação & Cultura Contemporânea**, v. 5 (10), p. 11,26, 2008.

STERNBERG, R. J.; BEN-ZEEV, T. **The nature of mathematical thinking**. Routledge, 1996.

SUTTON, M. J. Problem Representation, Understanding, and Learning Transfer Implications for Technology Education. **Journal of STEM Teacher Education:**, v. 40, n. 4, 1 jun. 2003.

TEIXEIRA, J. **Contribuições para o ensino de programação de computadores a futuros professores de matemática.** 2017. Tese (Doutorado em Ciências da Educação Especialidade em Tecnologia Educativa) - Universidade do Minho. Braga, Portugal, 2017.

TIKHOMIROV, O. K. The psychological consequences of computerization. **The concept of activity in Soviet psychology**, v. 256, p. 278, 1981.

TRIGUEROS, M.; URSINI, S. Understanding of different uses of variable. A study with starting college students. *In*: THE INTERNATIONAL CONFERENCE OF PSYCHOLOGY OF MATHEMATICS EDUCATION, 21, 1997. **Anais...** PME, 1997.

VALENTE, J. A. Informática na Educação: instrucionismo x construcionismo. **Educação Pública**, v. 2, 1, p. 14–26, 1997.

VALENTE, J. A**. Computadores e Conhecimento. Repensado a Educação**. 2. ed. Campinas - SP: Núcleo de Informática Aplicada - NIED, 1998.

VALENTE, J. A. Integração do pensamento computacional no currículo da educação básica: diferentes estratégias usadas e questões de formação de professores e avaliação do aluno**. Revista e-Curriculum**, v. 14, n. 3, p. 864-897, 2016.

VAN DEURSEN, A. J. A. M.; VAN DIJK, J. A. G. M.; VAN LAAR, E.; DE HAAN, J. **The relation between 21st-century skills and digital skills: A systematic literature review.** 2017. Disponível em: <http://dx.doi.org/10.1016/j.chb.2017.03.010>. Acesso em: 26 jul. 2019.

VANHOVE, E 2018. ***Scratch*-LN : flexible editing of the block-based programming language *Scratch***. Tese (Mestrado em Ciência em Engenharia da Computação). Universidade de Gante, Bélgica, 2018.

WILSON, A; MOFFAT, D C. Evaluating *Scratch* to introduce younger schoolchildren to programming. *In*: Annual Workshop of the Psychology of Programming Interest Group, **Anais...**p. 64–75, 2010

WING, J. M. Computational thinking. **Communications of the ACM,** v. 49, n. 3, p. 33-35, mar. 2006.

WING, J. M. Computational thinking and thinking about computing. **Philosophical transactions. Series A, Mathematical, physical, and engineering sciences**, v. 366, p. 3717–3725, 2008.

WING, J. M. Computational Thinking Benefits Society. Social Issues in Computing. **New York: Academic Press,** 2014.

WIRTH, Niklaus. **Algoritmos e estruturas de dados.** 1. ed. Editora LTC, 1986.

YADAV, A. *et al.* Computational thinking in elementary and secondary teacher education. **Transactions on Computing Education**, v. 14, n. 1, p. 5-16, March 2014.

YADAV, A; GOOD, J; VOOGT, J; *et al.* Computational thinking as an emerging competence domain. *In*: Technical and Vocational Education and Training. **Anais...** Springer Nature, v. 23, p. 1051–1067, 2017.

YIN, R. K. **Estudo de Caso: Planejamento e Métodos**. Tradução: Daniel Grassi. 2 ed. Porto Alegre: Bookman, 2001.

ZOPPO, B. M. O uso do *Scratch* no ensino da matemática. *In***:** ENCONTRO BRASILEIRO DE ESTUDANTES DE PÓS-GRADUAÇÃO EM EDUCAÇÃO MATEMÁTICA, 2016, Curitiba. **Anais...** Curitiba,: 2016, p. 11.

APÊNDICE A - TERMO DE CONSENTIMENTO LIVRE ESCLARECIDO.

Programa de Pós-Graduação EC&EM Ensino de Ciências e Educação Matemática

TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO

UEPG UNIVERSIDADE ESTADUAL DE PONTA GROSSA

Título do Projeto: Contribuições do Scratch para a formação inicial de professores de matemática.

Nome dos responsáveis: Luciane Grossi e Emerson Blum Corrêa.

Você está sendo convidado a participar como voluntário de uma pesquisa. Este documento, chamado Termo de Consentimento Livre e Esclarecido, visa assegurar seus direitos como participante. É elaborado em duas vias, uma que deverá ficar com você e outra com o pesquisador, sua via será encaminhada por e-mail.
Por favor, leia com atenção e calma, aproveitando para esclarecer suas dúvidas. Se houver perguntas antes ou mesmo depois de assiná-lo, você poderá esclarecê-las com o pesquisador. Não haverá nenhum tipo de penalização ou prejuízo se você não aceitar participar ou retirar sua autorização em qualquer momento.

Objetivo do estudo: Este trabalho tem por objetivo investigar quais contribuições atividades práticas envolvendo o Scratch propiciam para a formação dos licenciandos em Matemática.
Procedimentos: A participação nesta pesquisa consistirá em responder questionários e realizar atividades em duplas envolvendo o *software Scratch*. A coleta de dados será feita por meio de gravações, registros escritos produzidos pelos participantes durante a pesquisa e questionários.

Observações: * A participação neste projeto está restrita as atividades realizadas entre 01 de agosto de 2019 a 30 de agosto de 2019; * Não haverá nenhuma despesa decorrente da participação nesta pesquisa; * A qualquer momento os participantes poderão retirar seu consentimento e deixar de participar ou, sem precisar justificar, e não sofrerão qualquer prejuízo.

Benefícios - No que diz respeito aos benefícios da pesquisa para o participante são, porém, não limitados a esses: * Curto prazo: Entender um modelo estruturado de pensamento, auxiliando no processo de aprender a aprender; Melhorar o processo de formulação do problema; aperfeiçoar a capacidade de abstração e criatividade; Dar ênfase na criação do conhecimento; Desenvolver um domínio maior sobre Tecnologias Digitais.
* Com o benefício em longo prazo, destacam-se: Aprender a lidar com problemas variados e complexos, de forma crítica; Seguir tendências mundiais de ensino (competências do século XXI); Atender uma forte demanda de mercado; Aumentar a compreensão da fusão da computação com sua área de atuação.

Riscos: No que diz respeito aos riscos da pesquisa para o participante são: * De acordo com a Resolução Nº 196/96 versão 2012, do Conselho Nacional de Saúde, e da estrutura desenvolvida e apresentada neste projeto, pode-se inferir que os riscos de ordem física, psíquica, moral, intelectual, social, cultural ou espiritual são de baixa representatividade, como por exemplo: um possível desagrado do participante com seu desempenho nas atividades propostas. * Para evitar comparação de desempenho entre participantes, os dados individuais não serão divulgados. Ao término da pesquisa os participantes receberão um resumo dos resultados obtidos com a pesquisa.

Sigilo e privacidade: Você tem a garantia de que sua identidade será mantida em sigilo e nenhuma informação será dada a outras pessoas que não façam parte da equipe de pesquisadores. Na divulgação dos resultados desse estudo, seu nome não será citado sob nenhuma circunstância.
Os dados coletados serão utilizados, única e exclusivamente, para fins desta pesquisa, e os resultados poderão ser publicados.
Contato: Qualquer dúvida, pedimos a gentileza de entrar em contato pelo e-mail: emerblum@gmail.com.

_____
Emerson Blum Corrêa – Responsável por obter o consentimento livre e esclarecido

Após ter recebido esclarecimentos sobre os objetivos, métodos, benefícios previstos e sigilo das informações declaro que aceito participar deste estudo.

_____
Assinatura do Participante

Página **1** de **1**

APÊNDICE B – QUESTIONÁRIO PRÉ-IMPLEMENTAÇÃO.

Questionário inicial

TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO

Você está sendo convidado a participar como voluntário de uma pesquisa. O Termo de Consentimento Livre e Esclarecido visa assegurar seus direitos como participante.

Este formulário tem a intenção de coletar dados sobre o perfil dos acadêmicos do quarto ano do curso de licenciatura em Matemática sobre conhecimentos sobre computação e suas intersecções com a matemática.

Esta pesquisa está sendo conduzida no escopo de uma pesquisa no Programa de Pós-Graduação em Ensino de Ciências e Educação Matemática (PPGECEM) da UEPG, com Emerson Blum Corrêa como pesquisador responsável sob a orientação da profª Dra. Luciane Grossi. Para esclarecimentos de dúvidas ou obtenção de informações adicionais favor entrar em contato pelo e-mail: <emerblum@gmail.com>. O tempo estimado para responder ao questionário é de 70 minutos.

Os procedimentos da Pesquisa obedecem aos Critérios de Ética em Pesquisa com Seres Humanos conforme Resolução nº 466/12 do Conselho Nacional de Saúde, de forma que nenhum dos procedimentos usados oferece risco à dignidade do participante. As informações e os materiais de registro coletados durante a Pesquisa serão utilizados apenas para fins de investigação e de produção de conhecimento. Todo o material obtido será arquivado por um período de 05 anos a contar do término desta pesquisa.

CONDIÇÕES E ESTIPULAÇÕES

1. Eu entendo que todas as informações são confidenciais. Concordo em concluir o questionário para fins de pesquisa e que os dados derivados dessa pesquisa podem ser publicados de forma anônima em periódicos, eventos e publicações em blogs.

2. Entendo que minha participação nesse estudo é totalmente voluntária e que recusar a participar não envolverá penalidade ou perde de benefícios. Se eu escolher, posso retirar minha participação a qualquer momento. Eu também entendo que, se optar por participar, posso me recursar a responder qualquer pergunta que me deixe desconfortável.

3. Entendo que posso entrar em contato com o pesquisador se tiver alguma dúvida sobre a pesquisa. Estou ciente de que meu consentimento não me beneficiará diretamente. Estou ciente também de que o autor manterá os dados coletados em perpetuidade e poderá utilizá-los para trabalhos acadêmicos futuros.

4. Ao clicar no botão "Próxima" abaixo, eu livremente forneço consentimento e reconheço meus direitos como participante voluntário da pesquisa, conforme descrito acima, e forneço consentimento ao pesquisador para usar as informações fornecidas na condução de pesquisas sobre as áreas mencionadas acima.

1. Para você, a matemática e a computação estão relacionadas? Como?

2. O que você entende por programação?

3. Qual é o seu nível de interesse em programar? (Marque apenas uma opção)

( ) Muito interesse ( ) Algum interesse ( ) Não sei dizer ( ) Desinteressado

( ) Muito desinteressado

4. Como você avalia seus conhecimentos sobre programação? (Marcar apenas uma oval)

( ) Avançado ( ) Intermediário ( ) Básico ( ) Não tenho nenhuma noção sobre programação

5. Você já programou alguma vez? ( ) Sim ( ) Não

6. Você já programou em alguma linguagem? (Java, HTML, DQL, C++, Fortran, Assembly, Phyton, entre outras) ou ambientes de programação (Scilab, *Scratch*, Mathlab, Maple, Mathematica, Appinventor, Godot, Unity, entre outros)? Qual?

7. Em que contexto (curso livre, curso técnico, disciplina da licenciatura, programei por conta própria, entre outros)?

8. Você gosta de programar?

( ) Gosto muito ( ) Gosto um pouco ( ) Não gosto nem desgosto ( ) Desgosto um pouco ( ) Desgosto muito

9. Você vê possibilidades para se trabalhar matemática por meio da programação? Como?

10. Você conhece algum dos termos abaixo?

	Sim	**Não**
Ciclos (loops)	( )	( )
Sequência/algoritmo	( )	( )
Execução em paralelo	( )	( )
Evento	( )	( )
Condições lógicas	( )	( )
Operadores	( )	( )
Dados	( )	( )
Modularização	( )	( )
Processos iterativos	( )	( )
Depuração	( )	( )
Computar	( )	( )
Variáveis	( )	( )
Sub-rottina	( )	( )

Dos termos em que você assinalou a opção “Sim” na questão anterior, qual é a compreensão sobre ele(s)? Para que ele(s) serve(m)?

11. O desafio de resolver problemas usando conhecimentos da ciência da computação me estimula

	Discordo totalmente	Discordo parcialmente	Neutro	Concordo parcialmente	Concordo totalmente
Concordância	( )	( )	( )	( )	( )

12. Acho ciência da computação interessante.

	Discordo totalmente	Discordo parcialmente	Neutro	Concordo parcialmente	Concordo totalmente
Concordância	( )	( )	( )	( )	( )

13. Tenho competência para resolver problemas cotidianos usando computadores.

	Discordo totalmente	Discordo parcialmente	Neutro	Concordo parcialmente	Concordo totalmente
Concordância	( )	( )	( )	( )	( )

14. Tenho facilidade para aprender a trabalhar com um novo software.

	Discordo totalmente	Discordo parcialmente	Neutro	Concordo parcialmente	Concordo totalmente
Concordância	( )	( )	( )	( )	( )

15. O que você entende por Pensamento Matemático?

16. O que você entende por Pensamento Computacional?

17. Você consegue estabelecer/ver alguma relação entre estes dois tipos de pensamentos? Quais?

18. Pensar matematicamente é lembrar e aplicar a regra correta na ordem correta.

	Discordo totalmente	Discordo parcialmente	Neutro	Concordo parcialmente	Concordo totalmente
Concordância	( )	( )	( )	( )	( )

19. Pensamento Matemático não está relacionado a intuição ou imaginação.

	Discordo totalmente	Discordo parcialmente	Neutro	Concordo parcialmente	Concordo totalmente
Concordância	( )	( )	( )	( )	( )

20. Pensamento Matemático é um tipo de pensamento fechado, exato e livre de incertezas.

	Discordo totalmente	Discordo parcialmente	Neutro	Concordo parcialmente	Concordo totalmente
Concordância	( )	( )	( )	( )	( )

21. Pensamento Matemático envolve a habilidade de pensar de forma flexível.

	Discordo totalmente	Discordo parcialmente	Neutro	Concordo parcialmente	Concordo totalmente
Concordância	( )	( )	( )	( )	( )

22. Pensamento Matemático se limita a procedimentos e manipulações simbólicas.

	Discordo totalmente	Discordo parcialmente	Neutro	Concordo parcialmente	Concordo totalmente
Concordância	( )	( )	( )	( )	( )

23. Pensamento Matemático envolve abstração e linguagem.

	Discordo totalmente	Discordo parcialmente	Neutro	Concordo parcialmente	Concordo totalmente
Concordância	( )	( )	( )	( )	( )

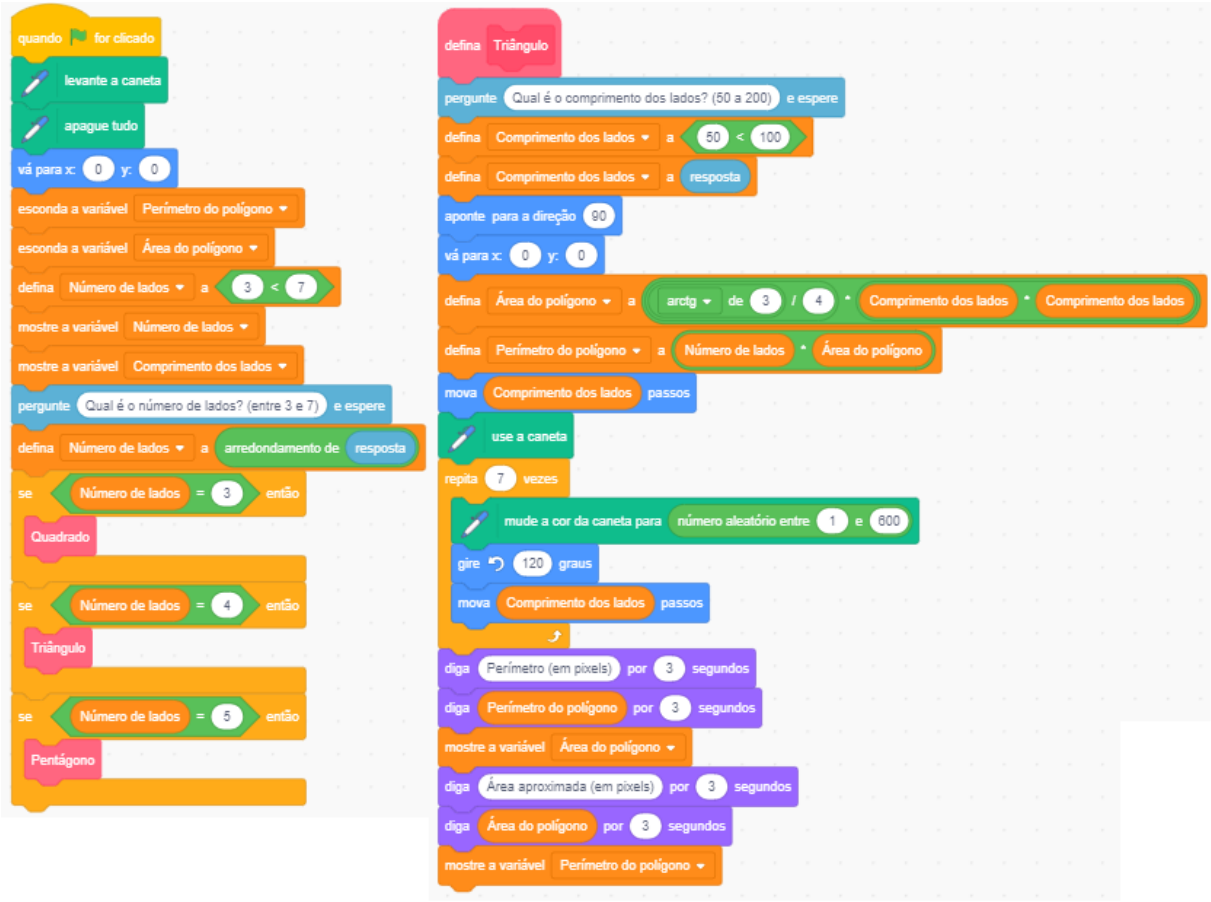
24. Pensamento Matemático envolve utilizar conhecimentos e práticas da matemática para resolver problemas em diversos contextos.

	Discordo totalmente	Discordo parcialmente	Neutro	Concordo parcialmente	Concordo totalmente
Concordância	( )	( )	( )	( )	( )

25. Pensamento Matemático está relacionado a imaginação e a criatividade.

	Discordo totalmente	Discordo parcialmente	Neutro	Concordo parcialmente	Concordo totalmente
Concordância	( )	( )	( )	( )	( )

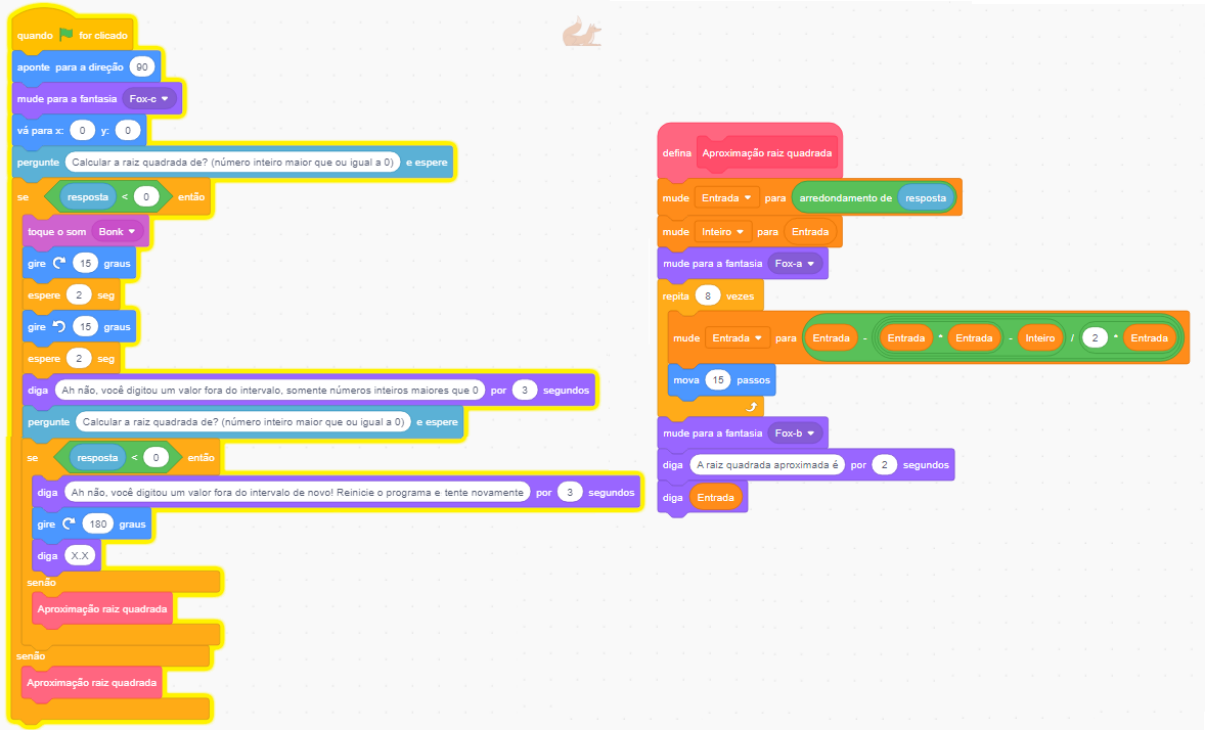
APÊNDICE C - CÓDIGO DEPURADO NA AULA 5.




defina Quadrado
pergunte Qual é o comprimento dos lados? (50 a 150) e espere
defina Comprimento dos lados a 50 < 150
defina Comprimento dos lados a resposta
aponte para a direção 90
vá para x: 0 y: 0
defina Área do polígono a Comprimento dos lados * Número de lados
defina Perímetro do polígono a Comprimento dos lados * Comprimento dos lados
use a caneta
mova Comprimento dos lados passos
repita 3 vezes
mude a cor da caneta para número aleatório entre 1 e 600
gire 180 graus
mova 10 passos
diga Perímetro (em pixels) por 3 segundos
diga Perímetro do polígono por 3 segundos
mostre a variável Perímetro do polígono
diga Área aproximada (em pixels) por 3 segundos
diga Área do polígono por 3 segundos
mostre a variável Área do polígono


defina Pentágono
pergunte Qual é o comprimento dos lados? (50 a 150) e espere
defina Comprimento dos lados a 50 < 100
defina Comprimento dos lados a resposta
aponte para a direção 90
vá para x: 0 y: 0
defina Área do polígono a 0
defina Perímetro do polígono a Número de lados * Comprimento dos lados
use a caneta
mova Comprimento dos lados passos
repita 4 vezes
mude a cor da caneta para número aleatório entre 1 e 600
gire 90 graus
mova Comprimento dos lados passos
diga Perímetro (em pixels) por 3 segundos
diga Perímetro do polígono por 3 segundos
mostre a variável Perímetro do polígono
diga Área aproximada (em pixels) por 3 segundos
diga Área do polígono por 3 segundos
mostre a variável Área do polígono

APÊNDICE D - ATIVIDADE DE ANÁLISE REALIZADA NA AULA 07.



Análise de códigos - Questões abertas

Prezado participante,

Eu enviei em seu e-mail pessoal o script (arquivo com código) do programa analisado nessa atividade. Sinta-se a vontade para explorá-lo na interface do Scratch para ajudá-lo a responder as questões abaixo.

Caso não tenha recebido o script peço que entre em contato comigo.

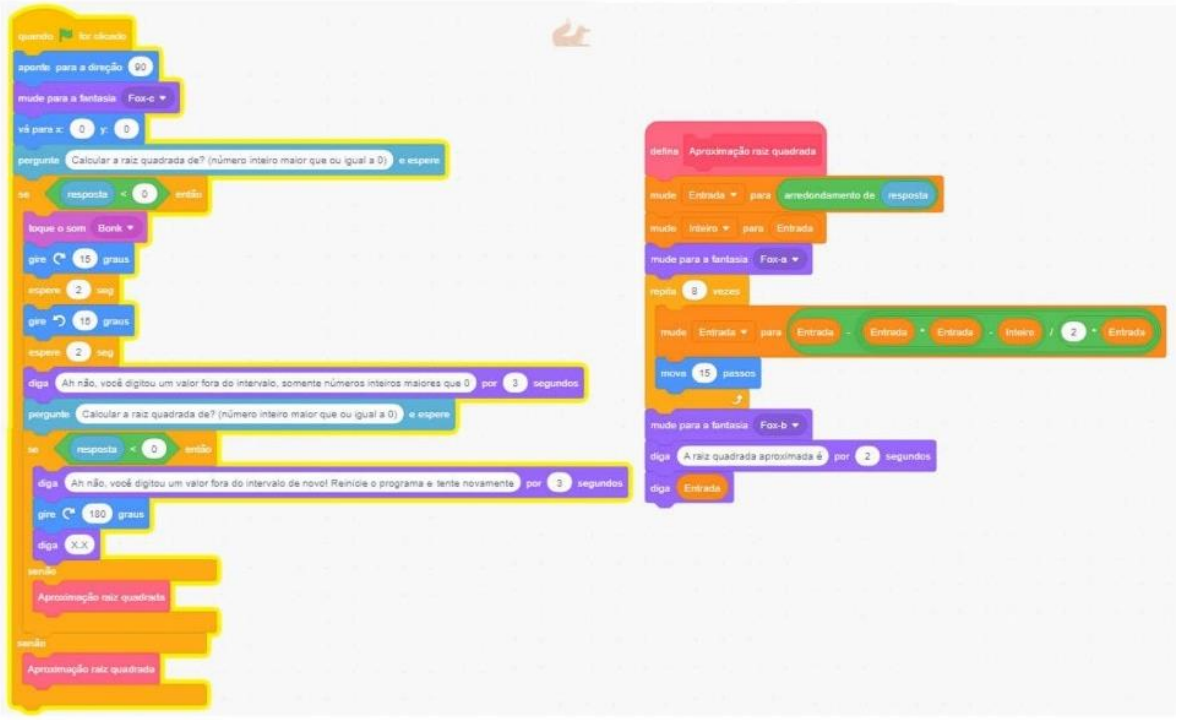
Peço gentilmente que faça a atividade individualmente, ou seja, sem discutir com os colegas.

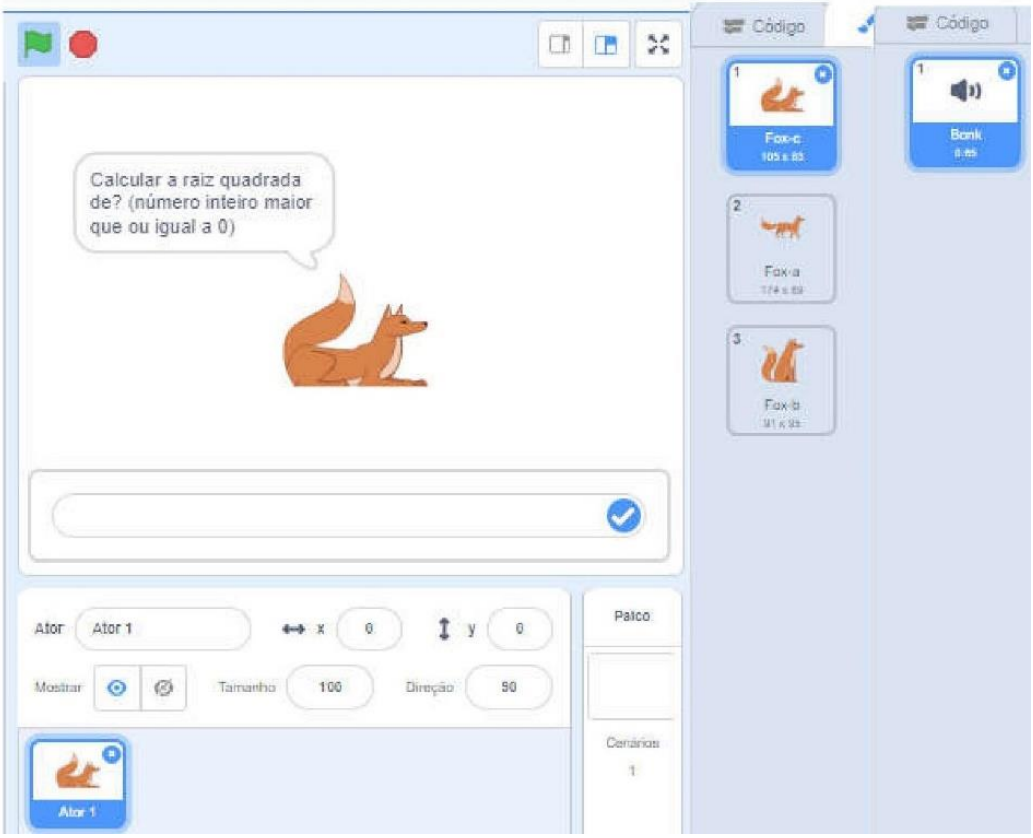
*Obrigatório

1. **R. A.** *

Atividade de análise 1 - Aspectos computacionais

Responda as questões com base no código abaixo.





2. Qual/quais são os comandos que impede(m) que o programa processe um valor fora do intervalo determinado?

3. Qual comando garante que o valor inicial da variável ‘Entrada’ sempre será um número inteiro?

4. Como os comandos de restrição da variável ‘Entrada’ foram construídos?

5. Por que os comandos ‘pergunte”, presentes no bloco de códigos encaixados na bandeira verde, são importantes para a execução do programa?

6. Qual é a diferença entre o comando ‘diga’ e ‘pergunte’?

7. O que aconteceria se o comando ‘diga x e espere y segundos’ fosse alterado para ‘diga x’ (apenas diga, sem o tempo de espera)?

8. Quais são os efeitos das condicionais empregadas no bloco de códigos encaixados na bandeira verde?

9. Quais comandos alteram o valor da variável ‘Entrada’?

10. Qual comando determina o valor da variável ‘Inteiro’?

11. Qual é o efeito do comando ‘repita’, presente no módulo ‘Aproximação raiz quarada’ para a execução do código?

12. Por que os comandos ‘mude Inteiro para entrada’ e ‘mude entrada para (aproximação da reposta)’ estão antes do comando ‘repita’?

13. O que aconteceria se esses comandos estivessem dentro do comando ‘repita’?

14. Como a fórmula matemática usada para a aproximação da raiz foi construída?

15. Quantos passos a raposa anda antes de dizer o resultado?

16. Por que ao invés da raposa dizer ‘resultado’ ela está programa para dizer ‘entrada’? (final do módulo ‘aproximação raiz quadrada’)

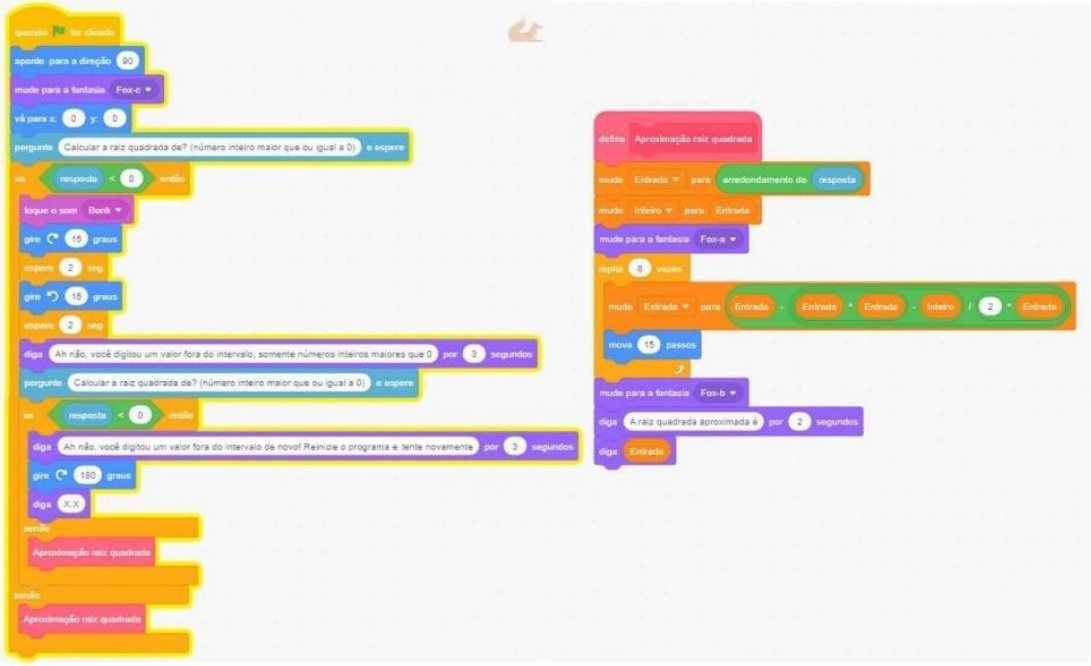
17. A variável ‘Entrada’ é alterada para o ‘arredondamento da resposta’ antes de iniciar o comando ‘repita’. Por que o valor dito pela raposa no final é diferente do valor inserido no começo?

18. Os três comandos ‘mude a fantasia’ geram que efeito na execução do código?

19. O que aconteceria se ordem dos comandos ‘mude a fantasia’ fosse alterada?

Atividade de análise 2 – Aspectos matemáticos.

Responda as questões com base no código abaixo.



20. Por que a variável ‘Entrada’ não pode ser menor que 0?

21. A variável ‘Entrada’ pode ser um número real positivo? Caso sim, qual mudança deve ser feita no código para permitir a entrada desse tipo de número?

22. Qual é o nome do método utilizado para se aproximar a raiz quadrada?

23. Por que a variável ‘Inteiro’ é alterada para ‘Entrada’ antes do comando ‘repita’?

24. Por que o programa executa todo o método mais de uma vez?

25. Qual é a importância do número de repetições do método para a aproximação?

26. Qual é o temo utilizado para processos onde se repete um mesmo procedimento múltiplas vezes?

27. O que acontece com a variável ‘Entada’ ao executar o comando ‘repita’?

APÊNDICE E – ALGORITMOS EM LINGUAGEM MATERNA.

Algoritmos sujeito I1

ROTEIRO DE AULA – ATIVIDADE 1| PARTE 1

Tarefa 1: Qual é a definição de quadrado?

Tarefa 2: Elabore um algoritmo que descreva a construção de um quadrado.

Tarefa 1: Quadrado é um polígono de quatro lados iguais, paralelos dois a dois, com quatro ângulos retos.

Tarefa 2: ① Definir um tamanho para o lado do quadrado.

② Desenhar um segmento de tamanho qualquer, com tamanho maior que o definido no passo 1.

③ Abrir o compasso com o tamanho definido no passo 1.

④ Colocar a ponta seca do compasso na extremidade esquerda do segmento feito no passo 2 e fazer um arco interceptando a reta, formando assim o segmento $\overline{AB}$.

⑤ Com o auxílio de esquadros, traçar uma perpendicular ao ponto A e ao ponto B com tamanho maior que o definido no passo 2.

⑥ Compasso com a mesma abertura, ponta seca no ponto A, fazer um arco interceptando a perpendicular ao segmento $\overline{AB}$ pelo ponto A, formando o ponto C.

⑦ Compasso com a mesma abertura, ponta seca no ponto B, fazer um arco interceptando a perpendicular ao segmento $\overline{AB}$ pelo ponto B, formando o ponto D.

⑧ Ligar os pontos C e D.

ROTEIRO DE AULA – ATIVIDADE 1| PARTE 1

Tarefa 3: Qual é a definição de triângulo escaleno?

Tarefa 4: Elabore um algoritmo que descreva a construção de um triângulo escaleno com comprimentos variáveis em seus segmentos.

- Tarefa 3: Triângulo escaleno é um polígono de três lados com medidas diferentes.

- Tarefa 4: ① Definir três medidas diferentes para os lados do triângulo, de maneira que nenhuma medida seja maior à soma das outras duas.

② Desenhar um segmento de tamanho qualquer, maior que a maior medida definida no passo 1.

③ Abrir o compasso com o tamanho da maior medida definida no passo 1.

④ Ponta seca na extremidade esquerda do segmento feito no passo 2, traçar um arco interceptando a reta, formando o segmento $\overline{AB}$.

⑤ Abrir o compasso com o segundo maior tamanho definido no passo 1.

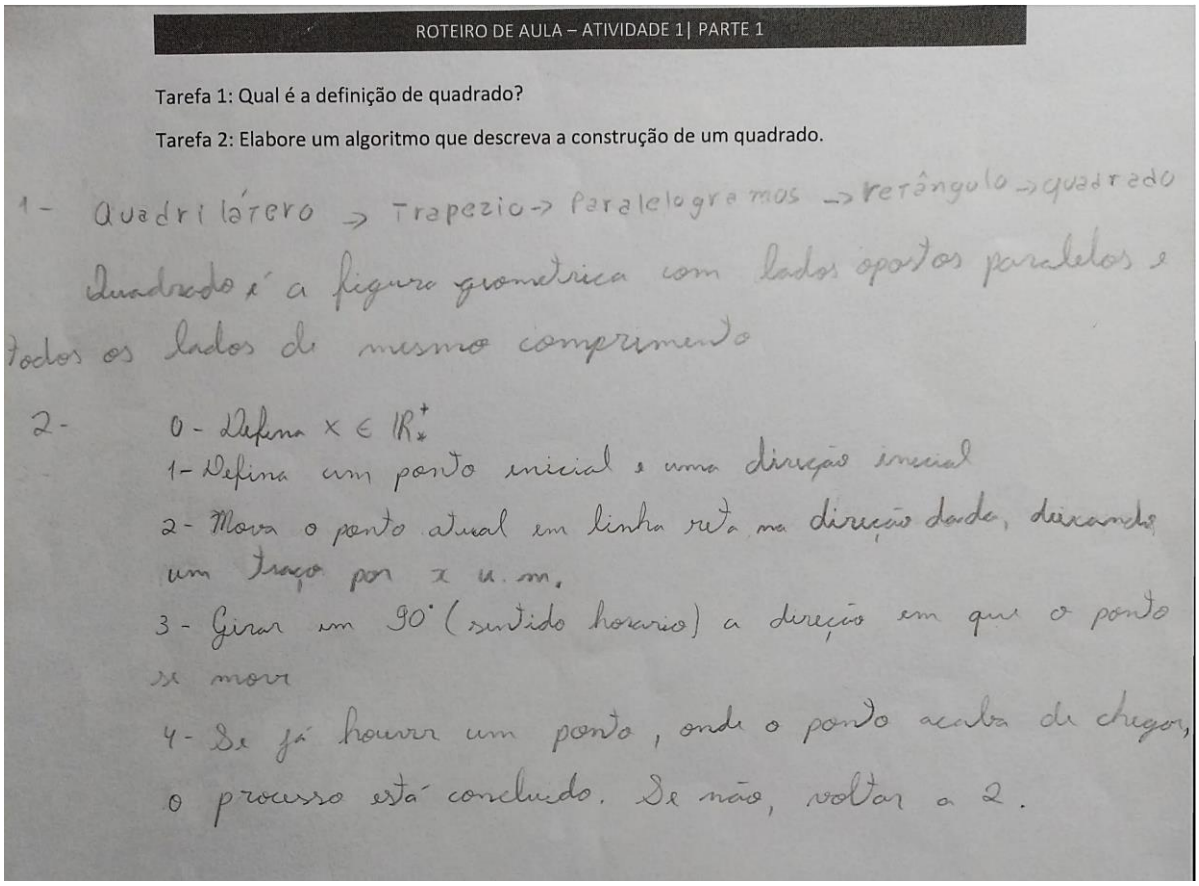
⑥ Ponta seca em A, fazer um arco grande acima do segmento $\overline{AB}$.

⑦ Abrir o compasso com o menor tamanho definido no passo 1.

⑧ Ponta seca em B, fazer um arco acima do segmento $\overline{AB}$, interceptando o arco feito no passo 6, formando o ponto C.

⑨ Ligar os pontos A e C e os pontos B e C.

Algoritmos sujeito I2



ROTEIRO DE AULA – ATIVIDADE 1| PARTE 1

Tarefa 1: Qual é a definição de quadrado?

Tarefa 2: Elabore um algoritmo que descreva a construção de um quadrado.

1- Quadrilátero → Trapezio → Paralelogramos → Retângulo → quadrado

Quadrado é a figura geometrica com lados opostos paralelos e todos os lados de mesmo comprimento

2-

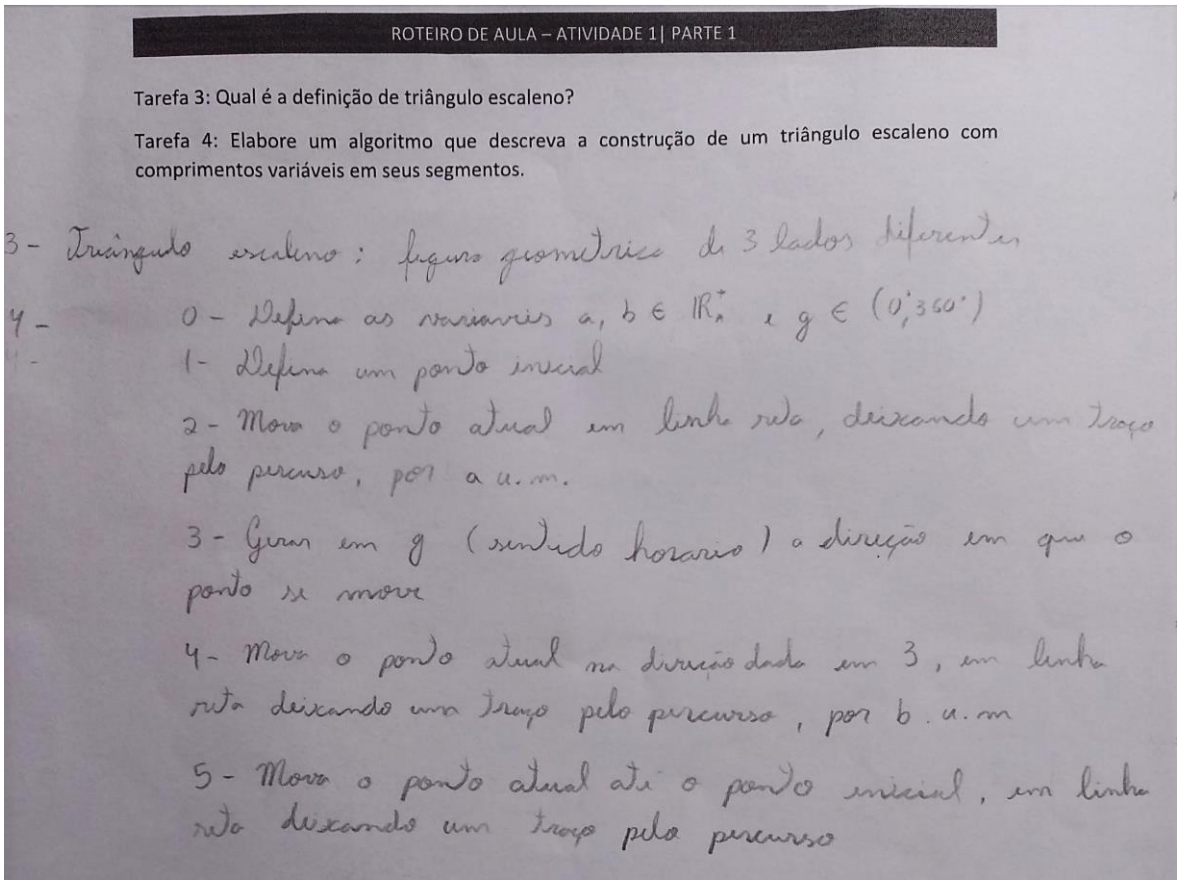
0 - Defina $x \in \mathbb{R}_*^+$

1- Defina um ponto inicial e uma direção inicial

2- Mova o ponto atual em linha reta na direção dada, deixando um traço por x u.m.

3 - Girar em 90° (sentido horario) a direção em que o ponto se move

4- Se já houver um ponto, onde o ponto acaba de chegar, o processo está concluido. Se não, voltar a 2.



ROTEIRO DE AULA – ATIVIDADE 1| PARTE 1

Tarefa 3: Qual é a definição de triângulo escaleno?

Tarefa 4: Elabore um algoritmo que descreva a construção de um triângulo escaleno com comprimentos variáveis em seus segmentos.

3 - Triângulo escaleno: figura geométrica de 3 lados diferentes

4 - 0 - Defina as variáveis $a, b \in \mathbb{R}_+^*$ e $g \in (0;360°)$

1 - Defina um ponto inicial

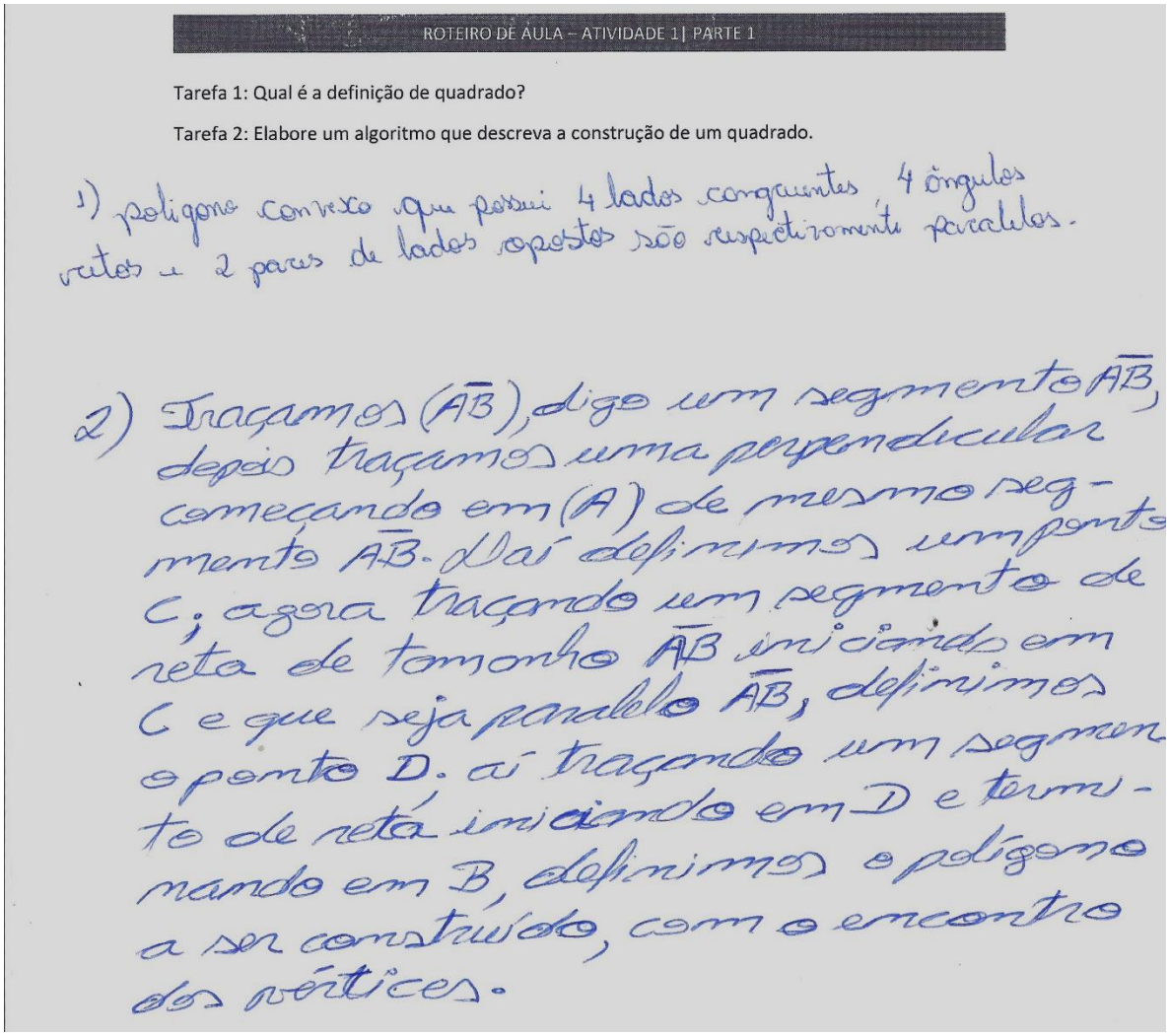
2 - Mova o ponto atual em linha reta, deixando um traço pelo percurso, por a u.m.

3 - Gire em g (sentido horário) a direção em que o ponto se move

4 - Mova o ponto atual na direção dada em 3, em linha reta deixando um traço pelo percurso, por b u.m

5 - Mova o ponto atual até o ponto inicial, em linha reta deixando um traço pelo percurso

Algoritmos sujeito N1



ROTEIRO DE AULA – ATIVIDADE 1| PARTE 1

Tarefa 1: Qual é a definição de quadrado?

Tarefa 2: Elabore um algoritmo que descreva a construção de um quadrado.

1) polígono convexo que possui 4 lados congruentes, 4 ângulos retos e 2 pares de lados opostos são respectivamente paralelos.

2) Traçamos ($\overline{AB}$), digo um segmento $\overline{AB}$, depois traçamos uma perpendicular começando em (A) de mesmo segmento $\overline{AB}$. Daí definimos um ponto C; agora traçando um segmento de reta de tamanho $\overline{AB}$ iniciando em C e que seja paralelo $\overline{AB}$, definimos o ponto D. aí traçando um segmento de retá iniciando em D e terminando em B, definimos o polígono a ser construído, com o encontro dos vértices.

ROTEIRO DE AULA – ATIVIDADE 1| PARTE 1

Tarefa 3: Qual é a definição de triângulo escaleno?

Tarefa 4: Elabore um algoritmo que descreva a construção de um triângulo escaleno com comprimentos variáveis em seus segmentos.

3) Polígono que possui 3 lados, todos com medidas diferentes, logo não são polígonos regulares e não possuem eixo de simetria, como um qualquer outro triângulo seus ângulos internos também medem 180°, porém todos os ângulos diferentes entre si.

4) Traçamos dois pontos quaisquer, seja A e B, ligando os pontos tem-se $\overline{AB}$, daí marcando um ponto aleatório no espaço, que não esteja no ponto médio AB, marcamos o ponto C. Agora ligando os pontos A, B e C, tem-se o triângulo escaleno de comprimentos variáveis.

Algoritmo sujeito N2

ROTEIRO DE AULA – ATIVIDADE 1| PARTE 1

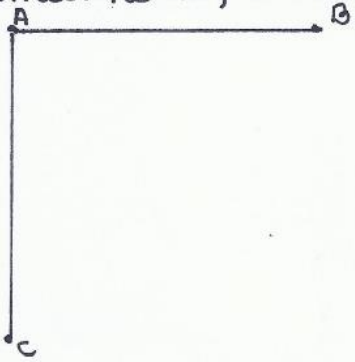
Tarefa 1: Qual é a definição de quadrado?

Tarefa 2: Elabore um algoritmo que descreva a construção de um quadrado.

① Quadrado é um poligono convexo que possui 4 lados congruentes, 4 ângulos retos e 2 pares de lados opostos são respectivamente paralelos.

② * Traça-se um segmento AB

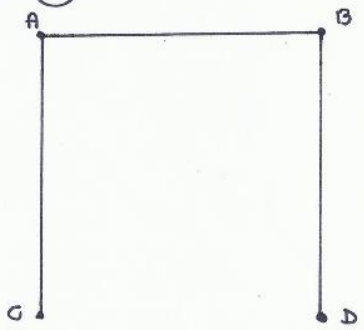
* No ponto A, constroi-se uma perpendicular do tamanho AB, encontrando o ponto C.



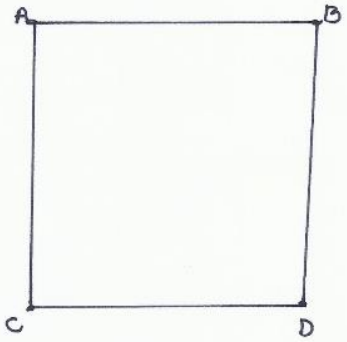
* A partir do ponto B, traça-se uma paralela ao segmento AC, do tamanho de AB, encontrando o ponto D.

(desenho na próxima folha).

Cont. (2)



* Traçar uma paralela de AB iniciando em C e terminando em D.



ROTEIRO DE AULA – ATIVIDADE 1| PARTE 1

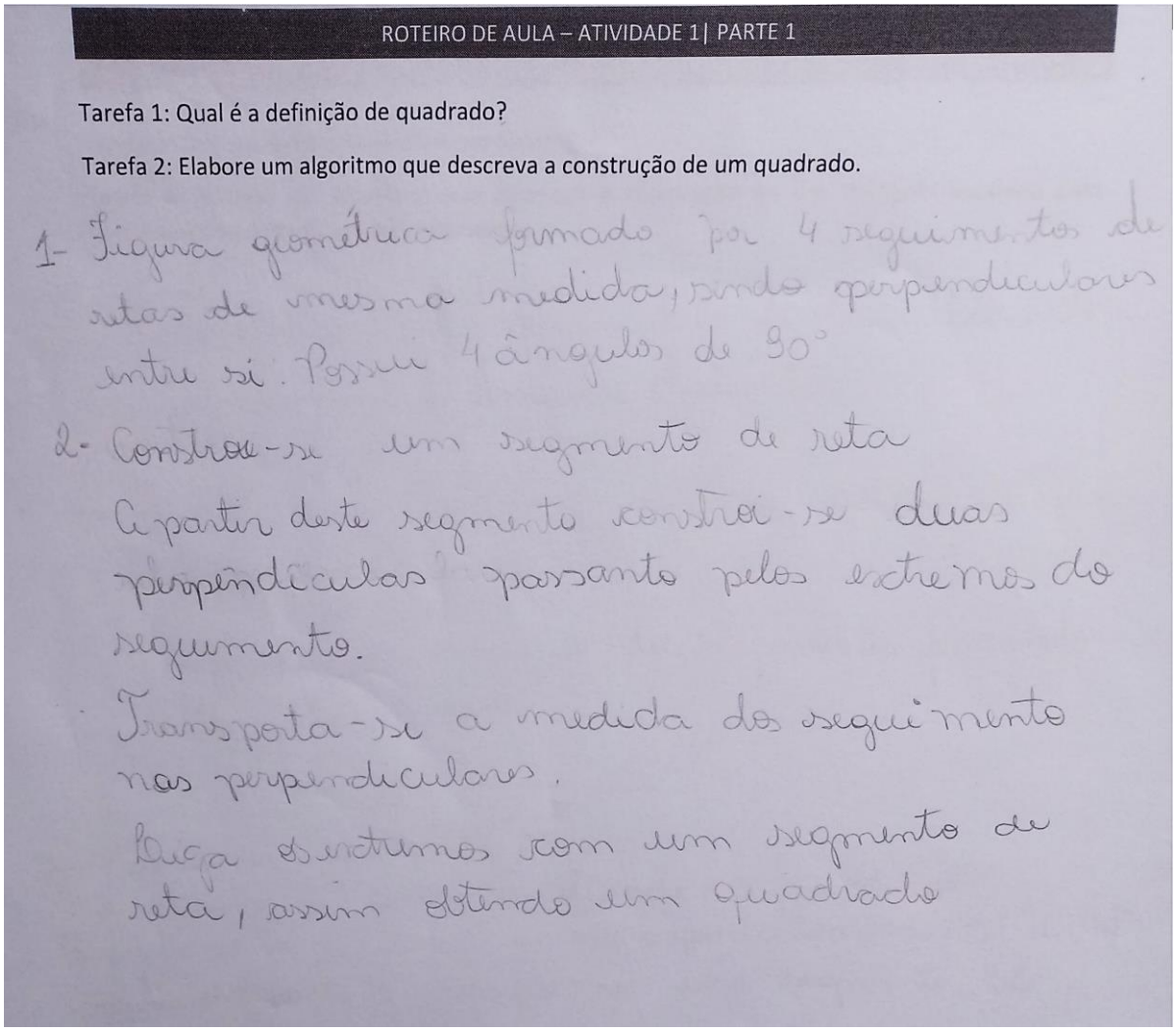
Tarefa 3: Qual é a definição de triângulo escaleno?

Tarefa 4: Elabore um algoritmo que descreva a construção de um triângulo escaleno com comprimentos variáveis em seus segmentos.

③ Triângulo escaleno é polígono que possui três lados, todos com medidas diferentes. Sendo assim, os triângulos escalenos não são polígonos regulares e nem possuem eixo de simetria.

④ Traça-se uma reta de medida qualquer determinada pelos pontos AB, coloca-se um ponto C não pertencente a mediatriz e nem no segmento AB, após traça-se um segmento ligando BC e outro ligado AC.

Algoritmos sujeito N3



ROTEIRO DE AULA – ATIVIDADE 1| PARTE 1

Tarefa 1: Qual é a definição de quadrado?

Tarefa 2: Elabore um algoritmo que descreva a construção de um quadrado.

1- Figura geométrica formado por 4 seguimentos de retas de mesma medida, sendo perpendiculares entre si. Possui 4 ângulos de 90°

2- Constroe-se um segmento de reta
A partir deste segmento constroi-se duas perpendiculas passando pelos extremos do segumento.
Transporta-se a medida do seguimento nas perpendiculares.
Liga os extremos com um segmento de reta, assim obtendo um quadrado

ROTEIRO DE AULA – ATIVIDADE 1| PARTE 1

Tarefa 3: Qual é a definição de triângulo escaleno?

Tarefa 4: Elabore um algoritmo que descreva a construção de um triângulo escaleno com comprimentos variáveis em seus segmentos.

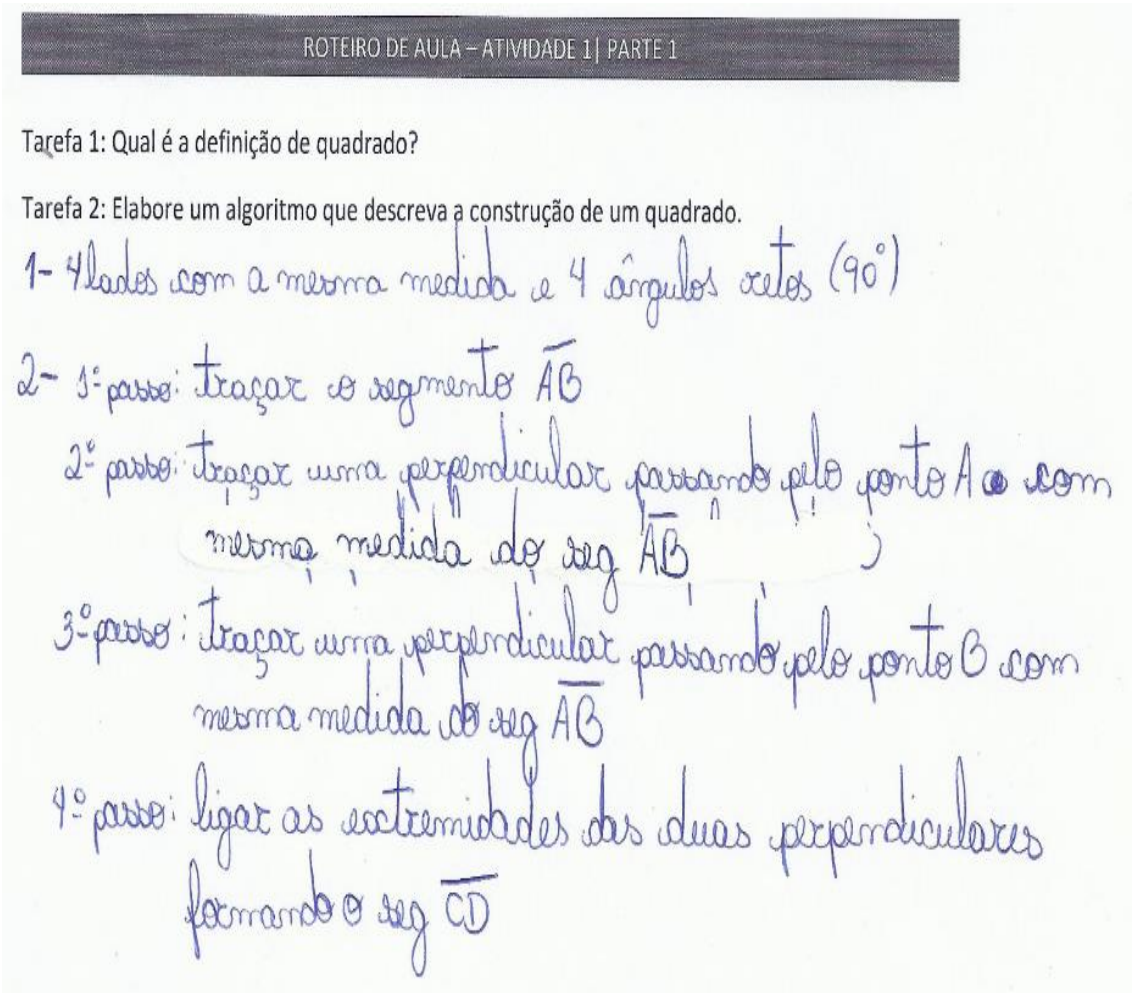
3- Triângulo que possui três lados diferentes e também três ângulos distintos.

4-

Liga se os extremos dos segmentos formando um triângulo escaleno

Constroi-se um segmento AB
Determina se o ponto médio M de AB
A partir de M traça-se uma perpendicular a AB (reta r)
A partir do ponto B constroi-se um segmento BC, sendo que C não pode pertencer a r.
Liga o segmento AC.

Algoritmos sujeito N4



ROTEIRO DE AULA – ATIVIDADE 1| PARTE 1

Tarefa 1: Qual é a definição de quadrado?

Tarefa 2: Elabore um algoritmo que descreva a construção de um quadrado.

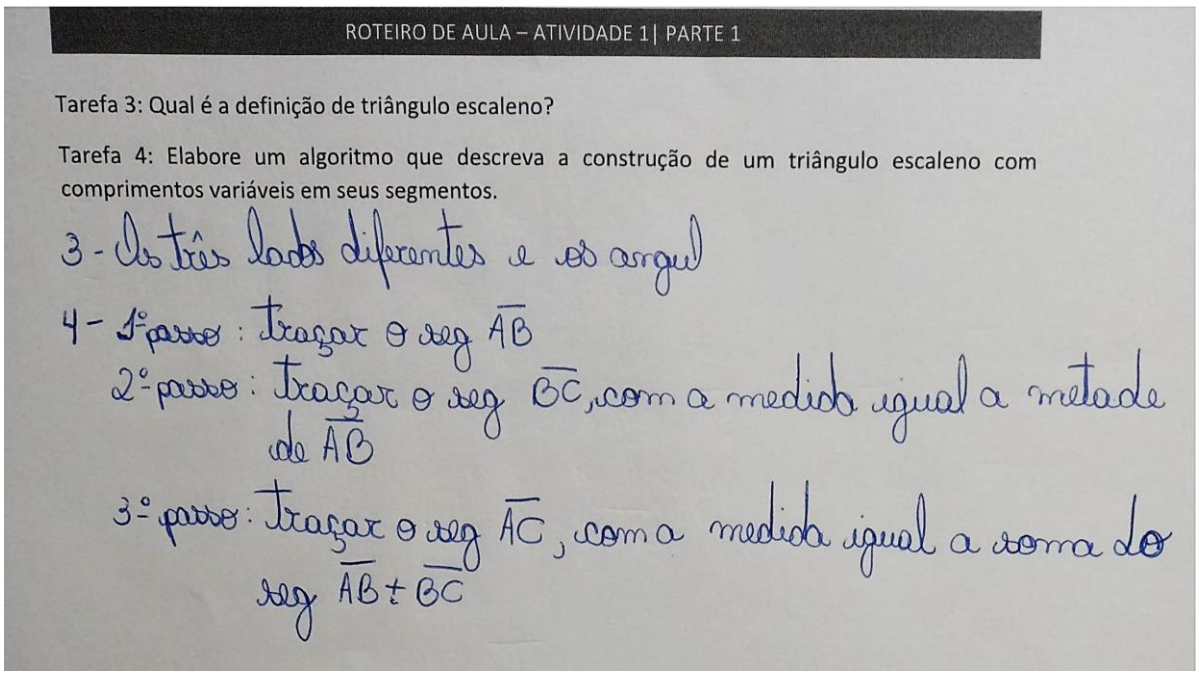
1- 4 lados com a mesma medida e 4 ângulos retos (90°)

2- 1° passo: traçar o segmento $\overline{AB}$

2° passo: traçar uma perpendicular passando pelo ponto A com mesma medida do seg $\overline{AB}$

3° passo: traçar uma perpendicular passando pelo ponto B com mesma medida do seg $\overline{AB}$

4° passo: ligar as extremidades das duas perpendiculares formando o seg $\overline{CD}$



ROTEIRO DE AULA – ATIVIDADE 1| PARTE 1

Tarefa 3: Qual é a definição de triângulo escaleno?

Tarefa 4: Elabore um algoritmo que descreva a construção de um triângulo escaleno com comprimentos variáveis em seus segmentos.

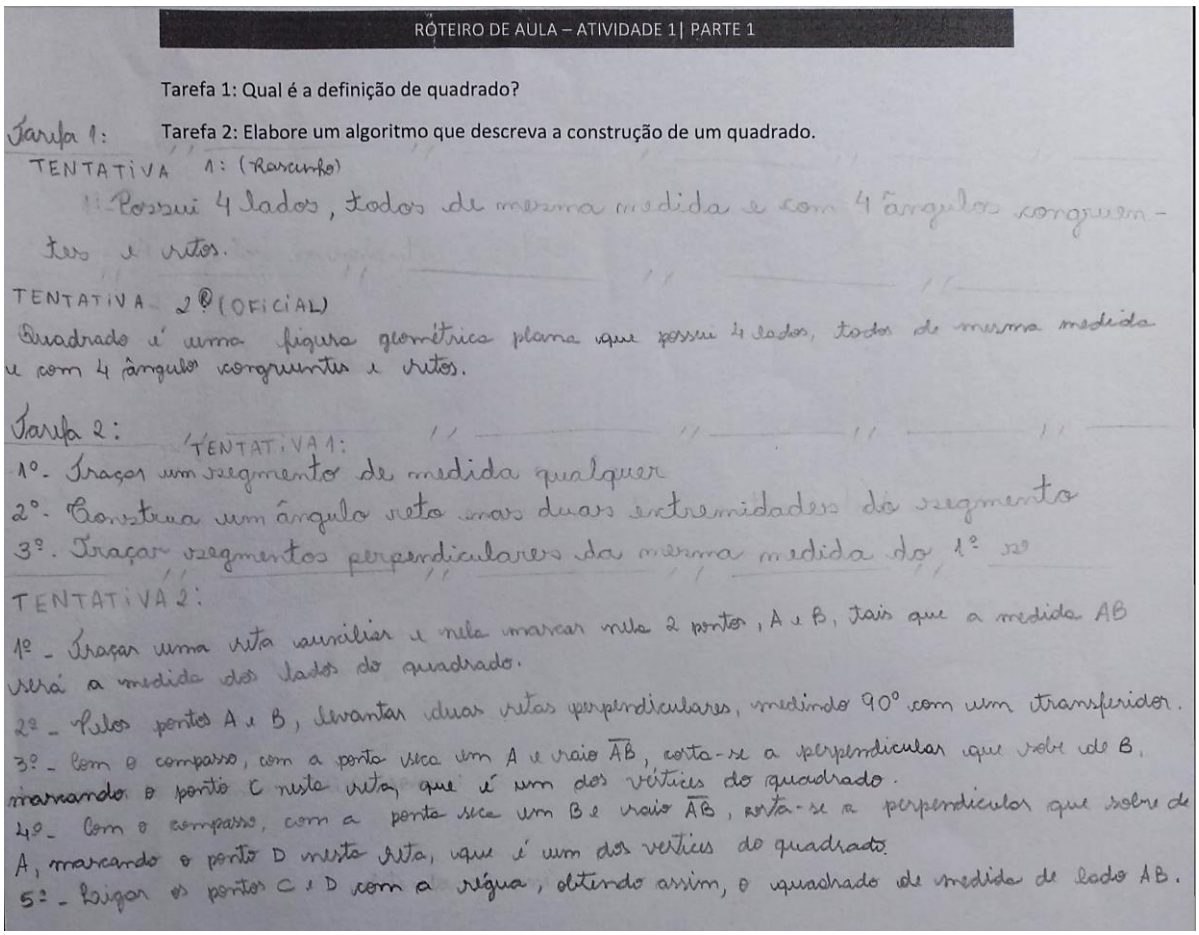
3 - Os três lados diferentes e os angul

4- 1° passo: traçar o seg $\overline{AB}$

2° passo: traçar o seg $\overline{BC}$, com a medida igual a metade de $\overline{AB}$

3° passo: traçar o seg $\overline{AC}$, com a medida igual a soma do seg $\overline{AB} + \overline{BC}$

Algoritmos dupla D1



RÓTEIRO DE AULA – ATIVIDADE 1| PARTE 1

Tarefa 1: Qual é a definição de quadrado?

Tarefa 2: Elabore um algoritmo que descreva a construção de um quadrado.

Tarefa 1:

TENTATIVA 1: (Rascunho)

1- Possui 4 lados, todos de mesma medida e com 4 ângulos congruentes e retos.

TENTATIVA 2ª (OFICIAL)

Quadrado é uma figura geométrica plana que possui 4 lados, todos de mesma medida e com 4 ângulo congruentes e retos.

Tarefa 2:

TENTATIVA 1:

1º. Traçar um segmento de medida qualquer

2º. Construa um ângulo reto nas duas extremidades do segmento

3º. Traçar segmentos perpendiculares da mesma medida do 1º

TENTATIVA 2:

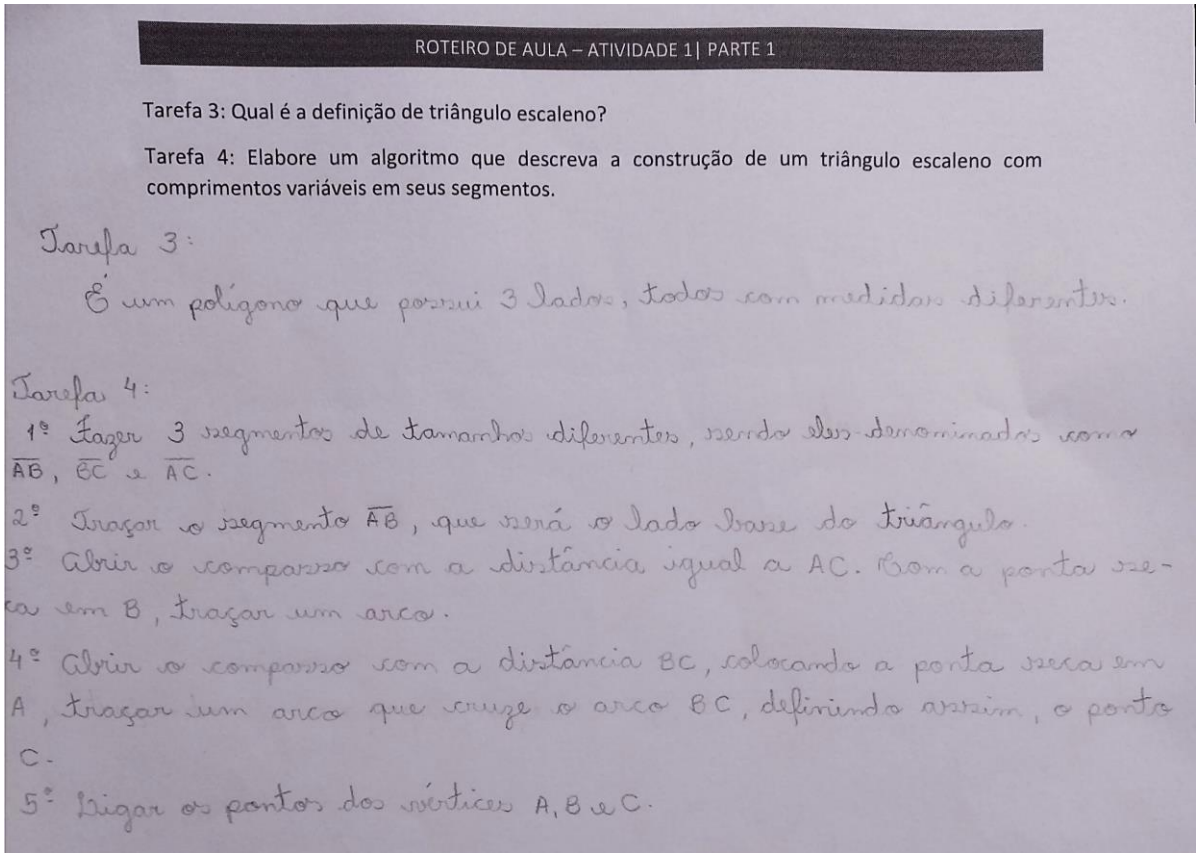
1º - Traçar uma reta auxiliar e nela marcar nela 2 pontos, A e B, tais que a medida AB será a medida dos lados do quadrado.

2º - Pelos pontos A e B, levantar duas retas perpendiculares, medindo 90° com um transferidor.

3º - Com o compasso, com a ponta seca em A e raio $\overline{AB}$, corta-se a perpendicular que sobe de B, marcando o ponto C nesta reta, que é um dos vértices do quadrado.

4º - Com o compasso, com a ponta seca em B e raio $\overline{AB}$, corta-se a perpendicular que sobe de A, marcando o ponto D nesta reta, que é um dos vertices do quadrado.

5º - Ligar os pontos C e D com a régua, obtendo assim, o quadrado de medida de lado AB.



ROTEIRO DE AULA – ATIVIDADE 1| PARTE 1

Tarefa 3: Qual é a definição de triângulo escaleno?

Tarefa 4: Elabore um algoritmo que descreva a construção de um triângulo escaleno com comprimentos variáveis em seus segmentos.

Tarefa 3:

É um polígono que possui 3 lados, todos com medidas diferentes.

Tarefa 4:

1º Fazer 3 segmentos de tamanhos diferentes, sendo eles denominados como $\overline{AB}$, $\overline{BC}$ e $\overline{AC}$.

2º Traçar o segmento $\overline{AB}$, que será o lado base do triângulo.

3º Abrir o compasso com a distância igual a AC. Com a ponta seca em B, traçar um arco.

4º Abrir o compasso com a distância BC, colocando a ponta seca em A, traçar um arco que cruze o arco BC, definindo assim, o ponto C.

5º Ligar os pontos dos vértices A, B e C.

Algoritmos dupla D2

ROTEIRO DE AULA – ATIVIDADE 1| PARTE 1

Tarefa 1: Qual é a definição de quadrado?

Tarefa 2: Elabore um algoritmo que descreva a construção de um quadrado.

① Quadrado é uma figura geométrica, sendo um quadrilátero regular, com quatro lados de mesmo comprimento e quatro ângulos retos.

② 1°) Construa dos pontos P e Q

2°) Faça um segmento de reta "a", ligando os pontos P e Q.

3°) Trace uma reta "b" que seja perpendicular a reta "a" e que intercepte o ponto P.

4°) Com a medida do segmento $\overline{PQ}$, defina um ponto S, que intercepte a reta "b", de modo que a medida $\overline{PQ} = \overline{PS}$

5°) Construa uma reta "c" que seja paralela a reta "a" e intercepte o ponto S.

6°) Com a medida do segmento $\overline{PQ}$, defina um ponto T, que intercepte a reta "c", de modo que a medida $\overline{PQ} = \overline{ST}$.

7°) Trace um segmento "d" que ligue os pontos Q e T.

ROTEIRO DE AULA – ATIVIDADE 1| PARTE 1

Tarefa 3: Qual é a definição de triângulo escaleno?

Tarefa 4: Elabore um algoritmo que descreva a construção de um triângulo escaleno com comprimentos variáveis em seus segmentos.

③ O triângulo escaleno é um tipo de polígono que possui três lados, todos com medidas diferentes

④ 1.º Precisaremos de compasso, lápis e regua

2.º Com lápis e regua traçar 3 segmentos AB, CD e EF com tamanhos diferentes

3.º Usando lápis e regua traçar uma reta auxiliar, definido um centro G, fazendo com compasso abertura em AB e transportar, com a ponta seca do compasso um G, a medida de AB sobre a reta r, formando o ponto H

4.º Pegar a medida CD e com centro em G fazer um semi circulo com o compasso

5.º Pegar a medida de EF, com centro em H, fazer o semi circulo

6.º O ponto de intercessão entre os semi circulos formam o ponto I

7.º Ligar os pontos G H I formando o triângulo escaleno.

Algoritmos dupla D3

ROTEIRO DE AULA – ATIVIDADE 1| PARTE 1

Tarefa 1: Qual é a definição de quadrado?

Tarefa 2: Elabore um algoritmo que descreva a construção de um quadrado.

01.) É uma figura geométrica fechada com 4 lados de mesma medida e 4 ângulos retos internos, lados 2 a 2 paralelos.

02.)

1. Traçar uma reta r.
2. Tracar uma reta perpendicular a reta r, a interseção dessas retas será o ponto A.
3. Com um compasso de abertura qlqr e ponta seca em A, marcar os pontos B e D sobre as retas r e s respectivamente
4. tracar um semi-arco a partir de B com a mesma abertura e realizar o mesmo procesvo no ponto D, obtendo assim o ponto C.
5. Com o auxílio de uma régua efetuar a ligação entre os pontos BC e CD.

ROTEIRO DE AULA – ATIVIDADE 1| PARTE 1

Tarefa 3: Qual é a definição de triângulo escaleno?

Tarefa 4: Elabore um algoritmo que descreva a construção de um triângulo escaleno com comprimentos variáveis em seus segmentos.

03) É uma figura geométrica com 3 lados de diferentes medidas.

04) • Traça 3 segmentos de reta de medidas diferentes AB, CD e EF.

• Construa uma semi-reta com ponto de origem O.

• Com compasso transporte a medida o segmento AB na semi-reta a partir da origem obtendo o ponto P sobre a semi. reta.

• Ponta seca na origem da semi-reta com a medida CD, traçando um semi-arco, o mesmo processo a partir do ponto P traçar outro semi-arco. No ponto onde ambos se encontram será o ponto Q.

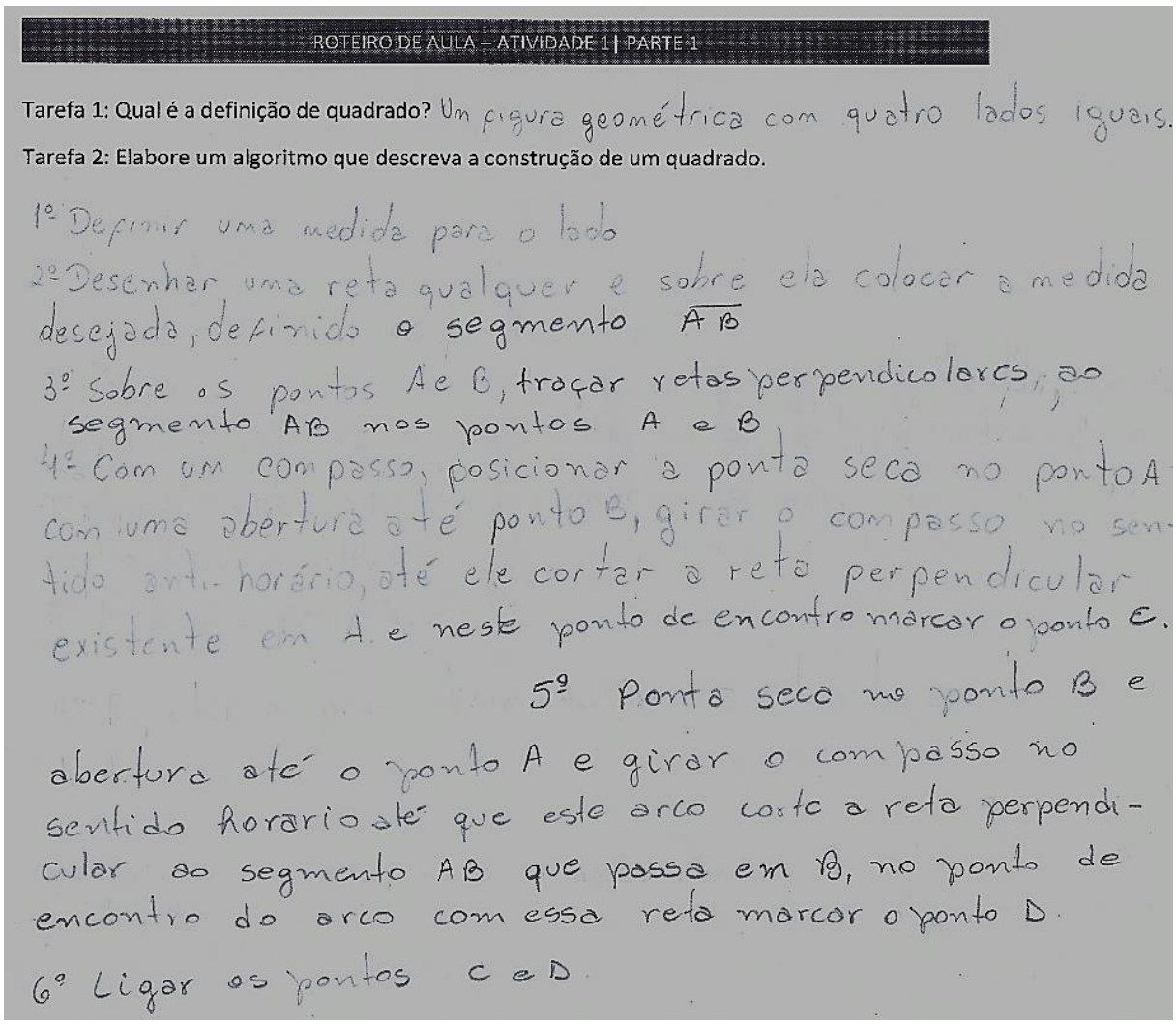
• Com auxílio da régua ligar todos os pontos obtendo assim um triângulo escaleno.

$\overline{AB} = \overline{OP}$

$\overline{CD} = \overline{OQ}$

$\overline{EF} = \overline{PQ}$

Algoritmos dupla D4



ROTEIRO DE AULA – ATIVIDADE 1 | PARTE 1

Tarefa 1: Qual é a definição de quadrado? Um figura geométrica com quatro lados iguais.

Tarefa 2: Elabore um algoritmo que descreva a construção de um quadrado.

1º Definir uma medida para o lado

2º Desenhar uma reta qualquer e sobre ela colocar a medida desejada, definido o segmento $\overline{AB}$

3º Sobre os pontos A e B, traçar retas perpendiculares, ao segmento AB nos pontos A e B.

4º Com um compasso, posicionar a ponta seca no ponto A com uma abertura até ponto B, girar o compasso no sentido anti-horário, até ele cortar a reta perpendicular existente em A e neste ponto de encontro marcar o ponto C.

5º Ponta seca no ponto B e abertura até o ponto A e girar o compasso no sentido horario até que este arco corte a reta perpendicular ao segmento AB que passa em B, no ponto de encontro do arco com essa reta marcar o ponto D.

6º Ligar os pontos C e D.

ROTEIRO DE AULA – ATIVIDADE 1| PARTE 1

Tarefa 3: Qual é a definição de triângulo escaleno?
Triângulo que possui três lados com medidas diferentes

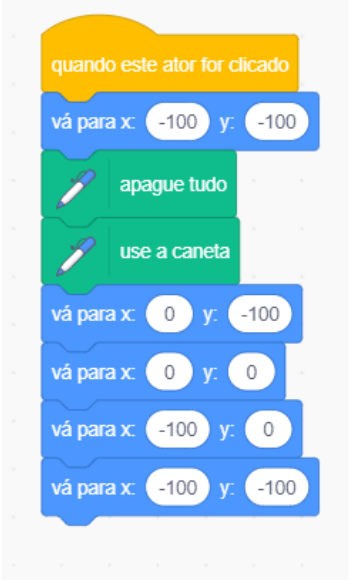
Tarefa 4: Elaboré um algoritmo que descreva a construção de um triângulo escaleno com comprimentos variáveis em seus segmentos.

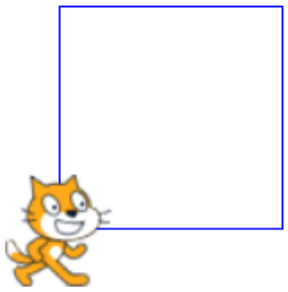
1º Construir uma reta com uma medida qualquer;
2º Cupartir da extremidade desta reta, fazer uma nova também com medida qualquer e uma abertura ângular entre elas.
3º Ligar as extremidade de ambas

APÊNDICE F – ALGORITMOS NO *SCRATCH*.

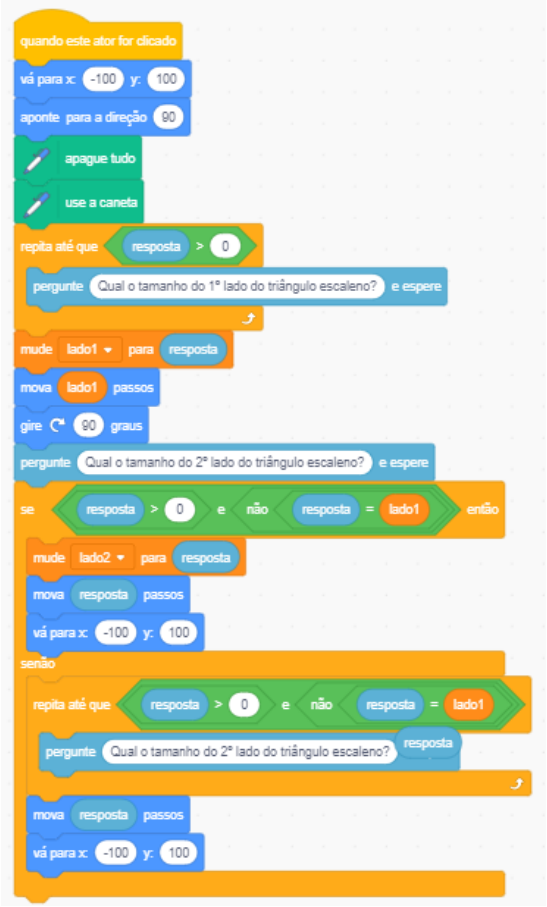
Códigos do sujeito I1

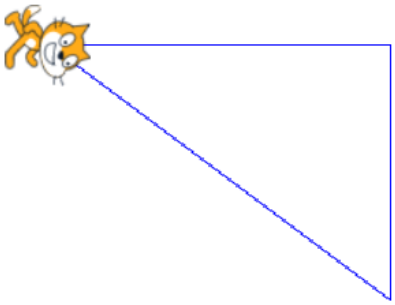
Quadrado





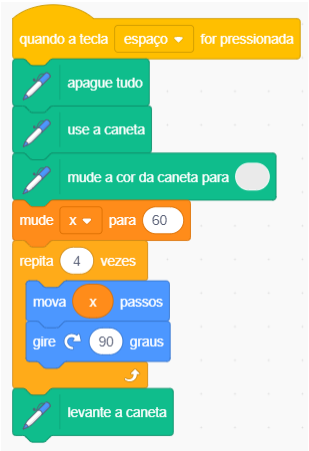
Triângulo

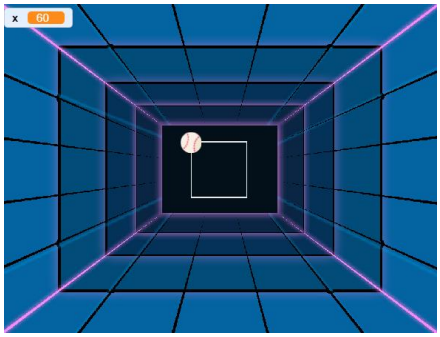




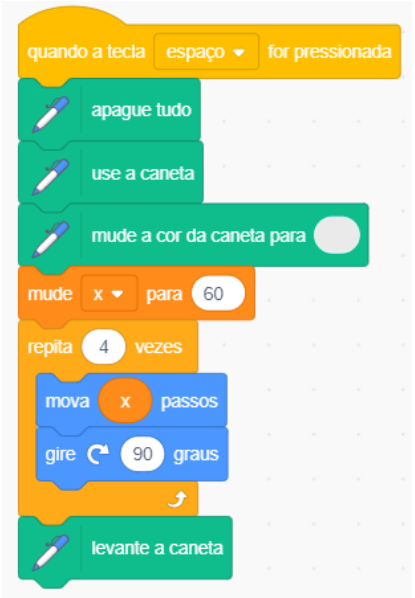
Códigos do sujeito I2

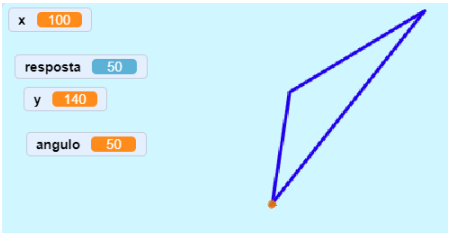
Quadrado





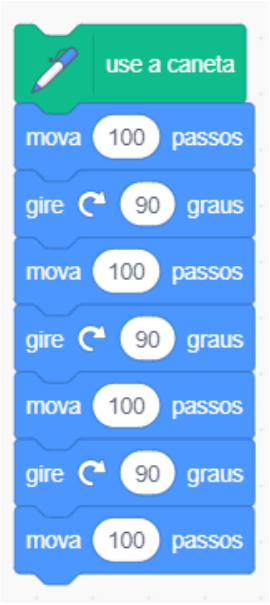
Triângulo

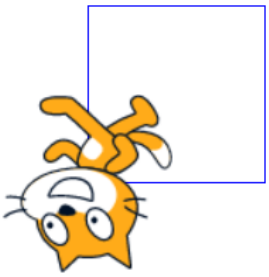




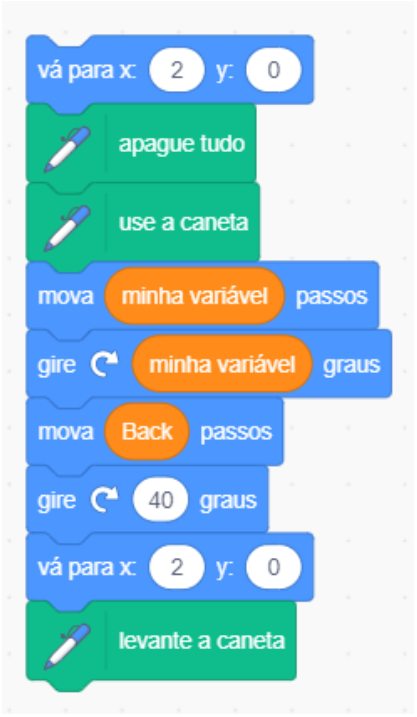
Códigos do sujeito N1

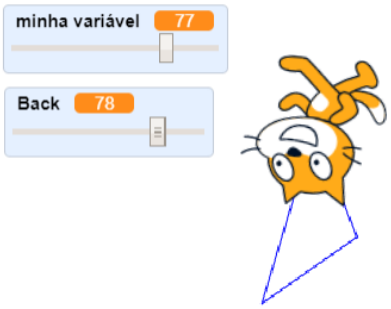
Quadrado





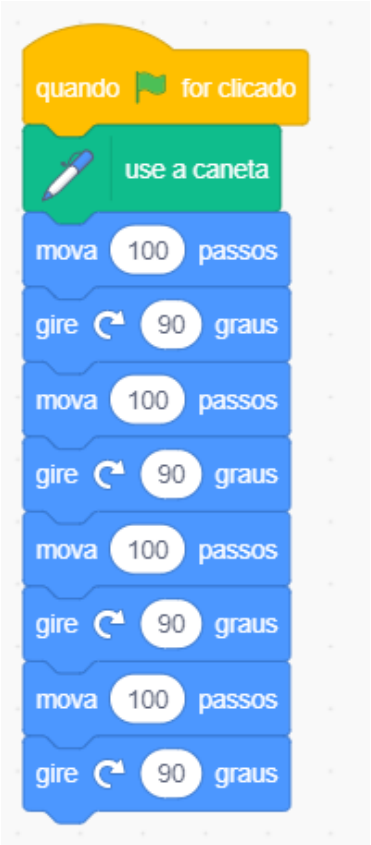
Triângulo

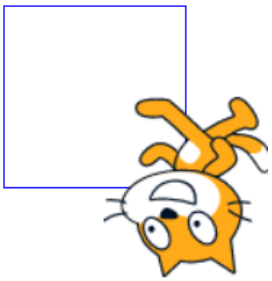




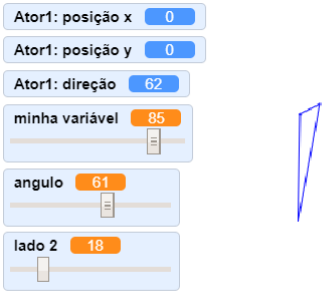
Códigos do sujeito N2

Quadrado



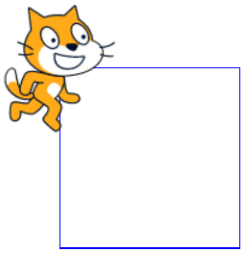


Triângulo

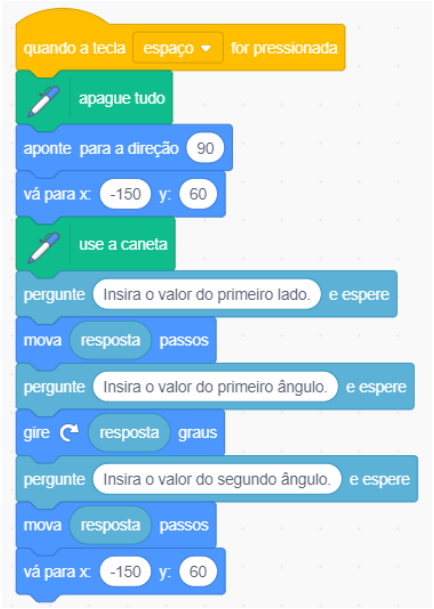


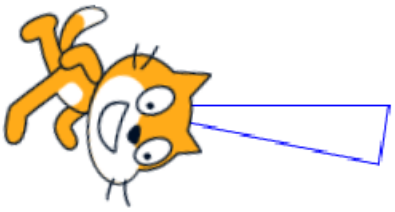
Códigos do sujeito N3

Quadrado



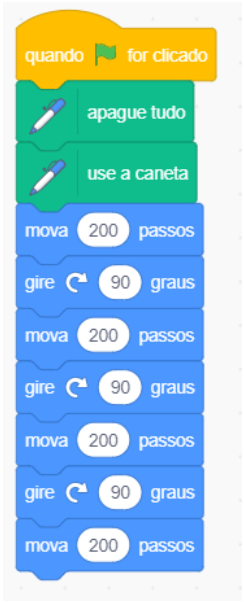
Triângulo

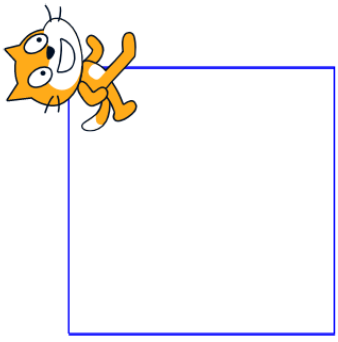




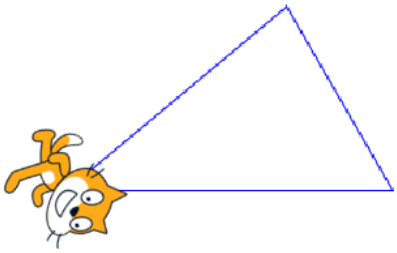
Códigos do sujeito N4

Quadrado



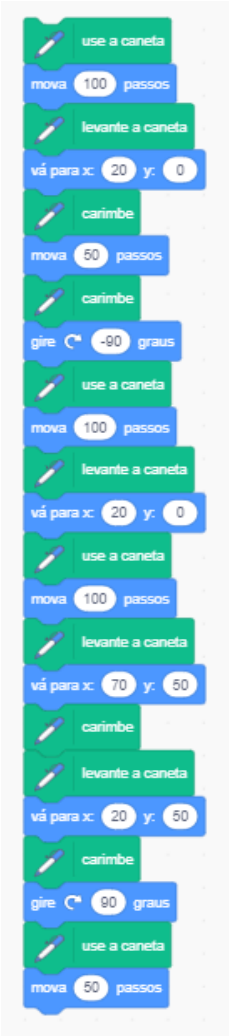


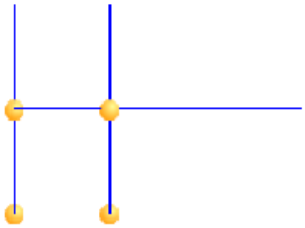
Triângulo



Códigos da dupla D1

Quadrado



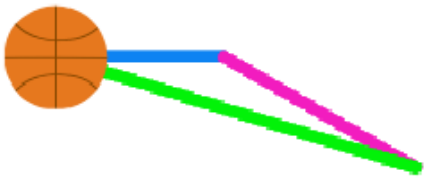


Triângulo

Códigos da dupla D2

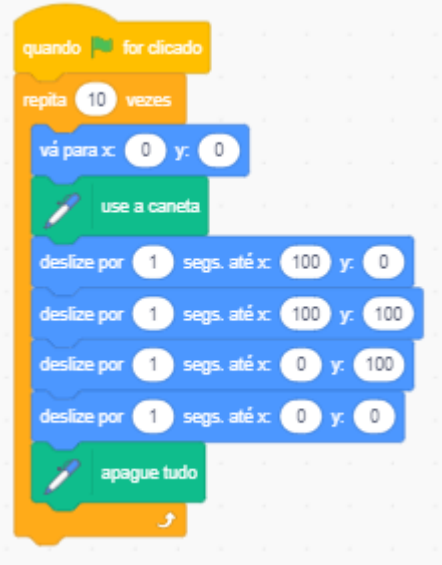
Quadrado

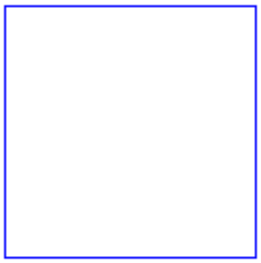
Triângulo



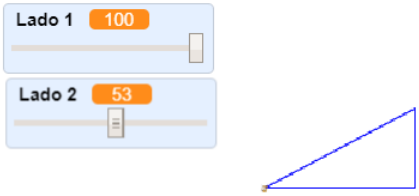
Códigos da dupla D3

Quadrado



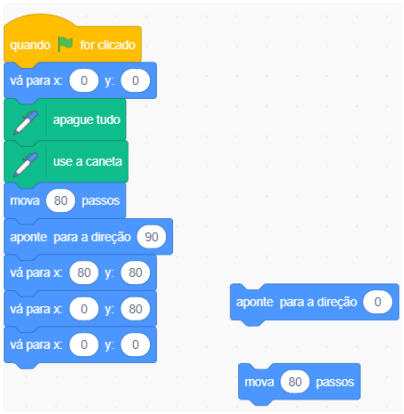


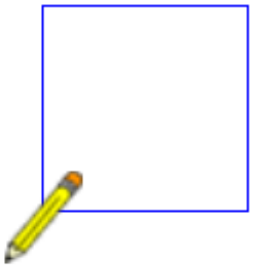
Triângulo



Códigos da dupla D4

Quadrado





Triângulo