UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE CIÊNCIAS APLICADAS A EDUCAÇÃO
DEPARTAMENTO DE CIÊNCIAS EXATAS
LICENCIATURA EM CIÊNCIA DA COMPUTAÇÃO

PENSAMENTO COMPUTACIONAL E PROGRAMAÇÃO INTRODUTÓRIA: UM ESTUDO DE CASO SOBRE COMPETÊNCIAS DESENVOLVIDAS NA PROGRAMAÇÃO EM BLOCOS COM O CODE.ORG

AHEMENSON FERNANDES CAVALCANTE

RIO TINTO - PB
2016

AHEMENSON FERNANDES CAVALCANTE

PENSAMENTO COMPUTACIONAL E PROGRAMAÇÃO INTRODUTÓRIA: UM ESTUDO DE CASO SOBRE COMPETÊNCIAS DESENVOLVIDAS NA PROGRAMAÇÃO EM BLOCOS COM O CODE.ORG

Monografia apresentada para obtenção do grau de Licenciado (a) à banca examinadora no Curso de Licenciatura em Ciência da Computação do Centro de Ciências Aplicadas e Educação (CCAE), Campus IV da Universidade Federal da Paraíba.
Orientadora: Profª. Msc. Ana Liz Souto Oliveira de Araújo.

RIO TINTO - PB
2016

C376p Cavalcante, Ahemenson Fernandes.
Pensamento computacional e programação introdutória: um estudo de caso sobre competências desenvolvidas na programação em blocos com o code.org. / Ahemenson Fernandes Cavalcante. – Rio Tinto: [s.n.], 2016.
88 f. : il.-

Orientador (a): Prof. Msc. Ana Liz Souto Oliveira de Araújo.
Monografia (Graduação) – UFPB/CCAE.

1. Programação - computação. 2. Programação Visual. 3. Ciência da computação.

UFPB/BS-CCAE CDU: 004.43(043.2)

AHEMENSON FERNANDES CAVALCANTE

PENSAMENTO COMPUTACIONAL E PROGRAMAÇÃO INTRODUTÓRIA: UM ESTUDO DE CASO SOBRE COMPETÊNCIAS DESENVOLVIDAS NA PROGRAMAÇÃO EM BLOCOS COM O CODE.ORG

Trabalho de Conclusão de Curso submetido ao Curso de Licenciatura em Ciência da Computação da Universidade Federal da Paraíba, Campus IV, como parte dos requisitos necessários para obtenção do grau de LICENCIADO EM CIÊNCIA DA COMPUTAÇÃO.

Assinatura do autor: _____

APROVADO POR:

_____
Orientadora: Profª. Msc. Ana Liz S. O. De Araújo
Universidade Federal da Paraíba – Campus IV

_____
Profª. Drª. Flávia Veloso Costa Souza
Universidade Federal da Paraíba – Campus IV

_____
Profª. Msc. Jarbele Cassia Da Silva
Universidade Federal da Paraíba – Campus IV

RIO TINTO – PB
2016

Dedicado aos meus pais que tanto amo, Elisabete e Antoniel, por todo o esforço para tornar seus filhos dignos e honrados.

AGRADECIMENTOS

Aos meus pais, que mesmo tendo sido negado o pleno direito a educação pelas difíceis circunstâncias impostas nunca me abdicaram deste privilégio e sempre me apoiaram para atingir grandes realizações como a que alcanço hoje. A minha orientadora Ana Liz Souto pelas sugestões e incentivos, contribuições sem as quais não teria alcançado tamanho feito, muito obrigado. Aos colegas da turma ingressante no semestre letivo de 2011.1, em especial ao grupo mais próximo de convívio, Danilo Raniery, Emiliano Oliveira, Gilson Trajano e Italo Dantas, companheiros que por meio das conversas e risos soltos contribuíram para alijar o peso dessa formação acadêmica. Também sou grato por ter conhecido outros colegas de curso no qual tive o prazer de contemplar seus valores e virtudes. Obrigado, meu amigo e irmão Leonardo dos Santos pelo apoio durante todo esse percurso e por ser meus braços quando não pude levantar, por enxergar quando não pude ver, por poder contar com você em todas as horas. Aos professores desta instituição que tiver o prazer de me deparar, seus ensinamentos transcendem ao que fora estampado no quadro branco, vocês me presentearam com gratificantes exemplos, verdadeiras lições que guardarei para toda a vida. Não poderia deixar de agradecer ao professor e doutor Lusival Antônio Barcellos, idealizador do projeto que concedeu oportunidade a mim e a centenas de jovens de origem humilde a obter o acesso a uma formação superior. E por fim, agradeço a você que soube agarrar a oportunidade dada e superar inúmeras barreiras impostas nessa caminhada, a você que fui me despedindo paulatinamente, obrigado por ter se superado, permitindo um novo renascer.

RESUMO

O Pensamento Computacional desponta como uma nova abordagem para resolver problemas no mundo contemporâneo. E se caracteriza por ir além dos processos mentais. A ele estão envolvidos atitudes e valores que reunidos formam as competências tão requeridas para realização de tarefas cotidianas cada vez mais informatizadas. A medida em que os computadores vão se tornando recursos indispensáveis no auxílio das atividades profissionais humanas, torna-se necessário preparar as novas gerações para exercer esse domínio, e a Programação Introdutória é uma porta de entrada para isso, pois também contribui para o desenvolvimento de habilidades provenientes do pensamento computacional. Este trabalho busca compreender essa relação investigando o aprendizado da programação introdutória com a plataforma Code.org e o desenvolvimento do pensamento computacional em alunos do ensino médio. Para isso foi realizado um estudo de caso na plataforma para identificar as competências promovidas nos alunos através das atividades de Programação em Blocos ou Programação Visual. Também foi realizado um teste experimental em uma oficina de programação para identificar quais habilidades os alunos participantes desenvolveram. Os resultados do estudo de caso demonstraram que há um conjunto de conceitos e práticas computacionais promovidas em atividades de programação na plataforma Code.org que podem ser compreendidas como competências do pensamento computacional. O resultado do teste aplicado na oficina demonstrou indícios de que os alunos podem desenvolver e ampliar essas competências após o ensino da programação introdutória, mas ainda há necessidade de realizar mais pesquisas para aprofundar conhecimentos e chegar as devidas conclusões de como e quais habilidades são desenvolvidas nessa relação.

Palavras-Chave: Pensamento computacional; Programação Introdutória; Code.org; Estudo de Caso; Programação em Blocos; Programação Visual.

ABSTRACT

Computational thinking stands out as a new approach to solving problems in the contemporary world. And is characterized by go beyond the mental processes, it involved attitudes and values that brought together form the skills so necessary for everyday tasks increasingly computerized. As computers become indispensable resources in aid of the professional human activities, it is necessary to prepare new generations to exercise that dominance, and the Introductory Programming is a gateway to this as it also contributes to the development of skills from the computational thinking. This work tries to understand this relationship investigating learning introductory programming with the Code.org platform and the development of computational thinking in middle school students. For this we conducted a case study on the platform to identify the skills promoted in students through the activities of programming in blocks or Visual Programming, was also carried out an experimental test in a programming workshop to identify what skills students have developed. The results of the case study showed that there is a set of computational concepts and practices promoted in programming activities in Code.org platform which can be understood as a computational thinking skills. The result of the test applied in the workshop demonstrated evidence that students can develop and extend these powers after the teaching of introductory programming, but there is still a need for more research to deepen knowledge and reach conclusions on how and what skills are developed in this relationship.

Keywords: Computational thinking; Introductory Programming; Code.org; Case study; programming in blocks; Visual Programming.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS

PIBID	Programa Institucional de Bolsas de Iniciação à Docência
IES	Instituição de Ensino Superior
ISTE	Sociedade Internacional de Tecnologia na Educação
CSTA	Associação de Professores de Ciência da Computação
MIT	Instituto de Tecnologia de Massachusetts

SUMÁRIO

1 INTRODUÇÃO

Os avanços tecnológicos produzidos pelo homem acarretaram mudanças em diversos aspectos da sociedade, exigindo assim aos indivíduos uma gradativa adaptação às novas exigências impostas a cada momento da história. E as progressivas transformações ocorridas nos últimos tempos possibilitaram a germinação de um novo padrão de formação educacional.

No início do século XX com o progresso das cidades e das máquinas entra em cena o paradigma fordista, e a educação influenciada por esse modelo aplica o ensino de conteúdos de forma extensa e fragmentada. “No Fordismo, o controle da produção está centralizado nas mãos de especialistas que planejam a tarefa, fragmentando-a em subtarefas simples para serem dominadas e realizadas por trabalhadores com pouca qualificação” (VALENTE, 1999, p. 32).

Em oposição ao modelo fordista o paradigma enxuto, estabelecido neste século exige melhorias na qualificação profissional e na formação dos indivíduos no que diz respeito ao desenvolvimento das capacidades cognitivas, afetivas e sociais.

Valente (1999) destaca as implicações desse paradigma vigente e a demanda de mudanças na Educação:

> Já a produção enxuta exige trabalhadores melhor qualificados, capazes de assumir responsabilidades, tomar decisões e buscar soluções para problemas que ocorrem durante o processo de produção. [...] Essa mudança na produção de bens e nos serviços implicará, certamente, mudanças no sistema educacional. A Educação deverá operar segundo esse novo paradigma. Isso implicará professores melhor qualificados, não para empurrar a informação ao aluno, mas para saber criar situações em que o aluno “puxa” a informação. Mais ainda, somente ter a informação não implica ter conhecimento. O conhecimento deverá ser fruto do processamento dessa informação, aplicação dessa informação processada na resolução de problemas significativos e reflexão sobre os resultados obtidos. Isso exigirá do aluno a compreensão do que está fazendo para saber tomar decisões, atuar e realizar tarefas (VALENTE, 1999, p. 32).

Dessa forma, por se caracterizar como um processo institucionalizado de formação de indivíduos, a educação escolar recebe a tarefa de preparar as novas gerações auxiliando no desenvolvimento de certas capacidades consideradas imprescindíveis para desempenhar funções na sociedade do conhecimento.

Para Valente (1999, p. 36) os alunos devem ser críticos, terem a capacidade de realizar reflexões e depurações que possibilitem alcançar níveis cada vez maiores de criatividade e atuação. Além disso, precisam saber trabalhar em equipe para atingir junto com outras pessoas a solução para problemas complexos. Em sintonia com o

seu pensamento, Gadotti (2007) defende o desenvolvimento de habilidades como autonomia, saber comunicar-se, pesquisar, construir, ter raciocínio lógico, saber trabalhar de forma colaborativa, articular conhecimento com prática e com outros saberes.

Logo, o desafio está no desenvolvimento de capacidades que permitirão aos sujeitos uma melhor interpretação do mundo ao seu redor. E como estamos em uma sociedade do conhecimento e um fator característico dela é a crescente evolução da computação e gradual presença desse instrumento na vida das pessoas. Tal presença é tão forte que poderíamos considerá-lo como uma poderosa ferramenta auxiliadora nos processos de formação educacional com vista ao desenvolvimento dessas capacidades.

A computação atualmente é uma área que exerce uma forte influência e seus avanços contribuem para impulsionar diretamente e indiretamente os rumos dos mais diversos setores da sociedade. “A tecnologia computacional tem mudado a prática de quase todas as atividades, das científicas às de negócio até às empresariais. E o conteúdo e prática educacionais também seguem essa tendência” (BARANAUSKAS *et al.*, 1999, p. 49).

A evolução da computação ocorrida nos últimos anos, a gradual presença desse instrumento na vida das pessoas, e a ampliação do acesso à informação possibilitado por esse aparelho colaboraram para a formação dessa sociedade atual. Desta forma, ela permeia todas as atividades humanas, das artes às tecnologias, e hoje não se pode imaginar uma sociedade sem computador (NUNES, 2008).

Diante deste panorama de novo século no qual a computação de forma gradual se faz presente a cada momento do nosso cotidiano, fica evidente que essas transformações irão requerer de todos nós uma maior relação com a computação, seja por meio da correlação entre diferentes profissões. Como afirma Wing (2006), futuros sociólogos, economistas, músicos, educadores deverão interagir com profissionais da Computação através de um pensamento interdisciplinar ou, seja como recurso para solucionar problemas através da construção de novas abordagens computacionais. O fato é que caberá a nós o entendimento e uma interação maior além do passivo uso dos produtos resultantes da computação.

1.1 MOTIVAÇÃO

Neste capítulo é descrito a motivação para a elaboração do trabalho, como também os objetivos gerais e específicos da pesquisa, a metodologia adotada para alcançá-los e a organização estrutural da presente obra.

1.1.1 Atuação no PIBID Computação

A motivação inicial para a escolha do tema de pesquisa referente a esse trabalho surgiu através da minha atuação ao longo de dois anos no subprojeto de Computação do Programa Institucional de Bolsas de Iniciação à Docência. O PIBID é um programa governamental que buscar aperfeiçoar e valorizar a formação dos alunos de Licenciatura pertencentes às IES Instituições de Ensino Superior.

Nossa atuação pode ser condensada em um apanhado de atividades oferecidas em forma de cursos e oficinas com conteúdo relativos da área da Computação. Essas atividades são dedicadas aos alunos das escolas estaduais vinculadas ao projeto, além disso, são promovidas ações para a capacitação de professores no uso das tecnologias em sala de aula. E foi por meio da experiência obtida em atividades de ensino de programação introdutória que fui apresentado ao tema pensamento computacional.

1.1.2 Um tema contemporâneo

O pensamento computacional, termo que está se popularizando a cada ano, e ganhou notoriedade em 2006 após o lançamento do artigo da pesquisadora Jannette Wing. Sua obra oferecia uma definição primária do tema, e estimava a importância e necessidade da sua inclusão nos processos de formação educacional.

Mediante um aprofundamento do tema pensamento computacional através de pesquisas ficou evidente se tratar de algo relativamente novo na literatura científica (BARCELOS *et al.*, 2015). Porém, apesar de ser um tema recente já se encontra acolhido por parte de diversos pesquisadores da comunidade acadêmica que defendem a adoção de práticas do pensamento computacional nos diversos níveis de ensino (MANNILA *et al.*, 2014) (BARR; STEPHENSON, 2009).

Práticas do pensamento computacional já são Realidade em currículos de escolas da América do norte e de países da Europa (CARVALHO; CHAIMOWICZ; MORO, 2013). No Brasil as iniciativas de pesquisa ainda são incipientes,

principalmente quando se trata de estudos sobre a formação e desenvolvimento do pensamento computacional em contexto da Educação Nacional.

O pensamento computacional se caracteriza como um processo com vista à resolução de problemas por meio de conceitos, recursos e ferramentas computacionais. Para Wing (2006), o pensamento computacional baseia-se em fundamentos da Computação, envolvendo a resolução de problemas, a capacidade de projetar sistemas e a compreensão do comportamento humano. É uma forma de aplicar conceitos bastante trabalhados na computação, mas que não são exclusivos somente desta área, mas que podem ser aplicados na resolução de problemas dos mais variados.

Sua aplicabilidade para resolução de problemas nos mais diversos campos do conhecimento o torna uma competência fundamental para todas as pessoas, não apenas para cientistas da computação, despontando como um requisito elementar para a formação básica dos profissionais de todas as áreas nos próximos anos (WING, 2006).

1.1.3 A programação introdutória e o pensamento computacional

O computador como ferramenta de ensino na educação inicialmente seguiu o paradigma instrucionista. Segundo Valente (1997), a abordagem pedagógica de instrução auxiliada pelo computador se dá quando sua utilização se restringe a transmissão de informações ao aluno, assumindo o papel de máquina de ensino. Atividades envolvendo uso de softwares tutoriais e exercícios são associados a esse paradigma. Essa abordagem educativa pode até alcançar os resultados pretendidos de forma satisfatória, porém, não propicia um aprimoramento do intelecto do aluno, não estimula sua criatividade e senso crítico.

Por outro lado, no paradigma construcionista o aluno assume o papel de instrutor da máquina. Deste modo, essa abordagem de ensino proporciona ao aluno a construção do seu conhecimento, permitindo que ele busque novas informações complementares para criar sua própria solução:

> Como auxiliar do processo de construção do conhecimento, o computador deve ser usado como uma máquina para ser ensinada. Nesse caso, é o aluno quem deve passar as informações para o computador [...]. Isso significa que o aluno deve representar suas idéias para o computador, ou seja, ensinar o computador a resolver a tarefa em questão [...] A construção do conhecimento acontece pelo fato de o aluno ter que buscar novas informações para complementar ou alterar o que ele já possui. [...]. Além

> disso, o aluno está criando suas próprias soluções, está pensando e aprendendo sobre como buscar e usar novas informações (aprendendo a aprender) (VALENTE, 1997, p. 3).

Dentro dessa abordagem a principal atividade utilizada é a programação de computadores, no qual o aluno através do uso de uma linguagem de programação transfere comandos à máquina de forma a descrever uma representação idealizada da solução proposta. O grande proveito do uso da programação é fazer o aluno empenhar-se na busca de estratégias para resolução dos problemas fazendo-o pensar e refletir a todo o momento.

O ensino de programação é uma área que recebeu grande destaque e nos últimos anos, muitas iniciativas foram realizadas para disseminar e estimular o ensino de programação introdutória.

O surgimento de plataformas com essa finalidade, e que adotam uma abordagem de ensino composta por traços lúdicos, ambientação amigável e fácil manuseio são frutos dessas iniciativas. Um projeto que vem se destacando é a plataforma de ensino de programação Code.org. Seu objetivo vai além do incentivo a aprender a programar. O projeto tem como perspectiva trazer os conceitos básicos da Ciência da Computação para dentro do ambiente escolar.

> Nossa visão é que todos os alunos, de todas as escolas, devem ter a oportunidade de aprender programação de computadores. Acreditamos que a ciência da computação deve fazer parte do currículo escolar, ao lado de áreas como: ciência, engenharia, matemática, biologia, física, química e álgebra" (CODE.ORG, 2016).

Tendo em vista os aspectos observados acerca da programação introdutória e pensamento computacional, o presente trabalho busca apresentar a relação existente entre ambos os temas, e mais especificamente mostrar os efeitos decorrentes da união das duas práticas nos estudantes. As indagações levantadas e que fazem parte deste estudo estão descritas na seção subsequente do presente documento.

1.2 QUESTÕES DE PESQUISAS

O presente trabalho de pesquisa busca responder a grande **Questão Geral**: *Qual a correlação existente entre o aprendizado da programação introdutória com a plataforma Code.org e o desenvolvimento do pensamento computacional em alunos do ensino médio?* Para atingir tal resultância primeiramente serão respondidas as seguintes **Questões Complementares**:

- *Quais as habilidades e as competências esperadas em abordagens de ensino de pensamento computacional?*

- *Quais são as habilidades e as competências do pensamento computacional que estão relacionadas ao ensino de programação introdutória?*

- *Quais as habilidades e as competências do pensamento computacional são desenvolvidas através de ambientes gráficos de ensino de programação introdutória como Code.org?*

1.3 OBJETIVOS

O corrente trabalho tem por **Objetivo Geral** compreender a relação existente entre o aprendizado da programação introdutória com a plataforma Code.org e o desenvolvimento do pensamento computacional em alunos do ensino médio.

1.3.1 Objetivos específicos

- Identificar quais as habilidades e as competências esperadas em abordagens de ensino de pensamento computacional;
- Identificar quais são as habilidades e as competências do pensamento computacional que estão relacionadas ao ensino de programação introdutória;
- Exemplificar quais habilidades do pensamento computacional podem ser desenvolvidas através de ambientes gráficos de ensino de programação introdutória como Code.org;

1.4 METODOLOGIA

A pesquisa bibliográfica realizada inicialmente nesse presente trabalho tem por intento identificar na literatura científica trabalhos que permitam a compreensão das habilidades e competências provenientes do pensamento computacional. Além disso, se faz necessário de forma preliminar buscar compreender a correlação entre o desenvolvimento dessas qualidades e o ensino de programação introdutória.

Na etapa posterior é realizado um estudo de caso. Essa abordagem adotada objetiva fazer uma investigação a respeito da presença de elementos na plataforma Code.org que possibilitem o desenvolvimento de habilidades e competências relacionadas ao pensamento computacional.

Nas subseções abaixo estão dispostos mais detalhes sobre as abordagens adotadas.

1.4.1 Definição de pesquisa

De acordo com Gil (2002), a pesquisa é definida como um procedimento racional e sistemático que tem como intuito fornecer respostas aos problemas levantados. Ainda segundo o autor, na pesquisa utilizam-se métodos e técnicas durante seus vários estágios, sendo que a etapa inicial parte da formulação do problema e finaliza na apresentação dos resultados.

Para Marconi e Lakatos a pesquisa pode ser compreendida como:

> Um procedimento formal, com método de pensamento reflexivo, que requer um tratamento científico e se constitui no caminho para conhecer a realidade ou para descobrir verdades parciais. O desenvolvimento de um projeto de pesquisa compreende seis passos: 1. Seleção do tópico ou problema para a investigação. 2. Definição e diferenciação do problema. 3. Levantamento de hipóteses de trabalho. 4. Coleta, sistematização e classificação dos dados. 5. Análise e interpretação dos dados. 6. Relatório do resultado da pesquisa (MARCONI; LAKATOS, 2003, p. 155).

Portanto, a pesquisa científica é caracterizada pela presença marcante de procedimentos que incluem métodos e técnicas rigorosas para obter um novo conhecimento, reafirmar um já existente ou invalidá-lo perante novas descobertas.

1.4.2 Objetivos da pesquisa: exploratória e descritiva

Este trabalho é classificado quanto aos seus objetivos como uma pesquisa exploratória e descritiva. Exploratória por realizar um levantamento bibliográfico a fim de obter maior compreensão acerca do tema pensamento computacional. Classifica-se como descritiva pela adoção de procedimentos como o estudo de caso para relacionar quais habilidades do pensamento computacional podem ser desenvolvidas em ambientes visuais de programação como o Code.org.

Prodanove e Freitas (2013, p. 51) destacam que pesquisas exploratórias possuem um caráter introdutório, cuja finalidade é levantar primeiras informações a respeito do objeto de estudo a fim de possibilitar uma maior definição e delimitação acerca do tema abordado. Ainda segundo os autores uma pesquisa do tipo descritiva busca realizar um levantamento como registro e descrição de fatos observados durante o estudo por meio de técnicas padronizadas de coleta de dados.

Deste modo, pesquisas exploratórias buscam proporcionar maior familiaridade com o problema, para assim torná-lo mais explícito ou mesmo construir hipóteses para a ocorrência de tais problemas (GERHARDT; SILVEIRA, 2009, p. 35). Enquanto que as pesquisas descritivas pretendem descrever os fatos e fenômenos de determinada realidade (TRIVIÑOS, 1987 Apud GERHARDT; SILVEIRA, 2009, p. 35).

1.4.3 Pesquisa bibliográfica

A pesquisa bibliográfica presente neste trabalho serviu como meio de coleta de informações acerca dos temas pensamento computacional e programação introdutória. Gil (2002, p. 44) denota que “a pesquisa bibliográfica é desenvolvida com base em material já elaborado, constituído principalmente de livros e artigos científicos”.

O estudo bibliográfico ocorreu por meio de consultas a artigos científicos disponibilizados em repositórios acadêmicos, revistas de divulgação científica e obras literárias.

1.4.4 Estudo de caso

O estudo de caso presente neste trabalho objetivou relacionar quais habilidades do pensamento computacional podem ser desenvolvidas a partir do ensino de Introdução à Programação em ambientes visuais como o Code.org.

O estudo de caso pode ser definido como:

> Um estudo de caso pode ser caracterizado como um estudo de uma entidade bem definida como um programa, uma instituição, um sistema educativo, uma pessoa, ou uma unidade social. Visa conhecer em profundidade o como e o porquê de uma determinada situação que se supõe ser única em muitos aspectos, procurando descobrir o que há nela de mais essencial e característico. O pesquisador não pretende intervir sobre o objeto a ser estudado, mas revelá-lo tal como ele o percebe (GIL, 2007 apud GERHARDT; SILVEIRA, 2009, p. 39).

Fonseca (2002) argumenta que o estudo de caso pode surgir a partir de uma visão interpretativa na busca pela compreensão de um dado fenômeno a partir dos seus participantes Fonseca (2002, p. 33). Ele ainda destaca que este método visa apresentar como resultado uma perspectiva global do objeto estudado segundo a visão do investigador.

Considerando as características descritas, o processo do estudo de caso inicia-se na seleção de um dentre os cursos online disponíveis na plataforma. Em seguida é feita a investigação propriamente dita, que consiste na análise e identificação de elementos do pensamento computacional em atividades e práticas relacionadas à resolução de desafios presentes no curso selecionado. Como suporte para tal realização foi selecionado um modelo da literatura de um ambiente similar que possa orientar no processo de avaliação do Code.org.

1.4.5 Critérios de escolha

Há uma série de fatores que denotam a relevância do Code.org para a educação em computação. Dois fatores em destaque são a adoção da plataforma em escolas espalhadas pelo o mundo, e o apoio recebido de organizações de tecnologia e educação em prol da expansão do ensino de programação.

Os números apresentados em sua página exaltam o seu sucesso. Desde o lançamento em 2013, mais de duzentos milhões de alunos utilizaram o Code.org, 49% composto pelo sexo feminino, fomentando a participação feminina na área da Ciência da Computação. Outro dado em destaque é a quantidade de professores cadastrados, 321.988 professores, o que novamente evidencia o uso da plataforma em atividades educacionais (CODE.ORG, 2016).

Além das características citadas acima, a escolha do Code.org como objeto de estudo é apoiada pela adequação a certos critérios, julgados neste trabalho como fatores importantes presentes em ambiente de programação. Os critérios aqui apresentados foram gerados pelo autor desta pesquisa e incluem:

- Ambiente gratuito.
- Suporte ao idioma em português.
- Direcionado ao público iniciante.
- Disponibilidade online.
- Conteúdo definido.

Um comparativo entre a plataforma e outro ambiente de programação em blocos é apresentado na seção 4.1 do presente documento.

1.5 ORGANIZAÇÂO DO TRABALHO

O presente trabalho é organizado em seis capítulos. O primeiro capítulo apresenta uma introdução sobre os temas abordados, os fatores de motivação para o presente estudo, seus objetivos e a metodologia adotada pelo mesmo. No segundo capítulo são aprofundados os temas abrangentes através do referencial literário composto. Os trabalhos relacionados encontrados na literatura estão expostos no terceiro capítulo. O quarto capítulo apresenta a análise da plataforma Code.org através da realização de um estudo de caso. E por fim, considerações finais e trabalhos futuros estão dispostos no último capítulo.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os fundamentos essenciais para o desenvolvimento do presente trabalho. As seções aqui apresentadas visam oferecer uma profunda compreensão dos principais temas abordados no presente trabalho.

2.1 O PENSAMENTO COMPUTACIONAL

A seção apresenta as definições do tema, argumentos dos principais pesquisadores e realiza um sublinhado das competências promovidas em contexto educacional.

2.1.1 As definições do pensamento computacional

O termo pensamento computacional (*Computational Thinking*) foi inicialmente introduzido pela pesquisadora Jeannette M. Wing, na época presidente do departamento de Ciência da Computação da Universidade *Carnegie Mellon*. Segundo Wing (2006) o pensamento computacional envolve a solução de problemas, a projeção de sistemas e a compreensão do comportamento humano, utilizando como base conceitos fundamentais da Ciência da Computação.

Anos mais tarde ela concede uma nova definição delimitadora, na qual salienta que o pensamento computacional trata-se de um modo de pensar analítico, isto é, objetiva decompor uma informação em partes menores facilitando assim o entendimento como um todo. Além disso, o pensamento computacional compartilha com o pensamento matemático o aspecto da resolução de problemas, com o pensamento da Engenharia na formulação e avaliação de sistemas grandes e complexos que atuam dentro das limitações do mundo real, e por fim, compartilha do pensamento científico porque requer a compreensão da computabilidade, da inteligência, da mente e do comportamento humano (WING, 2008).

Trata-se de uma atividade mental que utiliza conceitos pertencentes à computação na resolução/elaboração de problemas do cotidiano de forma eficiente. Dentre os principais conceitos largamente utilizados que fazem parte da Ciência da Computação e estão presentes no pensamento computacional temos a abstração e decomposição, que respectivamente, são adotadas no ato de isolar aspectos relevantes e na tarefa de divisão em módulos do problema. Ambos conceitos constituem habilidades muito importantes dentro desse modo de pensar.

O pesquisador Paulo Blikstein compreende o pensamento computacional como:

> Não se trata, por exemplo, de saber navegar na internet, enviar email, publicar um blog, ou operar um processador de texto. Pensamento computacional é saber usar o computador como um instrumento de aumento do poder cognitivo e operacional humano – em outras palavras, usar computadores, e redes de computadores, para aumentar nossa produtividade, inventividade, e criatividade (BLIKSTEIN, 2008).

O autor enfatiza que para este mundo de hoje não cabe somente o aprendizado e domínio de ações básicas como ler, escrever e realizar operações matemáticas. Ele afirma que, atualmente a lista de habilidades que devem ser consideradas para o exercício da cidadania neste século é extensa, e o pensamento computacional é um importante item incluso nesta listagem que requer maior destaque.

A influência do pensamento computacional em determinadas áreas científicas como Biologia, Estatística, Engenharia, Inteligência artificial e Física, contribui para avanços que culminou na modificação da forma de pensar dos cientistas (Wing, 2006). Ela argumenta que brevemente esse tipo de pensamento deixará de fazer parte exclusiva do âmbito das pesquisas e será cada vez mais presente da vida de todos. Devendo desta forma, ser introduzido desde cedo na vida das pessoas, assim como ocorre com a leitura, a escrita e a aritmética.

Blikstein em consonância com os argumentos de Wing alega que o uso do computador como ferramenta profissional para estudo e criação de modelos está exigindo tarefas cognitivas que vão além do letramento digital. Em sua opinião as importantes habilidades compreendem:

> A habilidade de transformar teorias e hipóteses em modelos e programas de computador, executá-los, depurá-los, e utilizá-los para redesenhar processos produtivos, realizar pesquisas científicas ou mesmo otimizar rotinas pessoais, é uma das mais importantes habilidades para os cidadãos do século XXI. E, curiosamente, é uma habilidade que nos faz mais humano (BLIKSTEIN, 2008).

Ele defende que é fundamental para profissionais da ciência, engenharia e economia, e brevemente demais áreas saber pensar computacionalmente para propor novas soluções através do computador de forma produtiva e criativa.

Ao longo dos anos, diversos pesquisadores também propuseram definições próprias a respeito do pensamento computacional, como também enunciaram a

importância da sua inserção na formação dos jovens da geração atual e vindouras. Segundo Philips (2008) “a essência do pensamento computacional está em pensar sobre dados e idéias, e usar a combinação desses recursos para resolver os problemas”. O autor ainda ressalta que seu uso em contexto educacional permite aos alunos o desenvolvimento de associações cientificas para propor explicações e prever eventos, que serão úteis no entendimento de um objeto estudado. Nunes (2011) declara que trata-se de um processo cognitivo que organiza por meio de um conjunto de passos a resolução de problemas, que pode ser aplicado em várias áreas.

A fim de discutir a natureza do pensamento computacional, o conselho nacional de pesquisa dos Estados Unidos realizou em 2009 o *Workshop on The Scope and Nature of Computational* (NRC10, 2009), que contou com participação de estudiosos do tema. No relatório produzido após esse encontro, Bill Wulf demonstra compreende-lo como centrado em processos e em fenômenos de abstração. Gerard Sussman o define como uma forma de formular métodos mais precisamente para fazer coisas. Edward Fox enfatiza que seu núcleo está em permitir através da manipulação da abstração a solução de problemas.

Embora tenha havido um consenso entre grande parte de estudiosos a respeito do seu valor para atividades humanas, ainda faltava um senso comum, uma definição norteadora que servisse como base para aplicações no âmbito da educação.

Em 2009, a agência governamental dos Estados Unidos (*National Science Foundation*) financiou um projeto dirigido em conjunto pela Sociedade Internacional de Tecnologia na Educação (*ISTE*) e a Associação de Professores de Ciência da Computação (*CSTA*). O projeto destinava-se a tornar o conceito de pensamento computacional acessível aos educadores, fornecendo assim, uma definição mais clara e comum a todos, além de conceder modelos de práticas de ensino do pensamento computacional em sala de aula de acordo com a faixa etária dos alunos e objetivos educacionais. Deste modo, conforme mencionado no artigo de Valerie Bar e Chris Stephenson, sua definição ficou caracterizada como:

> Pensamento computacional é uma abordagem para a resolução de problemas de forma que pode ser implementado por um computador. Os estudantes não se tornam meramente ferramentas de uso, mas construtores de ferramentas. Eles usam um conjunto de conceitos, tais como abstração, recursividade e iteração, para processar e analisar dados, e para criar os artefatos reais e virtuais. Pensamento computacional é uma metodologia para resolução de problemas que podem ser automatizados, transferidos e aplicados entre indivíduos (BARR; STEPHENSON, 2011, p. 51).

Todavia, como esclarecem os autores, este projeto não se propunha a determinar de uma vez por todas a definição do pensamento computacional, mas sim, que se delimitasse sua significação perante os membros da comunidade educacional, tendo como enfoque disseminar sua utilização na educação básica.

2.1.2 Habilidades e competências do pensamento computacional

Primeiramente é preciso compreender o significado do termo habilidade, dado que tal palavra é comumente empregada por estudiosos para destacar benefícios resultantes ao uso do pensamento computacional. Faz-se necessário também entender o sentido da palavra competência, termo por vezes mencionado realçando as exigências atuais no mercado de trabalho e as relações dinâmicas do mundo atual.

No artigo intitulado Habilidades e competências na prática docente: perspectivas a partir de situações-problema, as autoras Gabriele Bonotto Silva e Vera Lucia Felicetti discutem a luz de autores e documentos o significa desses dois termos no sentido do âmbito educacional.

Segundo o dicionário a palavra habilidade significa “qualidade daquele que é hábil” (FERREIRA, 2001, p. 387), o que implica que o indivíduo tenha a capacidade de agir com destreza nos trabalhos realizados. Após consultarem o significado da palavra, as autoras fazem relação com definição proposta por Perrenoud:

> Essa conceituação vai ao encontro do que Perrenoud (1999) escreve, pois para ele quando o sujeito passa a mobilizar conhecimentos e capacidades, para resolver uma situação-problema da vida real, sem ao menos pensar ou planejar, então ele está utilizando a habilidade. Para Perrenoud (1999), habilidade trata-se de uma sequência de modos operatórios, de induções e deduções, onde são utilizados esquemas de alto nível (PERRENOUD, 1999 apud BONOTTO; FELICETTI, 2014, p. 19).

Desta forma podemos compreender a princípio a habilidade como uma capacidade individual de resolver problemas utilizando conhecimentos previamente estabelecidos, na qual a partir deles podemos então realizar induções e deduções como forma de resolvê-los.

Competência é um termo que segundo o dicionário significa “faculdade para apreciar e resolver qualquer assunto” (FERREIRA, 2016). Competência é a capacidade de utilizar mais de um recurso para resolver algo de forma inovadora, criativa e no momento necessário (GARCIA, 2005 apud BONOTTO; FELICETTI, 2014, p. 19).

Acerca do significado da palavra competência as autoras ainda destacam que:

> Os autores afirmam que competência é a existência de estruturas cognoscitivas que permitem a ação. Tal habilidade é usada para resolver uma situação real e complexa de forma eficaz, rápida e criativa. Nesse sentido, é necessário articular conhecimentos, valores e atitudes de forma integrada. Observa-se que os autores mencionam que habilidades e atitudes estão vinculadas a competências, uma vez que elas precisam ser inter-relacionadas com conhecimentos para que haja uma atuação competente (ZABALA; ARNAU, 2010 apud BONOTTO; FELICETTI, 2014, p. 19).

Desta forma é possível compreender que competência é algo mais abrangente no qual há um conjunto de habilidades e atitudes vinculadas na realização de uma ação. A competência ainda é caracterizada pela forma eficaz, rápida e criativa de resolver situações reais e complexas.

2.1.3 O pensamento computacional no contexto escolar

Conforme Iste (2011, p. 13), os resultados do projeto destinado a definir o pensamento computacional foram resumidos e sintetizados em uma "definição operacional", ou seja, uma descrição dos componentes que os educadores podem usar para desenvolver habilidades nos alunos em todo o currículo escolar da educação básica. Desta forma, as habilidades do pensamento computacional que poderão ser desenvolvidas a partir do seu ensino são caracterizadas como:

> - Formular problemas de uma forma que nos permita usar um computador e outras ferramentas para ajudar a resolvê-los;
> - Organizar e analisar dados logicamente;
> - Representar dados através de abstrações tais como modelos e simulações;
> - Soluções de automação através do pensamento algorítmico (uma série de passos ordenados);
> - Identificar, analisar e implementar as soluções possíveis com o objetivo de alcançar a forma mais eficiente e eficaz de combinar etapas e recursos;
> - Generalizar e transferir um processo de solução de um determinado problema para uma ampla variedade de problemas.

Como destaca os autores, estas habilidades são suportadas e aprimoradas por um conjunto de disposições ou atitudes que são essenciais para a dimensão do pensamento computacional. Tais disposições ou atitudes conforme (ISTE, 2011, p. 13) incluem:

> - Confiança em lidar com a complexidade;
> - Persistência ao trabalhar com problemas difíceis;
> - Tolerância em lidar com ambiguidade;

- Capacidade de lidar com problemas em aberto;
- Capacidade de se comunicar e trabalhar em grupo para atingir um objetivo ou solução em comum.

2.1.4 O alicerce das habilidades do pensamento computacional

As habilidades do pensamento computacional caracterizadas acima estão diretamente relacionadas ao emprego de nove conceitos advindos principalmente da área da computação. Os pesquisadores Barr e Stephenson (2011) reúnem áreas como Ciência da Computação, Matemática, Ciência, Estudos Sociais e Linguagem da Arte para demonstrar a relação existente entre todas quanto ao uso destes conceitos e capacidades do pensamento computacional. Logo abaixo, são destacados tais conceitos e as relações entre Computação e Matemática:

I. **Coleta de dados** é a capacidade de reunir informações relevantes acerca de um determinado objeto. Este conceito na Ciência da Computação refere-se à atividade de encontrar a fonte de dados para a área do problema. No campo da Matemática possui o mesmo significado e é exemplificado como lançar moedas ou jogar dados;

II. **Análise de dados** é uma capacidade na qual se examina o objeto estudado com a finalidade de obter algum resultado. Também está incluído a habilidade de reconhecer padrões e fazer generalizações. Na Ciência da Computação este conceito significa escrever programas para realizar cálculos estatísticos básicos em um conjunto de dados. No campo da matemática este conceito é empregado na contagem de ocorrências de lances, no ato de jogar dados e analisar seus resultados;

III. **Representação de dados** é a capacidade de representar dados obtidos. Na Computação esse conceito é empregado no uso de estrutura de dados, como matriz, listas encadeadas, pilha, fila, tabelas hash e etc. Já na matemática esse conceito é empregado no uso de histograma, gráfico de pizza, gráfico de barras para representar dados; usar conjuntos, lista e gráficos;

IV. **Decomposição** é a habilidade de desmembrar em partes algo até então tido como único. Está prática visa facilitar a compreensão do objeto estudado, analisando suas partes separadamente para então entende-lo como um todo.

Na Ciência da computação esse conceito pertence a tarefa de definir objetos e métodos, consiste em definir o que seria considerado a estrutura principal do programa de computador e o que seriam suas funções, partes menores que são separadas da estrutura principal. Na matemática esse conceito é utilizado na prática de aplicar a ordem das operações em uma expressão;

V. **Abstração** é a habilidade de abstrair, e está relacionada ao ato mental de se concentrar nos aspectos mais importantes de um determinado elemento e ignorar seus aspectos menos relevantes. Essa separação visa facilitar a compreensão e representação da essência do elemento analisado. Na computação este conceito é bastante empregado no ato de usar procedimento para encapsular um conjunto de comandos muitas vezes repetidos que executam uma função. Na matemática é utilizar variáveis em álgebra, identificar fatos essenciais do texto do problema, estudar funções sem álgebra comparando com funções na programação. Usar iteração para resolver problemas de palavras;

VI. **Algoritmo e Procedimentos** é uma forma de desempenhar uma sequência de passos/atividades com vista a alcançar um objetivo específico. Procedimento está ligado a um modo de agir, uma forma de realização de uma série de etapas. Na computação estes conceitos derivam do ato de estudo de algoritmos clássicos; implementar um algoritmo para uma área de um problema. Na Matemática consiste em realizar divisões longas, fatoração e prova dos nove;

VII. **Automação** propriamente dita é o ato de automatizar uma atividade. Na matemática é empregado no uso de ferramentas tais para construção automática de modelos;

VIII. **Paralelização** é a capacidade de realizar atividades em paralelo, no qual duas ou mais ações são empenhadas em atividades, com isso poupando tempo. Na computação, essa capacidade é empregada na divisão de dados ou tarefas em paralelo para serem processadas simultaneamente ("*threading","pipelining*"). Na matemática é utilizado para resolver sistemas lineares, como fazer multiplicação de matrizes;

IX. **Simulação** é a capacidade de tentar reproduzir algo pertencente ao mundo real o mais próximo possível e a testar possíveis respostas ao problema. Na computação podem ser empregados na animação de algoritmos para melhor compressão dos seus resultados. Na matemática, em gráfico de uma função no plano cartesiano e na modificação de valores de variáveis.

2.2 A PROGRAMAÇÃO INTRODUTÓRIA

Nesta seção são destacadas as características do tema, como também algumas das principais ferramentas representantes desse modo de aprendizagem.

2.2.1 Programação para iniciantes e seus desafios

O aprendizado de programação pode ser considerado para alguns alunos iniciantes como uma tarefa complexa. Segundo Kelleher e Pausch (2005), no ato de programar, há uma série de desafios impostos aos iniciantes. O primeiro obstáculo surge de fato na construção da solução planejada de maneira estruturada. Além disso, no ato da construção do programa, o aluno precisa constantemente entender o funcionamento do mesmo nos mínimos detalhes.

Outro desafio está na compreensão da sintaxe rígida pertencente a diversas linguagens de programação. Conforme Scaico *et al. (2013)* também mencionam, os obstáculos partem da sintaxe das linguagens de programação:

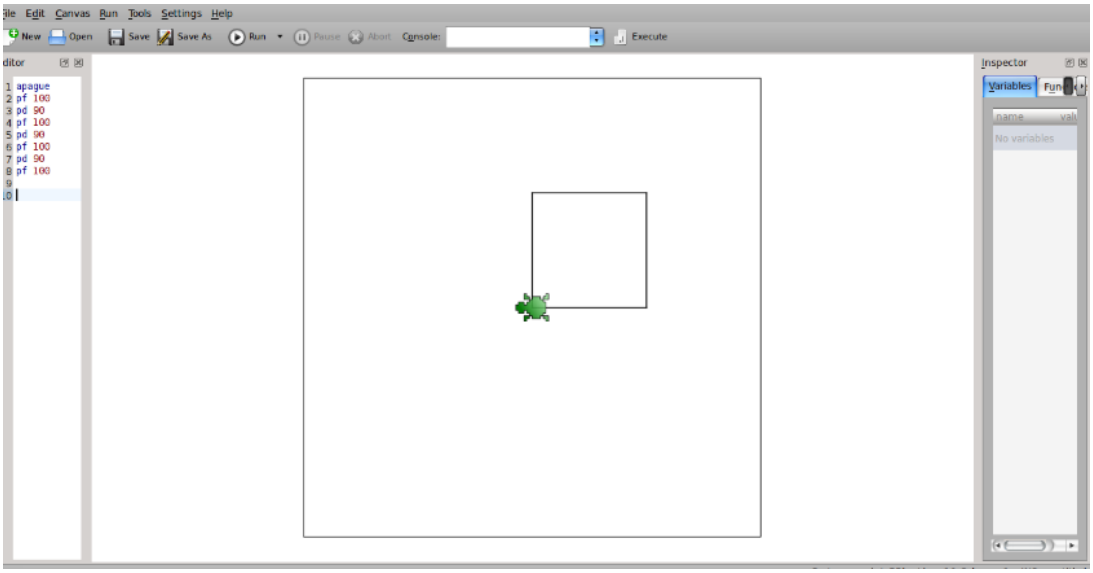
> A principal dificuldade encontrada pelos programadores iniciantes é a questão da sintaxe, pois eles precisam compreender a linguagem que o computador entende e também o idioma, já que nem todas as sintaxes das linguagens de programação possuem a clareza suficiente para que um novato entenda a funcionalidade e o uso dos seus comandos (SCAICO *et al.*, 2013, p. 95).

Tendo em vista tal problemática, pesquisas foram iniciadas no sentido de prover melhorias nos processos de ensino de programação. “Desde o início de 1960, os pesquisadores construíram uma série de linguagens e ambientes de programação com a intenção de fazê-la acessível a um maior número de pessoas” (KELLEHER; PAUSCH, 2005).

2.2.2 A linguagem Logo e a abordagem construcionista

Em meados dos anos 60 surge a linguagem de programação Logo, desenvolvida dentro dos laboratórios do MIT (*Massachussetts Institute of Technology*), cujo principal pesquisador e desenvolvedor do projeto foi o professor Seymour Papert. Com propósitos educacionais, o ambiente Logo permite a abordagem de aprendizado construcionista, na qual o aluno ativamente constrói suas soluções enquanto ao professor cabe o papel de auxiliá-lo nesse processo de descobertas.

Figura 1 - O ambiente Logo



Fonte: informaticageo.wordpress.com (2011).

2.2.3 Programação introdutória e os ambientes de programação visual

Embora a linguagem Logo e o seu ambiente de desenvolvimento construcionista tenham propiciado um maior acesso e aprendizado da programação introdutória, tal modelo de ensino ainda baseava-se no paradigma tradicional quanto ao uso de pseudocódigos como recurso para a construção de algoritmos. Esse modo de criação de programas se mostrou para alguns alunos um tanto limitador, já que diferentemente dos demais muitos não acompanhavam o conteúdo com facilidade.

Recentemente, surgem novas propostas de ensino de programação introdutória. Tais iniciativas abandonam o uso do paradigma tradicional e se voltam a uma nova abordagem de ensino.

Um novo paradigma de ensino de programação visual se instaura e surgem ferramentas baseadas nessa proposta, conforme Silveira *et al*. destacam:

> A programação em blocos é uma técnica utilizada no desenvolvimento de programas através de elementos visuais interativos, tendo como seu público alvo pré adolescentes e adolescentes que estão cursando o ensino médio e não mantiveram contato com programação. Consiste em arrastar e ligar blocos, que quando unidos em uma sequência lógica produzem uma ação programada (SILVEIRA *et al.*, 2015, p. 2).

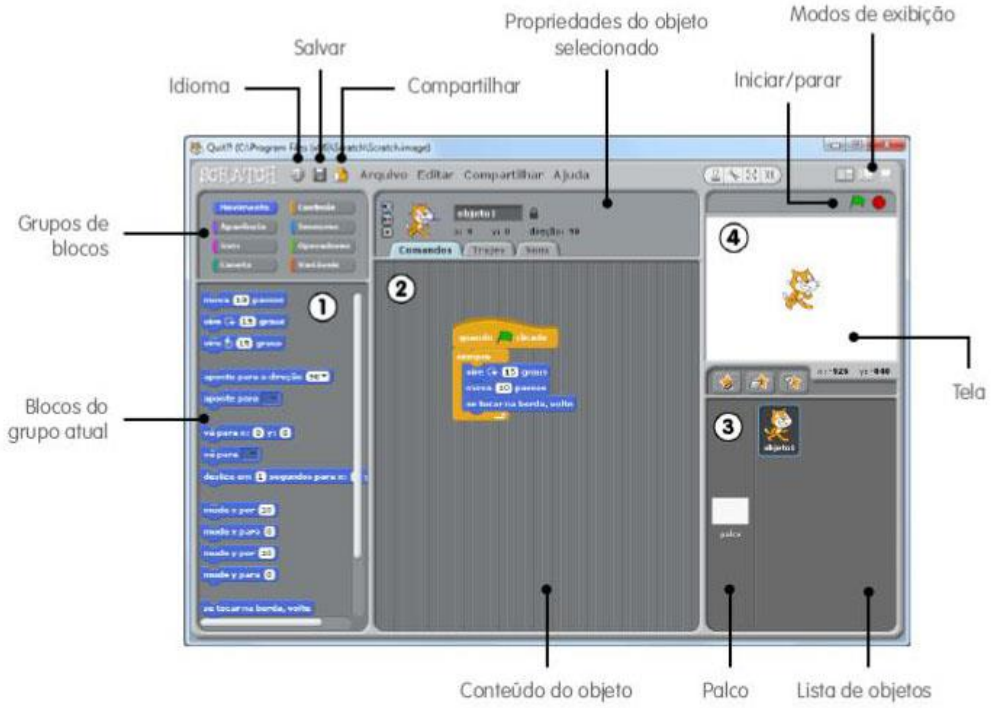
Na programação visual (*visual programming*) ou programação em blocos (*programming blocks*) o aluno programa através do empilhamento de blocos representativos, essa forma de interação entre ambiente e aluno permitem a reduzir a complexidade e problemas de aprendizados associados ao ensino de programação do modelo tradicional.

Fincher *et al.* (2010) em uma análise das principais características existentes nessas ferramentas de programação visual, denominou-as como Ambientes Introdutórios de Ensino. Segundo os pesquisadores todas em comum obedecem ao propósito de promover o engajamento imediato do ato de programar utilizando de recursos lúdicos que, de forma atraente, motivam o aprendizado dos alunos de maneira divertida e estimulante. Esses ambientes ainda partilham do aspecto de funcionarem em ambiente online e contarem com o apoio de comunidades para publicação e compartilhamento de materiais.

2.2.4 O ambiente Scratch

O Scratch é um ambiente de programação visual que permite através de uma experiência interativa criar projetos compostos por mídias como estórias animadas, jogos, tutoriais e simulações. A programação neste ambiente se dá a partir da manipulação de blocos representativos que podem ser encaixados entre si, essa junção permite a execução do algoritmo. A principal vantagem deste e demais ambientes semelhantes é que aqui não há qualquer preocupação com a sintaxe por parte do aluno na construção da sua solução, cabendo-o apenas buscar compreender os conceitos pertencentes a linguagem e utilizá-los na resolução de um dado problema. Na figura 2 são destacados os elementos pertences a interface *Scratch*.

Figura 2 - A interface do Scratch



Fonte: www.labpc.com.br/scratchday (2012).

Entre os diversos elementos que compõem a interface do *Scratch* temos o palco (3), que é o espaço no qual são organizados os objetos que serão manipulados. O local pertencente as propriedades do objeto (2), no qual são agrupados os blocos para efetuar comandos que serão exibidos na tela (4).

O objetivo do ambiente é destacado por Maloney:

> O objetivo essencial do Scratch é introduzir a programação para aqueles sem experiência prévia em programação. Esse objetivo influenciou os aspectos de design da plataforma. Alguns são evidentes, tais como a escolha de blocos visuais, idioma, o layout da interface do usuário de janela única, e o conjunto mínimo de comandos (MALONEY *et al.*, 2010, p. 3).

Uma característica marcante presente no *Scratch* é evitar mensagens de erro que impossibilitem a execução do projeto. Ainda que existam erros na abordagem de resolução adotada pelo aluno, partes corretas do código serão executadas, cabendo ao aluno o papel de depurar o código, fazendo-o buscar o trecho com erro e repará-lo.

2.2.5 A plataforma online Code.org

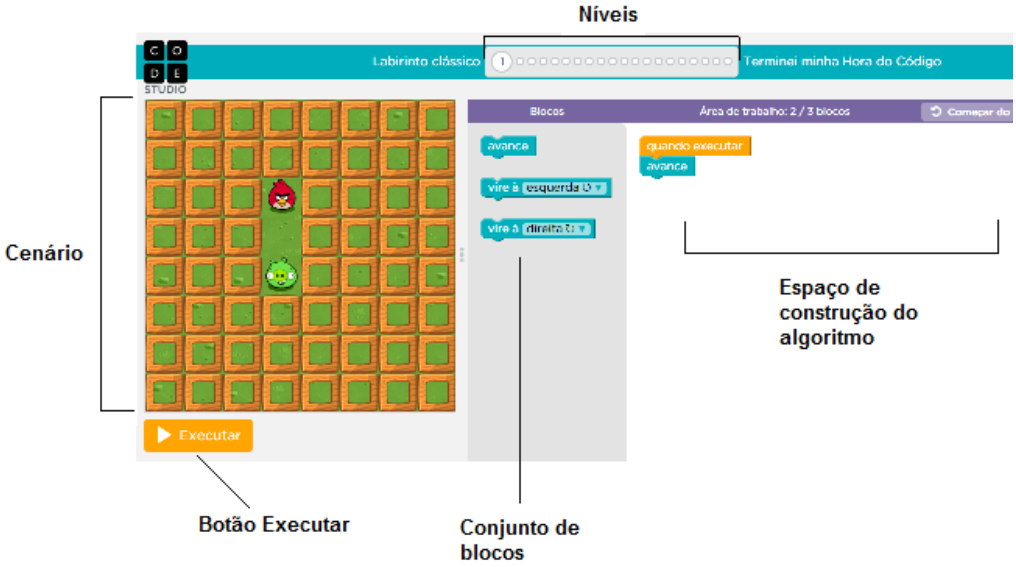
A plataforma de programação introdutória *Code.org* se assemelha em vários aspectos ao *Scratch* como a programação visual através da manipulação de blocos,

e também pela presença de uma interface intuitiva. O projeto iniciou em 2003 e conta com a colaboração de diversas empresas e organizações. A plataforma busca disseminar o ensino de Ciência da Computação nas escolas ao redor do mundo, o projeto visa desenvolver nos alunos as habilidades cognitivas dos cientistas da computação.

As pretensões se encontram na página do projeto, “Nós acreditamos que a ciência da computação e programação de computadores deve ser parte do currículo na educação, ao lado de outras ciências, tecnologia, engenharia e matemática, como biologia, física, química e álgebra” (CODE.ORG, 2016).

Na figura 3 é estacado os elementos da interface do *Code.org*.

Figura 3 - Etapa do Labirinto Clássico



Fonte: www.studio.code.org/hoc/1 (2016).

Os cursos oferecidos pela plataforma seguem um formato de espiral, em que conceitos e habilidades são revistos a cada ciclo. Cada curso é composto por níveis, a partir da progressão desses níveis novos conceitos são apresentados enquanto outros conceitos já apresentados vão sendo reforçados, essa tática visa reforçar o aprendizado dos usuários na plataforma.

O *Code.org* é voltado a **criatividade, colaboração, comunicação, persistência e resolução de problemas**. A presente interface como sua utilização carrega aspectos similares à dinâmica proveniente de jogos. Além disso, personagens conhecidos do público jovem e originários de jogos digitais como *Angry Birds* e *Plants VS Zumbies* entre outros provocam uma motivação inicial nos jovens.

As características da plataforma permitem o desenvolver de capacidades, conforme:

> A forma de ensino de programação da plataforma Code.org dá-se através de jogos, onde por junções de blocos de comandos, a criança irá desenvolver uma forma lógica de solucionar o que é pedido, desenvolvendo assim seu raciocínio lógico, como também a capacidade de superar problemas, de forma divertida e intuitiva, já que são usados personagens já conhecidos por elas (SOUSA *et al.*, 2015, p. 3).

Esse caráter lúdico trazido dos jogos acaba por motivar a realização das tarefas pelos alunos na plataforma e consequentemente prover um aprendizado. Tarouco (2004), em seu trabalho cita que os jogos possuem essa característica que motiva e faz engajar o jogador e isso facilita o aprendizado e aumenta a capacidade de retenção de ensinamentos através do exercitar mental e intelectual.

2.3 AS COMPETÊNCIAS DO PENSAMENTO COMPUTACIONAL E A PROGRAMAÇÃO INTRODUTÓRIA

Interessados em obter estratégias para avaliar o desenvolvimento do pensamento computacional pelos jovens através do envolvimento com a programação introdutória, os pesquisadores Karen Brennan e Mitchel Resnick realizaram um estudo sobre as competências desenvolvidas pelos alunos em atividades que envolvem a criação de projetos de programação por meio da plataforma interativa Scratch. Como resultado desse estudo, os pesquisadores projetaram um *framework* acerca das competências do pensamento computacional. Esse *framework* serve de guia pedagógico para professores que queiram desenvolver o pensamento computacional nos jovens a partir do uso de ambientes de introdutórios de programação.

Parte da motivação se deve ao crescente surgimento de ferramentas de programação introdutória voltadas aos jovens. Tais ferramentas são caracterizadas como ambientes de programação visual, na qual permitem a criação de códigos através do empilhamento de blocos gráficos representativos. A plataforma *Scratch* é um ambiente interativo que proporciona a criação de jogos, animações e histórias. Essa plataforma foi escolhida por ofertar um contexto propiciador do desenvolvimento do pensamento computacional e por ser caracterizada como uma ferramenta baseada em design, o que significa que é voltada a concepção de projetos.

A partir da avaliação de projetos desenvolvidos por alguns jovens alunos, realização de entrevistas e observações, os pesquisadores definiram que pensamento

computacional no contexto de uso de ambientes interativos de programação introdutória como o *Scratch* envolvem três grandes dimensões:

> Nós desenvolvemos uma definição de pensamento computacional que envolve três dimensões chave: conceitos computacionais (os conceitos empregados pelos projetistas no momento de programar), práticas computacionais (as práticas desenvolvidas pelos projetistas para programar), e perspectivas computacionais (as perspectivas dos projetistas sobre a forma de ver o mundo a sua volta e a si mesmos) (BRENNAN; RESNICK, 2012, p. 3).

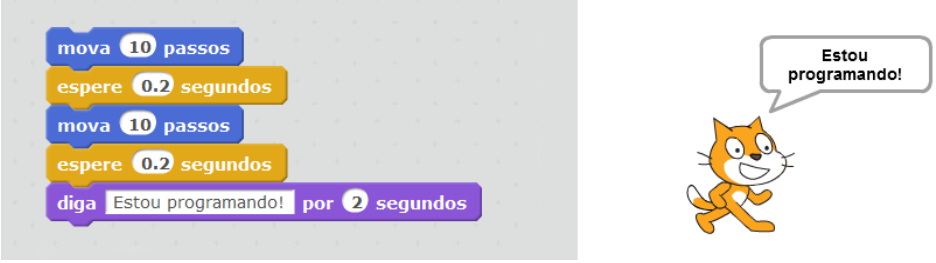
Os autores relatam que o uso do ambiente *Scratch* proporciona aos jovens um envolvimento com conceitos computacionais que também são comuns as demais linguagens de programação. “Foram identificados sete conceitos que são extremamente úteis em uma ampla gama de projetos no *Scratch*, e que são transferíveis para outros contextos de programação (e de não programação): sequências, loops, paralelismo, eventos, condicionais, operadores e dados” (BRENNAN; RESNICK, 2012, p. 3).

2.3.1 Os conceitos computacionais

A primeira grande dimensão estabelecida pelos autores refere-se aos conceitos computacionais empregados na construção de projetos de programação. Como os autores realçam, tais conceitos não se restringem ao universo da programação e podem ser aplicados em outros contextos. Os conceitos computacionais incluem:

Sequência é um conceito chave na programação, expressa uma série de procedimentos a serem realizados, esses procedimentos são acionados passo a passo, uma linha por vez. Como demonstrado na figura 4, os blocos demonstram a série de passos que serão realizados no *Scratch*, ao lado, o objeto gato realiza as ações a cada momento, primeiramente movendo 10 passos à esquerda, em seguida aguarda certo tempo, e por fim, o gato emite uma mensagem na tela.

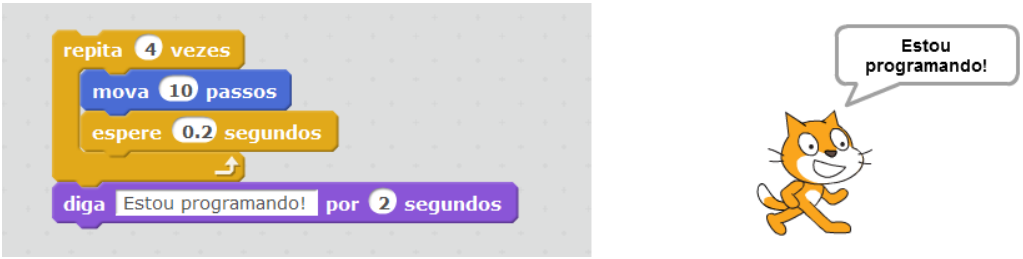
Figura 4 - Sequência de instruções no Scratch



Fonte: Elaborado pelo autor.

Loops na programação pode ser entendido como uma estrutura de repetição de instruções. Desta maneira um determinado passo ou um conjunto de passos serão realizados repetidamente de acordo com a quantidade de repetições estabelecidas. Este conceito é muito utilizado para reduzir e simplificar o código criado. Na figura 5 o objeto gato realiza as mesmas ações demonstradas na figura 1 repetidas quatro vezes.

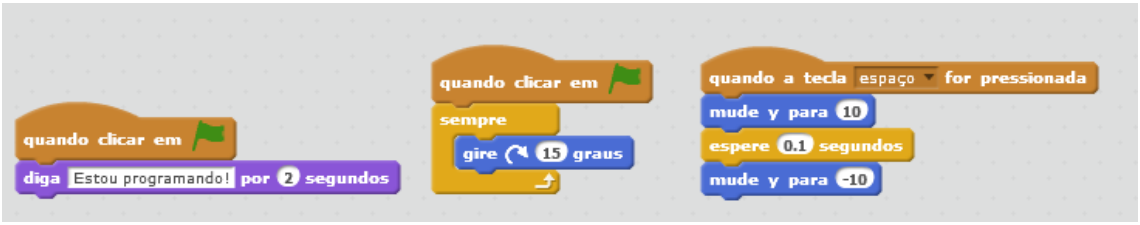
Figura 5 - Estrutura de repetição no Scratch



Fonte: Elaborado pelo autor.

Eventos são mecanismos programados para serem acionados a partir da ocorrência de uma situação especifica, quando um evento é despertado determinadas instruções do código serão desempenhadas. Na figura 6, o evento bandeira verde do *Scratch*, quando clicado, aciona duas instruções simultâneas. Também temos o evento que será despertado quando a tecla de espaço do teclado for pressionada.

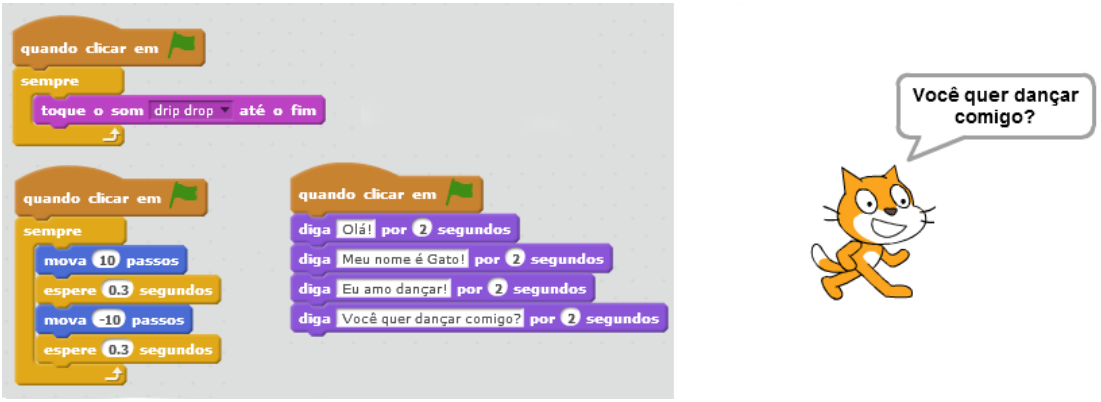
Figura 6 - Eventos e ações associadas no Scratch



Fonte: Elaborado pelo autor.

Paralelismo é empregado para executar sequência de instruções ao mesmo tempo em paralelo, minimizando assim o tempo empregado na realização de uma tarefa. Na figura 7, para fazer o gato dançar e falar, a bandeira verde aciona os três conjuntos de instruções simultaneamente. O primeiro conjunto é referente a execução de um áudio. O segundo conjunto faz o objeto gato mover-se. O terceiro conjunto realiza a exibição de mensagens na tela.

Figura 7 - Paralelismo sendo aplicado no Scratch



Fonte: Elaborado pelo autor.

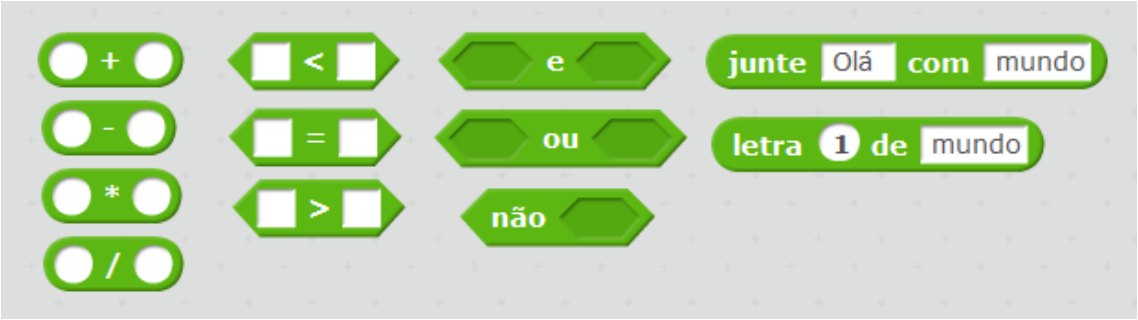
Condicionais: esse conceito é utilizado para permitir através de condições estabelecidas a execução de ações, sendo que essas ações podem ou não ocorrer. Para a execução deve se obedecer a uma condição estabelecida. No exemplo da figura 8, o objeto gato foi programado para sempre mover 20 passos enquanto emite uma mensagem. No momento em que a condição de tocar no objeto banana é verdadeira, o gato emite outra mensagem e para de mover-se.

Figura 8 - Instruções condicionais no Scratch

Fonte: Elaborado pelo autor.

Operadores são compostos por estruturas da matemática, da lógica booleana e expressões em caracteres. São utilizadas em operações matemáticas, em comparações de valores e na junção e exibição de caracteres.

Figura 9 - Operadores do Scratch



Fonte: Elaborado pelo autor.

Dados refere-se ao fato de armazenar, recuperar e atualizar valores que serão contidos em variáveis e *containers*. Variável é um espaço reservado na memória do computador para guardar valores. Um container é similar a uma variável, porém não se restringe a guardar apenas um valor por vez. O container é utilizado para alocar muitos valores na sua estrutura, tais valores podem ser incluídos ou removidos da sua estrutura a qualquer momento. No exemplo da figura 10 estão representadas as variáveis e *containers* utilizados para guardar valores e exibi-los.

Figura 10 - Variáveis e Containers no Scratch

Fonte: Elaborado pelo autor.

2.3.2 As práticas computacionais

As práticas computacionais são a segunda grande dimensão relatada pelos autores e foram evidenciadas a partir de observações realizadas nos projetos desenvolvidos pelos alunos. “Práticas computacionais estão concentradas nos processos de pensar e aprender, focando não somente naquilo que se está aprendendo, mas em como se está aprendendo” (BRENNAN; RESNICK, 2012, p. 7). As práticas computacionais envolvem quatro principais práticas: iterativo e incremental, teste e depuração, reuso e reformulação e abstração e modularização. E são caracterizadas como:

Iterativo e incremental a construção de projetos não é simples, requer um processo sequencial na qual primeiro se identifica os conceitos que serão utilizados, na sequência desenvolve-se um plano de execução, para enfim implementar o que fora planejado em forma de codificação. Em entrevistas com os alunos, eles descreveram que realizavam vários ciclos que envolviam primeiramente imaginar e depois construir. “É um processo adaptativo, no qual o plano pode ser alterado em resposta para se aproximar de uma solução em pequenos passos” (BRENNAN; RESNICK, 2012, p. 7).

Esta prática consiste em desenvolver parte do código, verificar a integração desta parte com o restante do projeto. Em casos de mal funcionamento, é preciso refazer a parte desenvolvida ou considerar novas abordagens. Em casos de sucesso surgem novas etapas. É um procedimento que ocorre aos poucos e é um processo que requer análise e refinamento constantes.

Teste e depuração em projetos de programação é obrigatória a presença de práticas que buscam realizar testes para identificar erros, essas práticas envolvem etapas de análise e reflexão detalhada do código, para descobrir a raiz de um problema (depuração).

Reuso e reformulação são conceitos comumente empregados em práticas de programação. “Um dos objetivos da comunidade online *Scratch* é apoiar os jovens projetistas na reutilização e reformulação, ajudando-os a encontrar idéias e códigos para construir em cima, permitindo que potencialmente criem coisas muito mais complexas do que poderiam criar sozinhas” (BRENNAN; RESNICK, 2012, p. 8).

O reuso está relacionado ao fato de utilizar trechos próprios de códigos desenvolvidos em trabalhos anteriores ou fazer uso de trechos que outras pessoas desenvolveram, ambas abordagens possuem a finalidade de reutilizar partes funcionais em novos projetos. Essa prática buscar aproveitar códigos existentes com o propósito de não gerar retrabalhos. Porém, nem sempre as adições de trechos de códigos se adéquam devidamente aos projetos, sendo necessário realizar a prática de reformulação, o que consiste em repensar e realizar mudanças no trecho baseado no contexto do novo projeto.

Abstração e modularização são duas práticas importantes nas atividades que envolvem programação. Abstrair significa buscar simplificar um problema dado como complexo. Essa prática consiste em desconsiderar uma série de informações irrelevantes para focar apenas nas informações importantes. Após essa separação

buscamos compreender o núcleo do elemento estudado e, o que de fato é de vital importância para o seu funcionamento ou representação. Já a modularização diz respeito ao fato de separar mentalmente, em pequenas partes, a estrutura compreendida do objeto estudado. Essa abordagem tem a finalidade de diminuir a complexidade existente ao se buscar entender um elemento como um todo. Essa prática é bastante difundida na área da Computação pela expressão “Dividir para conquistar”.

2.3.3 As perspectivas computacionais

As perspectivas computacionais enquadram a terceira grande dimensão observada pelos autores no estudo:

> Em nossas conversas com *Scratchers*, ouvimos jovens projetistas descreverem sua evolução no entendimento de si mesmos, suas relações com outras pessoas e o mundo tecnológico em torno de si. Esta foi uma dimensão surpreendente e fascinante de participação com *Scratch* – uma dimensão não capturada pelo nosso enquadramento de conceitos e práticas. Assim, como a etapa final em articular nosso framework do pensamento computacional, nós adicionamos a dimensão das perspectivas para descrever mudanças na perspectiva que observamos em jovens trabalhando com *Scratch* (BRENNAN; RESNICK, 2012, p. 10).

As perspectivas computacionais estão relacionadas as atitudes desempenhadas pelos alunos durante o contato com a programação. Essas atitudes são relativas ao modo de se expressar, se conectar, se questionar e o modo de agir colaborativamente.

As atitudes incorporadas às perspectivas computacionais incluem:

Se expressar: como aponta os autores, vivemos cada vez mais em contato com mídias interativas, porém, em nossas experiências de forma predominante nós nos restringimos ao consumir passivo. Segundo Brennan e Resnick (2012), a maior parte das nossas atividades de ensino envolvendo medias interativas são ótimas para a aquisição de conhecimento sobre o uso da tecnologia, o que de fato é bastante importante, mas não é suficiente para formar “pensadores computacionais”, pessoas que criam e expressam idéias a partir da utilização de computadores.

Se conectar: o *Scratch* é um ambiente que oferece interação através da sua comunidade online, no qual é possível compartilhar projetos com outras pessoas. Conforme (BRENNAN; RESNICK, 2012, p. 10) afirma: “Em entrevistas e observações, notamos uma grande variedade de formas em que a prática criativa individual foi

beneficiada pelo acesso aos outros, através de interações face a face ou (particularmente no caso da comunidade online do *Scratch*)".

O mais importante dessas perceptivas são seus valores associados e transmitidos aos jovens. Brennan e Resnick (2012) descrevem que os alunos em entrevistas realizadas admitiram que foram muito mais capazes de construir seus projetos auxiliados pelos outros colegas, do que se fossem realizando sozinhos. Isso se constitui como o primeiro valor destacado pelos autores, o valor em "*criar junto com outros"*, ocorria através da troca de informações entres os alunos na comunidade online do *Scratch*, por meio do estudo e reformulação de códigos de outros alunos disponíveis na plataforma online, e também no estabelecimento de parcerias e colaborações entre os alunos com a finalidade de desenvolverem os projetos juntos.

O segundo valor desenvolvido é o de "*criar para os outros"*. Os alunos descreveram que gostaram da forma como outros colegas se envolveram e apreciaram seus projetos disponibilizados na comunidade. Como destacam os pesquisadores, a natureza desses trabalhos era diversa, por vezes buscavam servir de *entretenimento* como jogos e histórias. Alguns trabalhos envolviam *pesquisas*, já outros objetivaram *equipar* outros projetos, compartilhando na plataforma trechos de seus códigos desenvolvidos com o intuito de ajudar os demais membros da comunidade. Outros tinham um propósito *educativo*, e forneciam variados tutoriais.

Se questionar compreende o fato de enxergar as tecnologias como potenciais ferramentas auxiliadoras. Trata-se de fazer questionamentos a respeito de como podemos utilizar computadores para propor soluções que nos afligem. Os autores exemplificam a partir do caso em que um grupo de usuários insatisfeitos com algumas limitações existentes da plataforma *Scratch* se uniram para desenvolver uma nova versão e disponibilizá-la para download. "Isto envolveu não só reconhecimento que o *Scratch* como um artefato projetado no mundo que pode ser modificado, mas também que eles, como projetistas de mídia computacional, foram habilitados para modificá-lo" (BRENNAN; RESNICK, 2012, p. 11).

3 TRABALHOS RELACIONADOS

No trabalho de Dantas e Costa (2013) é discutido o uso da plataforma Code.org como meio de incentivo a programação nas escolas e como tal prática construtiva de ensino pode oferecer benefícios à formação dos estudantes. Os autores puderam concluir que o ensino de programação em contexto escolar se apresenta como fator contributivo no processo de formação dos alunos e a abordagem adotada pela plataforma é eficiente nesse aspecto. Os autores ainda evidenciam sua importância por auxiliar no desenvolvimento de habilidades como: raciocínio lógico, conhecimentos matemáticos, trabalho em equipe, capacidade de resolver problemas e estímulo a criatividade.

O presente trabalho também apresenta o valor do ensino da programação introdutória para a formação dos jovens. Do mesmo modo, realiza um estudo sobre a potencialidade da plataforma Code.org em desenvolver certas capacidades. No entanto, o nosso trabalho aprofunda-se na investigação de habilidades e competências exclusivas de um modo de pensar específico, denominado por pensamento computacional.

França e Amaral (2013), apresentam uma proposta de ensino e avaliação do desenvolvimento do pensamento computacional em programação introdutória com Scratch. As autoras utilizaram um framework da literatura para avaliar o desenvolvimento de competências nos alunos através dos seus projetos desenvolvidos. A oficina de programação ocorrida em Pernambuco, contou com a participação de 24 estudantes e os resultados apontaram que os alunos desenvolveram certas competências relacionadas aos conceitos computacionais.

Nosso trabalho utiliza o mesmo framework que França e Amaral (2013), um guia proposto por Brennan e Resnick (2012) para nortear os conceitos, práticas e perspectivas computacionais no contexto de pensamento computacional e programação introdutória. Por outro lado, nosso trabalho não aborda Scratch, e sim um curso da plataforma code.org que apresenta desafios aos alunos e ao mesmo tempo estimula a aprendizagem de programação introdutória.

4 ESTUDO DE CASO NA PLATAFORMA CODE.ORG

A justificativa de escolha do objeto de estudo é apresentada na seção 4.1, na seção 4.2 o objeto de estudo é descrito, posteriormente a seção 4.3 faz uma recapitulação do documento da literatura utilizado como fundamento para essa pesquisa. Os limites estabelecidos ao estudo são apresentados na seção 4.4. Análise e resultados do estudo de caso são expostos na seção 4.5. E por fim, na seção 4.6 é apresentado um apanhado dos resultados obtidos com o estudo.

4.1 COMPARATIVO ENTRE AMBIENTES DE PROGRAMAÇÃO EM BLOCOS

A partir dos critérios apresentados na subseção 1.4.5 do presente documento foi realizado um comparativo entre duas ferramentas de programação introdutória. O comparativo entre os dois ambientes de programação em blocos mais populares, o Scratch e o Code.org têm como intuito demonstrar a razão pela escolha da segunda opção. O quadro abaixo reapresenta os critérios estabelecidos para o estudo.

Quadro 1 - Comparativo entre as plataformas

Nome	Ambiente gratuito	Suporte ao Idioma em Português	Direcionado ao público iniciante	Disponibilidade online	Conteúdo definido
Scratch	Abrange	Abrange	Abrange	Abrange	Não Abrange
Code.org	Abrange	Abrange	Abrange	Abrange	Abrange

Fonte: Elaborado pelo autor (2016).

Em relação aos três primeiros itens, ambas ferramentas preenchem esses critérios. O livre acesso à plataforma possibilita que resultados almejados pelo estudo possam servir como um simples norteador para atividades de desenvolvimento do pensamento computacional em qualquer instituição de ensino. Em outras palavras, o livre acesso facilita a propagação, a iniciativa de práticas de ensino de programação nas escolas, principalmente as da rede pública.

O suporte ao nosso idioma é essencial. Sem este recurso presente, nem todos os alunos conseguiriam compreender as atividades. A presença deste recurso de certa forma democratiza o direito de aprendizado entre todos os alunos.

Outro critério preenchido tanto pelo Scratch quanto o Code.org é o foco nos alunos iniciantes. As duas se propõem a ensinar os fundamentos da programação a

alunos sem experiência prévia em programação. Esse critério é de suma importância pelo fato do estudo investigar o desenvolvimento de habilidades e competências em estudantes que não tiveram nenhum contato anterior com programação.

O critério relativo ao acesso online, condiz com o ideal de acessibilidade, o que propicia que escolas possuidores do mínimo necessário (internet e dispositivos eletrônicos) possam usar a ferramenta sem a necessidade de instalação. Tanto o Code.org quanto a recente versão 2.0 do Scratch podem ser acessados online.

O Scratch apesar ser um ambiente de ensino de programação provedora de experiências ricas e dinâmicas não se enquadra no último critério. O estudo pretende fazer uma análise do conteúdo fixos no ambiente de programação, nesse caso o Code.org se mostra mais indicado. No Scratch os alunos são livres para criarem o que quiserem, já no Code.org as atividades de programação realizadas são fixadas a certos conteúdos. O que se pretende é a replicação da experiência de uso em outros contextos, principalmente no que tange a ambientes educacionais.

A inclinação natural para selecionar o Code.org como objeto de estudo decorre a princípio pela experiência pessoal do autor do TCC de ensino de programação. A partir da prática em sala pude perceber que a ferramenta baseada em programação em blocos desfruta de um potencial fortalecedor para a aprendizagem As atividades desempenhadas na plataforma permitiam interiorizar no aluno tudo o que fora transmitido em aula. O contato direto com a plataforma foi a centelha inicial, o impulso gerador dos questionamentos compostos nesse trabalho.

4.2 OS CURSOS DA PLATAFORMA

A plataforma Code.org possui um ambiente voltado exclusivamente aos educadores. Esse módulo provê ao professor os recursos de criação de turmas, gerenciamento de alunos e acompanhamento do progresso deles nas atividades exercidas. O destaque deste módulo é a disponibilidade de cursos exclusivos projetados para que professores os utilizem em sala de aula.

Os cursos são constituídos de uma série de etapas intituladas pela ordem em que se apresentam e pelo tipo de solução associada. A cada etapa, os alunos encaram um conjunto de desafios, que variam em complexidade. Para alcançar os objetivos é necessário o emprego de conceitos computacionais para a solução. Por vezes, apenas um conceito é necessário para atingir a resolução, em outros casos,

só é possível com a combinação de um grupo de conceitos. A medida em que os desafios vão sendo superados os alunos passam para as próximas etapas.

Atualmente estão disponíveis quatro cursos na plataforma. O primeiro curso é voltado aos alunos em fase da alfabetização. O segundo curso é dirigido aos alunos que já possuem a capacidade de interpretação de textos e estão cursando o ensino fundamental ou médio. Nos cursos 3 e 4 são aprofundados os conteúdos abordados no segundo curso, sendo que o nível de complexidade dos problemas aumenta à medida em que o aluno avança.

A figura 11 abaixo apresenta o resumo disponibilizado pela plataforma contendo o propósito dos cursos e recomendações a respeito do público destinado.

Figura 11 - Cursos da plataforma Code.org

Fonte: https://studio.code.org (2016).

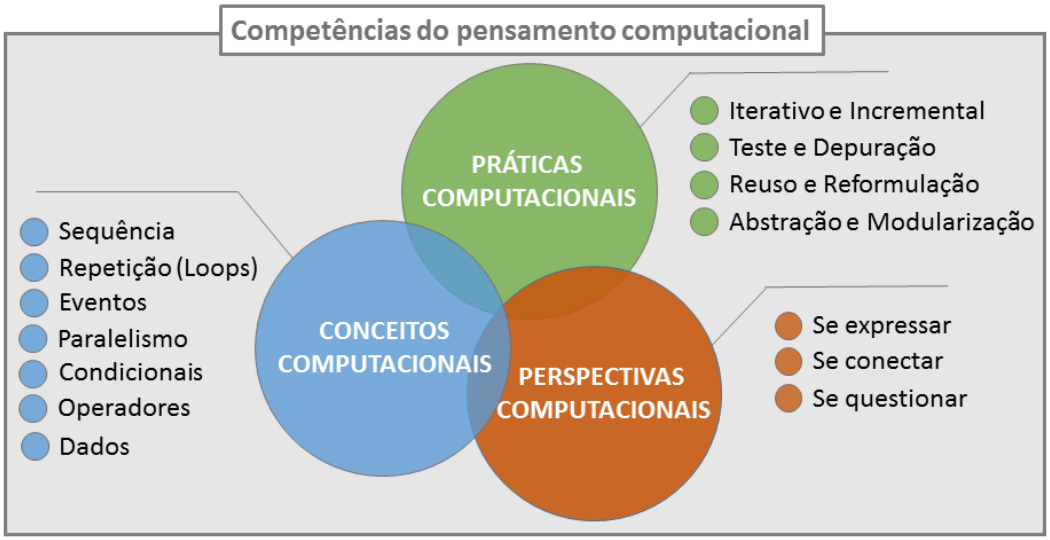
O curso 2 é composto por 19 etapas, das quais 12 são atividades online associadas à programação em blocos. As 7 etapas restantes são classificadas como atividades offline (atividades sem o uso da plataforma). Essas atividades utilizam a modalidade de ensino denominada Computação Desplugada, na qual a plataforma não é utilizada, nem o computador, haja vista que o objetivo da computação desplugada é o ensino-aprendizagem de conceitos computacionais de forma lúdica e divertida utilizando materiais de fácil acesso, como papeis, lápis, cartolinas entre outros, exceto o computador.

4.3 RECAPITULAÇÃO DO FRAMEWORK

O estudo do framework proposto por Brennan e Resnick (2012), apresentado na seção 2.3, na Fundamentação teórica deste trabalho, permite a compreensão da

expansão de habilidades e competências dos alunos relacionadas ao pensamento computacional através da utilização do Scratch como ferramenta de ensino de programação inicial. Com base no resultado desse estudo foi projetado um framework, um guia aos professores que concede a compreensão do desenvolvimento de competências relacionadas a conceitos, práticas e perspectivas computacionais. A figura 12 resume as competências exploradas em cada parte.

Figura 12 - Esquema representativo das competências do framework



Fonte: Elaborado pelo autor.

Este estudo de caso se fundamenta nesta proposta e visa analisar e identificar competências estabelecidas no modelo, mas tendo como foco o estudo na plataforma Code.org. Dessa forma, investigamos se o framework proposto por Brennan e Resnick (2012) é expansível a outras plataformas de programação ou limitada ao ambiente Scratch.

4.4 DELIMITAÇÕES DO OBJETO DE ESTUDO

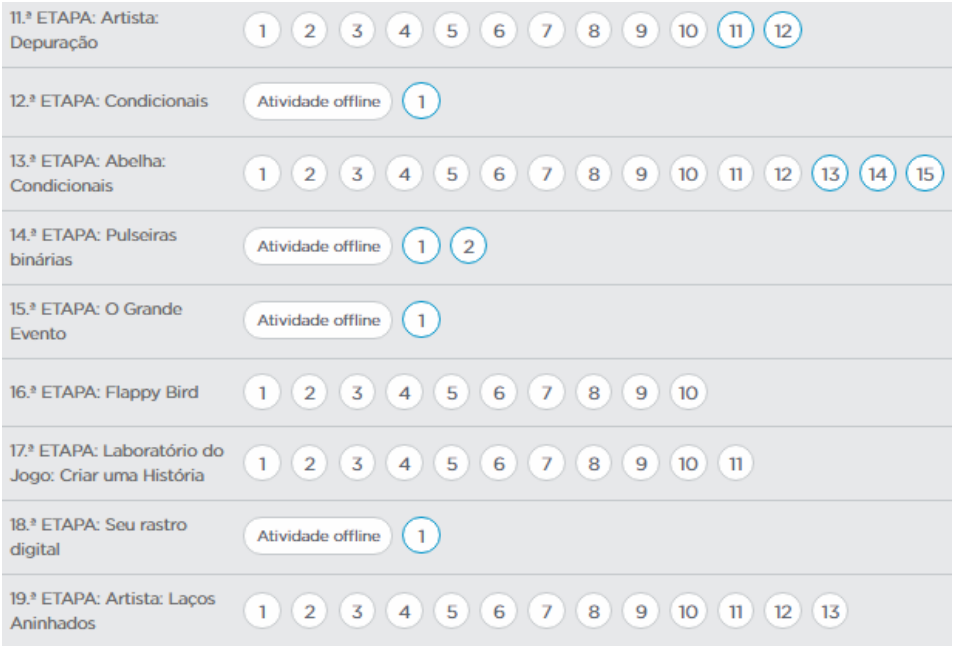
Neste estudo de caso nos restringimos às etapas do curso 2. A justificativa para a escolha é a sua abrangência de conceitos de programação trabalhados e por ser o curso mais indicado ao público composto pelas oficinas de Ensino de programação introdutória ofertadas, estudantes matriculados no Ensino Médio e sem experiência prévia com programação. O estudo de caso desconsiderou as etapas realizadas offline pelo fato de não utilizarem a plataforma para sua execução. Dessa forma, foram desconsideradas as etapas 1, 2, 5, 9, 12, 14, 15 e 18.

As figuras 13 e 14 apresentam o quadro de progresso presente na plataforma. O quadro exibe todas as etapas do curso, destacando as atividades offline. Os desafios das etapas online são representados pelos círculos brancos enumerados.

Figura 13 - Quadro de progressos do aluno no curso (parte 1)

Fonte: https://studio.code.org/s/course2/ (2016)

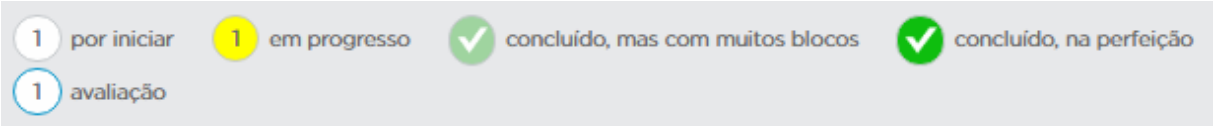
Figura 14 - Quadro de progresso do aluno no curso (parte 2)



Fonte: https://studio.code.org/s/course2/ (2016)

A progressão em cada etapa do curso é demarcada através de símbolos que transmitem como feedback ao estudante o seu desempenho nos desafios. A figura 15 apresenta as possíveis marcações obtidas durante e após a resolução dos desafios agregados às etapas do curso.

Figura 15 - Marcações que indicam o desempenho do estudante



Fonte: https://studio.code.org/s/course2/ (2016)

Há dois símbolos importantes na figura que transmitem o progresso do aluno nas etapas do curso. O símbolo “concluído, mas com muitos blocos” sinaliza que o desafio não foi solucionado de forma eficiente, isto é, o aluno utilizou mais blocos do que deveria. Já o símbolo “concluído, na perfeição”, contrariamente sinaliza que o aluno construiu sua solução dentro do mínimo de blocos permitidos.

4.5 RESULTADOS

Apresentamos o resultado da análise de cada etapa do curso 2 da plataforma code.org sob a ótica do framework de Brennan e Resnick (2012).

4.5.1 Os conceitos computacionais no Code.org

I. Sequencia

Em todas as etapas da plataforma a habilidade de elaborar procedimentos para a resolução de uma determinada tarefa é requerida aos alunos. Assim, percebemos que pensar em passos sequenciais é uma competência trabalhada em todas as etapas.

O conceito base para tornar tal feito possível é apresentado inicialmente na etapa 3 (Labirinto), uma vez que é nela que se iniciam as primeiras atividades online na plataforma. Nesta etapa inicial é lançado o desafio ao usuário de fazer o Red, o pássaro vermelho, chegar até o porco. Ao longo de todas as etapas e seus alusivos desafios, o modo de resolução segue o mesmo princípio, variando apenas (i) no aumento da distância percorrida, (ii) na maior presença de obstáculos e (iii) no acesso a novos blocos representativos.

Percebemos que a medida em que o usuário supera desafios e se depara com novas etapas são necessários novas ações cognitivas relativas a pensar linearmente, de forma algorítmica. Essas habilidades são requeridas para identificar os passos necessários para se atingir o objetivo e assim, reorganizar o conjunto de ações para tal feito. Nesse processo são adotadas estratégias relativas ao uso de blocos como (**virar à esquerda**), (**virar à direita**) e (**avance**), conforme exemplificado na figura 16.

Figura 16 - A construção sequencial em diferentes desafios

Fonte: https://studio.code.org/s/course2 (2016).

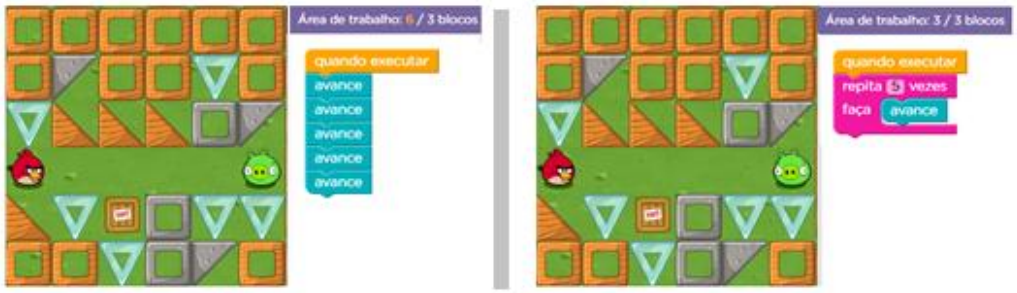
II. Repetição

Na etapa 6 (Labirinto: Ciclos) o conceito de repetição é exposto pela primeira vez. As consecutivas etapas 7 (Artista: Ciclos) e 8 (Abelha: Ciclos) reforçam ainda mais o seu uso. Após uma apresentação do significado da estrutura de repetição, os alunos são incentivados a utilizar um novo bloco na composição do seu algoritmo.

Os desafios são construídos de maneira a fazer o aluno identificar no problema proposto a existência de ações repetitivas. O bloco representativo (**repita n vezes**) é disponibilizado para que, com base na identificação e estimativa do número de repetições, o aluno possa utilizá-lo para vincular uma sequência de outros blocos repetidos que representam as ações identificadas. O bloco de repetição se encarregará de realizar as iterações de acordo com a quantidade estimada e informada pelo estudante.

A figura 17 exemplifica o processo de identificar as ações repetidas e estimar o número de realizações (**lado esquerdo**), levando o estudante a repensar a construção do seu algoritmo fazendo uso do bloco de repetição (**lado direito**).

Figura 17 - Reconstrução da solução com o uso do bloco de repetição



Fonte: https://studio.code.org/s/course2 (2016).

A dinâmica que envolve a resolução de problemas nessas três etapas não sofre grandes variações, salvo a (i) inclusão de novos personagens, (ii) aumento do grau de complexidade e (iii) vinculação de novos blocos representativos ao bloco de repetição. A figura 18 abaixo exemplifica essas variações.

Figura 18 - Uso do bloco repetição em diferentes desafios

Fonte: https://studio.code.org/s/course2 (2016).

Percorrer essas etapas é fundamental para que o estudante aprofunde seus conhecimentos sobre o conceito de repetição. Abordagens de programação considerando a ocorrência de ciclos repetitivos eventualmente serão requeridas aos alunos ao longo de etapas posteriores no curso.

Em síntese, ao avaliarmos as atividades de programação do curso denotamos o uso da estrutura de repetição, item pertencente aos conceitos computacionais. O emprego desse conceito nas abordagens realizadas pelos estudantes está relacionando a uma prática que identificamos denominada Reconhecimento de Padrões, prática que se encontra destacada na seção 4.2.2.

III. Eventos e Paralelismo

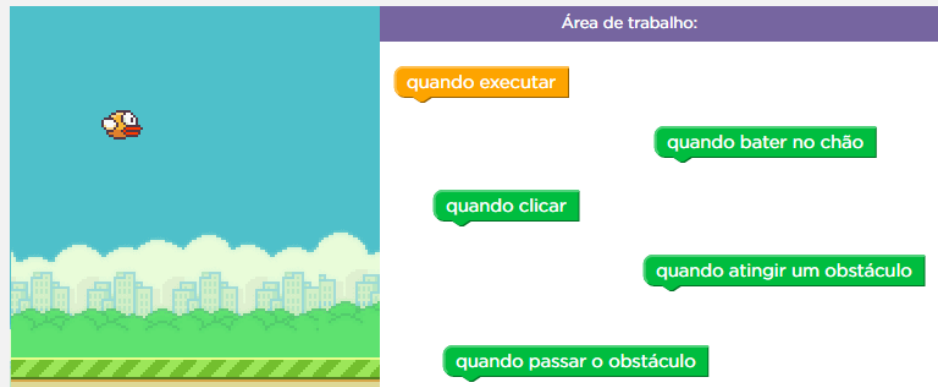
O uso de eventos como mecanismo auxiliar na resolução de problemas está presente nas etapas 16 (Flappy Bird) e 17 (Criar uma História), sendo que a presença

desse conceito é mais acentuada na etapa introdutiva. Neste momento do curso, o estudante deverá construir a solução idealizada empregando estruturas conceituais de denotam a ocorrência de eventos e as suas respectivas ações.

Os blocos representativos informam ao estudante qual tipo de evento especificado. Esses eventos, em maioria, fazem relação aos objetivos do desafio, que na etapa 16 é fazer o pássaro seguir sua trajetória. Os blocos dispostos no espaço reservado a construção do algoritmo (Área de trabalho) informam ao aluno quais eventos poderão vir a ocorrer.

A figura 19 apresenta os principais blocos de eventos presentes na etapa 16.

Figura 19 - Blocos eventos em um desafio da etapa 16

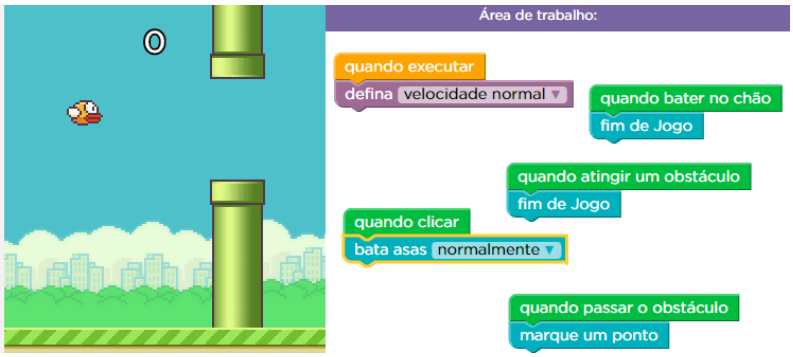


Fonte: https://studio.code.org/s/course2 (2016).

A disposição desse conjunto de blocos permite a identificação dos eventos associados a cada desafio. Isso auxilia o aluno a projetar mentalmente sua solução. Visto que, ao estudante cabe a atividade mental de projeta-la antevendo a ocorrência de determinados eventos e a medida em que os identifica estabelece ações específicas, ações interpretadas aqui como outros blocos representativos que podem ser acoplados a um bloco de evento específico.

A ocorrência do evento põe em ação seus blocos associados, que podem variar em quantidade. A figura 20 apresenta ações relacionadas a ocorrência de determinados eventos.

Figura 20 - Evento e suas ações na plataforma

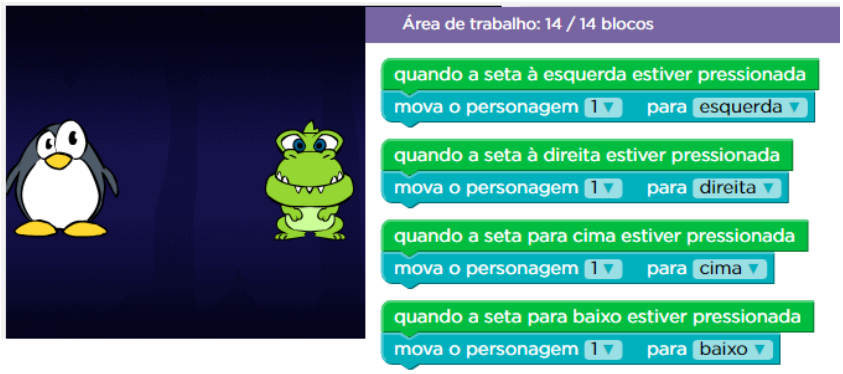


Fonte: https://studio.code.org/s/course2 (2016).

A utilização em conjunto de eventos pode ser considerada aqui como utilização do conceito de paralelismo, no qual o aluno percebe a necessidade de estabelecer mecanismo múltiplos para solucionar o problema numa dinâmica em paralelo. O paralelismo é encontrado nos desafios da etapa 16 e 17, onde é possível construir a solução através do uso de conjuntos de eventos e suas respectivas ações que acionadas simultaneamente possibilitam a conclusão do desafio proposto.

Na figura 21 é possível ver o uso simultâneo de eventos e suas ações agregadas.

Figura 21 - Uso de paralelismo na etapa 17



Fonte: https://studio.code.org/s/course2 (2016).

IV. Condicional

A estrutura condicional é exposta a princípio na 13° etapa (Abelhas Condicionais). Neste momento o estudante é provocado a refletir acerca de algumas circunstâncias envolvidas para a resolução do problema. A distinção existente entre essa etapa e as demais anteriores é que esse estágio concede uma nova provocação da maneira de pensar do aluno para se chegar à solução.

O aluno vinha até então desempenhando práticas relativas a projeção dos passos necessários para se chegar a solução, desconsiderando a ocorrência de circunstâncias adversas que impediriam tal feito. Em outras palavras, a resolução de problemas em etapas anteriores se caracteriza pela construção linear do algoritmo pelo aluno. Conforme o estudante avançava nos desafios, ele foi direcionando a projetar soluções baseadas em um único caminho a ser percorrido.

A partir da apresentação do conceito de condicional entra em cena a ocorrência de circunstâncias. Trata-se de situações ao qual não temos controle e não podemos prever o momento devido do seu acontecimento, apenas podemos estabelecer condições que tratem por ventura a ocorrência dessas situações, possibilitando chegar aos resultados pretendidos. Isso leva o aluno a tarefa cognitiva de pensar nos acontecimentos de determinadas situações e criar soluções para todas elas.

O conceito é introduzido propriamente no terceiro desafio, no qual o estudante deve construir um algoritmo que faça a abelha "*Bee*" coletar o néctar de todas as flores presentes e depositar a substancia colhida nos favos de mel. Porém, diferentemente dos desafios inicias, a abelha antes terá que verificar a existência da substância. Caso não haja, a abelha deverá seguir outro caminho.

Para realizar essa atividade, o aluno terá que utilizar o bloco condicional (**se "condição" faça**), onde o valor 'condição' é uma variável verificada em determinados momentos. Em casos em que a verificação seja verdadeira são acionados outros blocos acoplados a sua estrutura. A figura 22 demonstra o uso do bloco condicional em diferentes desafios.

Figura 22 - Uso do conceito de condicional durante os desafios

Fonte: https://studio.code.org/s/course2 (2016).

Ao longo dos desafios dessa etapa, o aluno deve implementar algoritmos que façam a abelhar colher o mel em momentos oportunos ou seguir sua trajetória. As

dinâmicas variam em (i) aumento do número de flores roxas, que representam o desconhecimento da presença do néctar, e portanto uso de blocos condicionais; (ii) presença de flores vermelhas entre as flores roxas. As flores vermelhas representam a certeza do néctar presente, o que faz o aluno repensar quais momentos deve incluir o bloco condicional em sua implementação.

V. Operadores e Dados

Embora os conceitos de operadores e dados estejam presentes em alguns desafios do curso 2, não identificamos nenhuma menção explicativa ao estudante sobre seus significados, diferentemente do que ocorre com outros conceitos computacionais encontrados.

A presença deles se caracteriza por não conceder ao aluno uma maior liberdade de uso, cabendo apenas a utilização de estruturas pré-determinadas. Isso pode restringir a criatividade do aluno e aprofundamento desses conceitos.

Os operadores identificados são qualificados pela exibição de caracteres, sendo que não foram encontrados os demais referentes a estrutura matemática e lógica booleana. A figura 23 demonstra a presença de blocos na etapa 17 que armazenam variáveis para posterior exibição de seus conteúdos na tela.

Figura 23 - Bloco utilizado para exibição de mensagens da tela

Fonte: https://studio.code.org/s/course2 (2016).

Acerca do conceito de dados, a presença de variáveis pôde ser identificada em praticamente todas as etapas. Porém o aluno, em maior parte das atividades, não realiza as operações características sobre elas como armazenar, recuperar e atualizar valores, em exceção nos casos de utilização de blocos de repetição, e em alguns blocos para expressar mensagens.

A figura 24 apresenta ao lado esquerdo o armazenamento de um valor informado pelo aluno em uma variável utilizada pelo bloco de repetição. Ao lado direito

a figura apresenta o uso de variáveis já determinadas pela plataforma, no qual apenas o podemos trocar seus valores.

Figura 24 - Uso de dados de forma restrita nos desafios

Fonte: https://studio.code.org/s/course2 (2016).

Portanto, podemos concluir que a dinâmica de manipulação de informações no presente curso se limita a simples troca de variáveis já pré-dispostas. Por consequência, há pouca liberdade para modificação de valores. Sendo assim, esses conceitos são utilizados pelo aluno durante a construção da solução, mas como o foco em cada etapa está em outro conceito eles ficam encobertos, passando despercebido pelo aluno.

4.5.2 As práticas computacionais no Code.org

I. Iterativo e incremental

As práticas computacionais iterativo e incremental se fazem presente por meio da abordagem de resolução dos desafios adotada pelo aluno. Quando o mesmo se depara com um novo desafio ainda em nível cognitivo, ele busca pensar a cada passo, a cada posição, qual bloco utilizar. Desta forma, o aluno constrói a solução ainda em sua cabeça de forma cíclica, seguindo passos sucessivos que o fazem aproximar gradativamente da solução idealizada. A medida em que torna isso real na plataforma, ou seja, faz o encaixe dos blocos e constata que ainda não concluiu o desafio, ou mesmo que há erros na resolução, novamente inicia-se o processo.

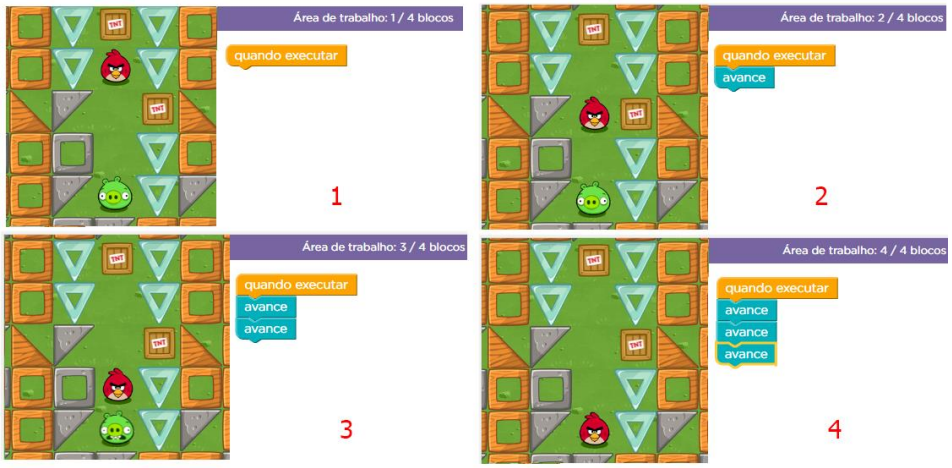
É importante ressaltar que ocorrem variações na abordagem adotada pelos estudantes. Em certos casos, alunos buscam construir toda a solução em suas cabeças para depois efetivamente torná-las reais com a adição dos blocos na plataforma. Em outros casos, os alunos buscam pensar apenas no passo em questão, depois de decidido, fazem a adição do bloco para então pensar no próximo passo.

Tais fatos foram observados durante a realização da oficina nos estudantes do ensino médio.

As aplicações dessas práticas sequem durante todo o curso, em todas as etapas online, em todos os desafios. Dessa forma, o exercício contínuo fortalesse essa habilidade no aluno que em outros contextos poderá a vir a adotar abordagens semelhantes para a resolução de problemas.

A figura 25 retrata a habilidade da construção da solução pelo aluno através da prática iterativa e incremental. Caracterizada por processos cíclicos de implementação o aluno a cada iteração realiza o incremento de parte do código e o executa, em caso de não alcance do objetivo o processo se repetirá até obter resultado desejado. Os números na figura transmitem os ciclos de atividades.

Figura 25 - As enumerações retratam a prática de iteração e incremento para se chegar na solução do desafio



Fonte: https://studio.code.org/s/course2 (2016).

II. Teste e depuração

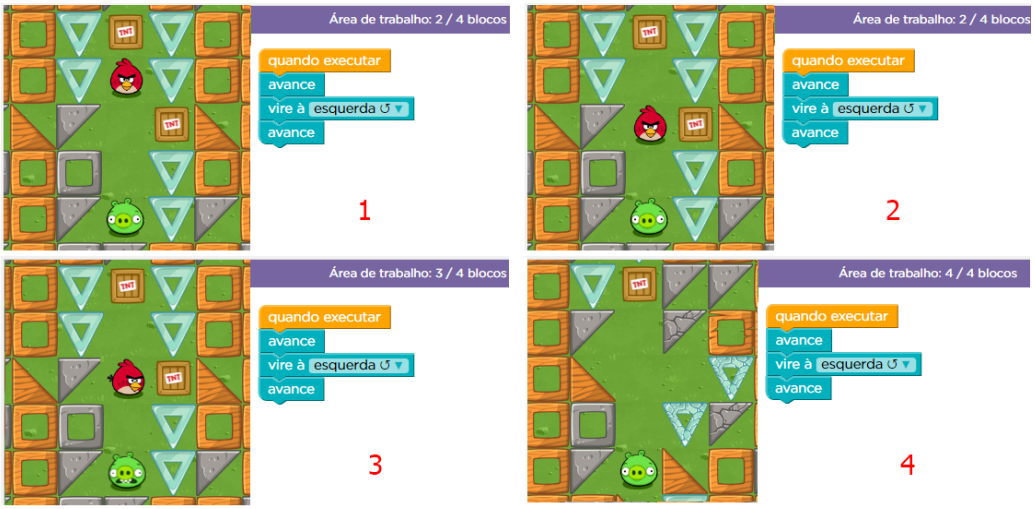
A prática de testar a resolução adotada pelo aluno é uma atividade fundamental presente no Code.org. Quando os alunos dão por concluída a implementação adotada, realizam a verificação através do acionamento do botão executar. Ao clicá-lo, uma animação é exibida. Esta animação segue exatamente a sequência dos blocos empilhados pelo aluno, reproduzindo suas representações. Caso a solução esteja correta o aluno conquista o desafio e passa para o próximo.

A animação possibilita a visualização real do que antes fora apenas uma idealização feita pelo aluno em sua mente (e no código). É através dessa possibilidade que o aluno pode acompanhar e verificar a ocorrência de erros. Em casos de falhas

na implementação, cabe ao aluno iniciar a prática de depurar seu código. Primeiramente, ele deve identificar o ponto, ou pontos, de ocorrências de tais falhas para então realizar as devidas modificações.

Na figura 26 é possível ver que após acionar o botão *executar*, uma sequência de animações é realizada. A existência de erros na implementação resulta em desvios e falhas que não levam ao objetivo do desafio. Na parte 3 da figura, Red vira a esquerda e é atingindo pela bomba de TNT e não consegue atingir seu objetivo: chegar ao porco verde. Nesse caso, o aluno é redirecionado a repetir a tarefa novamente.

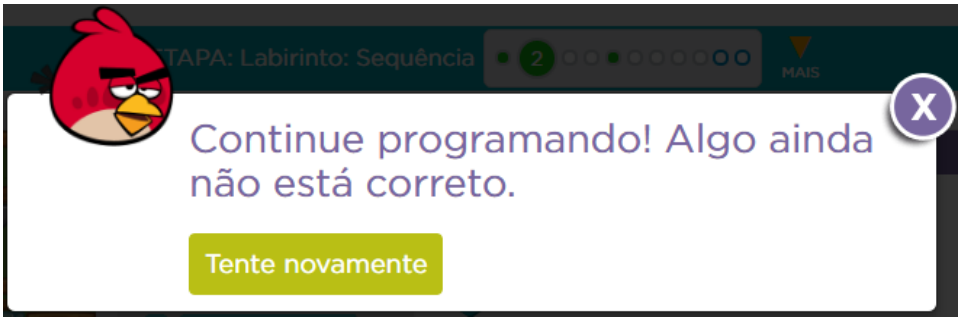
Figura 26 - As enumerações retratam a prática de teste no desafio



Fonte: https://studio.code.org/s/course2 (2016).

O alerta emitido pela plataforma quando há ocorrência de erros no código implementado pelo aluno é demonstrado na figura 27. Diante do indício de erro emitido, caberá ao aluno identificar o ponto originário do erro e repará-lo em seu código. A essa prática denominamos Depuração.

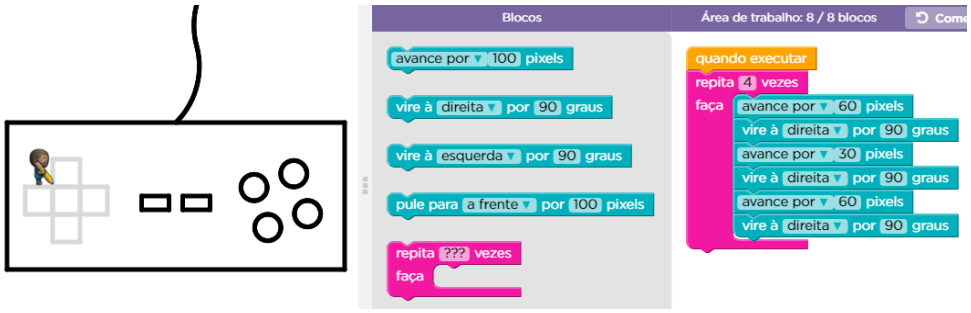
Figura 27 - Mensagem exibida na tela denotando a existência de erros na implementação do aluno



Fonte: https://studio.code.org/s/course2 (2016).

Há uma etapa exclusiva dedicada a realizar reparos em implementações. A etapa 11 (Artista: Depuração) consiste basicamente em realizar depurações nos códigos apresentados nos desafios. A figura 28 mostra um desafio característico nessa etapa. O objetivo aqui é fazer modificações no código para que o personagem possa contornar as linhas de forma correta.

Figura 28 - O objetivo na etapa 17 é depurar os códigos apresentados



Fonte: https://studio.code.org/s/course2 (2016).

III. Reuso e reformulação

Reuso e reformulação no sentido em que o trabalho de Brennan e Resnick (2012) tratam não foi identificado no curso 2, pois no sentido descrito no framework, reuso se refere a pegar trechos de códigos anteriores para aplicá-los em uma nova codificação. Isso não é possível no Code.org, visto que a cada desafio, os blocos têm que ser inseridos um a um. Uma exceção a isso é quando há blocos inseridos previamente, mas o objetivo da tarefa é realizar depuração, e não a construção linear da implementação.

Entretanto, apesar da plataforma Code.org não permitir copiar e colar códigos de desafios anteriores, em determinados momentos certos desafios se repetem. Isso permite ao aluno reutilizar a mesma abordagem mentalmente proposta para o desafio anterior. Em outras palavras, o aluno recorre a informações conservadas em sua mente sobre os passos necessários que permitiram a solução do desafio anterior. A partir do acesso a elas, o aluno retoma a implementação do novo desafio.

Conforme demonstrado na figura 29, os desafios 1 e 2 da 6° etapa se diferem apenas na forma de implementação adotada, prevalecendo o mesmo passos para alcançar os objetivos.

Figura 29 - O aluno pode praticar reuso a partir da lembrança da implementação do desafio anterior

Fonte: https://studio.code.org/s/course2 (2016).

Dessa forma a prática de reuso estaria associada a essa recorra de informações na memória. Essa releitura permitiria uma rápida implementação no desafio atual. Em outras palavras, o reconhecimento obtido por meio da memorização de passos anteriores que, de forma semelhante se aplicaria no novo contexto, possibilitaria uma rápida associação ao que seria necessário para se concluir o objetivo atual.

A prática de reformulação estaria presente nos momentos em que o aluno se depara com trechos que requerem nova forma de pensar a solução para conseguir o objetivo. Isso consistiria na reorganização, no qual o aluno teria que mensurar o que de fato continua semelhante e o que é inédito para a implementação.

Na figura 30 é demonstrado o que viria a ser uma prática de reformulação por parte do aluno. Em desafios consecutivos, o aluno se depara com implementações semelhantes. O reconhecimento de passos idênticos facilitaria a implementação da solução, enquanto que surgimento de trechos inéditos fariam o aluno pensar em novas estratégias de implementação.

Figura 30 - Prática de reformulação realizada para solucionar desafio semelhante

Fonte: https://studio.code.org/s/course2 (2016).

Dessa forma, a nível cognitivo, o aluno estaria fazendo o reuso quando reúne informações que se encaixam na implementação atual do desafio. A reformulação estaria na prática decorrente da ocorrência de novidades de demandariam o julgamento do que ser acrescido na implementação e o que se repetir da abordagem anterior.

IV. Abstração e modularização

Pode se considerar que nas atividades de programação do curso o aluno não realiza a tarefa de abstração por não simplificar problemas dados como complexos. Diferentemente, em outros contextos de programação que exigiriam o exercício da abstração, o aluno, para implementar sua solução, teria primeiramente que compreender o problema pertencente ao mundo real, e através de sua análise, extrairia desse problema apenas o essencial. Com base nessas informações, traduziria a solução em forma de programação.

Dessa forma a prática de abstração no seu sentido restrito não está presente nas atividades desempenhadas pelo aluno. Pois, não é ele que atua nos problemas do mundo real e os molda como implementação, isto é, não abstrai com vista a facilitar o entendimento do problema e realizar sua solução via codificação.

Já a prática de modularização está presente quando o aluno, em suas abordagens, “quebra” o problema em partes. Primeiramente por meio da observação geral do problema do desafio, o aluno cria pequenas metas que somadas permitem alcançar o resultado pretendido. Dessa forma, ele vai construindo seu algoritmo atento somente ao alcance da meta específica. Uma vez tendo sido alcançada, toma a próxima como meta principal, e assim sucessivamente até alcançar a conclusão definitiva.

Portanto, a modularização é uma prática recorrente para superar os desafios do curso. Já a abstração no sentido de trazer algo do mundo real, e extrair sua essência para passar para a programação de forma simplificada, não é praticado pelo aluno.

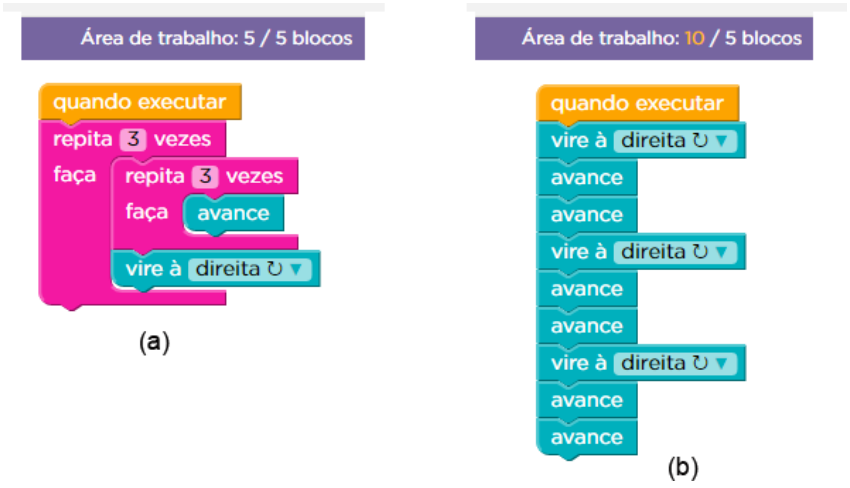
V. Eficiência

Observamos que em praticamente todos os desafios o aluno é sempre estimulado a construir suas soluções de maneira eficiente. Eficiência nesse contexto está no uso da quantidade mínima de blocos disponíveis para resolução do problema.

Em outras palavras, o estudante em sua atividade de programação deve construir a solução utilizando o mínimo de blocos permitidos.

Conforme ilustrado pela figura 31, durante o ato de programação, o aluno é informado quanto a eficiência de sua abordagem (limite de blocos para atingir a solução do desafio). No lado (a) da figura temos um exemplo de uso eficiente. Já no lado (b) temos um código não eficiente, onde há uma indicação de superação do limite.

Figura 31 - O limite de blocos indicados a cada desafio

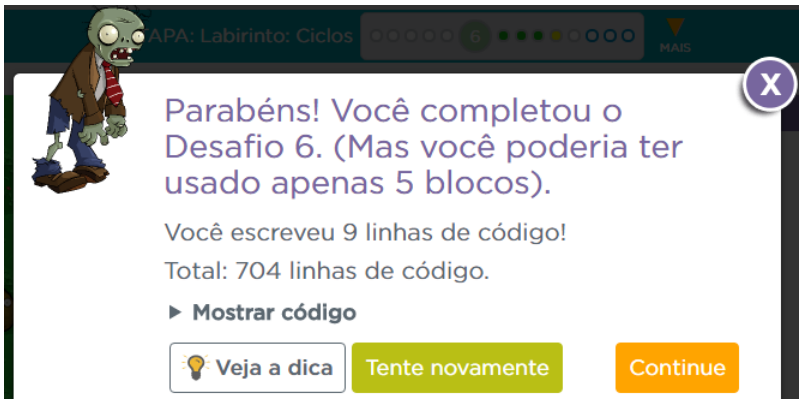


Fonte: https://studio.code.org/s/course2 (2016).

Essa imposição presente em cada desafio, mesmo que não obrigatória, possibilita ao estudante a estimulação de tarefas cognitivas associadas ao planejamento e execução de estratégia de solução. O que leva o aluno a raciocinar de forma a prever consequências de sua estratégia adotada.

Ao executar sua implementação, em casos de não eficiência, a plataforma emite uma notificação na tela solicitando a utilização de menos blocos. Isso não impede que o aluno prossiga para os próximos desafios, mas notifica-o que ele poderia ter usado menos blocos. A figura 32 mostra um exemplo de notificação.

Figura 32 - Notificação emitida na tela após a execução de um código não eficiente



Fonte: https://studio.code.org/s/course2 (2016).

Desse modo, a busca pela eficiência é uma prática importante presente no curso pelo fato de contribuir para que o aluno planeje melhor a construção de sua abordagem de resolução do problema. Isso o que acarreta códigos pequenos e eficazes, além disso denota o domínio do aluno no uso de conceitos de programação.

VI. Reconhecimento de padrões

O reconhecimento de padrões é uma prática não descrita no framework, mas que identificamos como prática relacionada ao emprego do conceito computacional de repetição. O reconhecimento da existência de passos repetitivos é o ponto de partida para a utilização do conceito de repetição na implementação do aluno.

A figura 33 mostra dois exemplos de desafios no qual primeiramente é preciso identificar essa padronização para posteriormente adotar estruturas de repetição no código.

Figura 33 - O reconhecimento de padrões precede a escolha pelo uso do bloco de repetição

Fonte: https://studio.code.org/s/course2 (2016).

4.5.3 As perspectivas computacionais no Code.org

As perspectivas computacionais estão relacionadas as atitudes desempenhadas pelos alunos durante o contato com a programação. Essas atitudes são relativas ao modo de se expressar, se conectar, se questionar e ao modo de agir colaborativamente. Como tais condutas são de cunho individual, e envolvem também suas perspectivas em relação aos outros. Essa dimensão não é abordada nesse estudo, visto a necessidade de realizar análises que vão além da plataforma Code.org, o que implicaria em observações e entrevistas com alunos para poder captar a dimensão do aprimoramento dessas atitudes neles.

4.6 RESUMO DOS RESULTADOS

Os resultados do estudo de caso demonstram que as competências estabelecidas no modelo de framework de Brennan e Resnick (2012) podem ser ampliadas para outra plataforma de programação introdutória como no caso o Code.org.

Os conceitos computacionais se encontram como recursos para o uso dos alunos em abordagens para resolução das atividades. Há uma certa variação quanto a autonomia do emprego de alguns conceitos, mas em linhas gerais a maior parte dos conceitos estão bem representados e empregados conforme o modelo de framework.

No percurso para alcançar os objetivos das etapas analisados do curso os alunos realizam as práticas computacionais estipuladas no framework. As abordagens realizadas se caracterizam por seguirem fielmente ao estudo, com variações impostas pela característica linear e fixa da plataforma.

A necessidade de ações de pesquisa que vão além da análise da plataforma incapacitam obter resultados acerca do desenvolvimento das perspectivas computacionais nos estudantes. Dessa forma, apurações relativas a essa esfera de competências poderão ser alcançadas em trabalhos futuros.

5 CONCLUSÃO

O presente trabalho teve como propósito compreender a relação existente entre o aprendizado da Programação Introdutória com a plataforma Code.org e o desenvolvimento do Pensamento Computacional em alunos do ensino médio. Para alcançar tal finalidade foram delimitados quatro objetivos específicos relacionados a questões complementares de igual quantidade.

O primeiro objetivo específico foi alcançado e a questão complementar associada: *Quais as habilidades e as competências esperadas em abordagens de ensino de Pensamento Computacional.* Pode ser respondida por meio da revisão bibliográfica. As subseções 2.1.3 e 2.1.4 no capítulo 2 da Fundamentação Teórica possibilitaram compreender quais são as habilidades do Pensamento Computacional e também que há uma organização para promover o desenvolvimento de competências no contexto educacional.

O segundo objetivo específico atingido e relacionado com a questão complementar: *Quais são as habilidades e as competências do Pensamento Computacional que estão relacionadas ao ensino de Programação Introdutória.* Também pode ser respondida através da revisão bibliográfica. O estudo do trabalho de Brennan e Resnick (2012) exposto na seção 2.3 do capítulo 2 permitiu identificar as habilidades e competências presentes nessa relação e que formam três grandes dimensões que se complementam, são os conceitos, práticas e perspectivas computacionais.

O terceiro e último objetivo específico alcançado remetente a questão complementar: *Quais as habilidades e as competências do Pensamento Computacional são desenvolvidas através de ambientes gráficos de ensino de Programação Introdutória como Code.org.* Foi respondida mediante ao estudo de caso realizado na plataforma e apresentado no capítulo 4. A conclusão do estudo disposta na subseção 4.6 que demonstrou quais competências podem ser encontradas na plataforma.

5.1 TRABALHOS FUTUROS

Dada a importância de se compreender a relação entre esses dois temas, torna-se necessário realizações de trabalhos futuros que visem pesquisar outros aspectos não abrangidos na presente obra. Uma vertente deixada para uma próxima

pesquisa seria a investigação do desenvolvimento de competências do pensamento computacional em atividades de computação desplugada encontradas na plataforma Code.org. Outra pesquisa poderá estender a investigação iniciada nesse trabalho e realizar análises dos demais cursos disponíveis não analisado no estudo de caso. Outro trabalho que poderá ser continuado e futuramente aprimorado, cujo os passos iniciais foram dados nessa pesquisa, e dizem respeito a investigação do desenvolvimento das competências demonstradas no 4° capítulo. Essa investigação foi inicialmente executado por meio de uma oficina de programação com o uso do Code.org descrita na subseção abaixo.

5.1.1 Execução da oficina e análise do teste

A oficina foi realizada nos dias 9 e 10 de Março de 2016 com duração diária de 4 horas. A divulgação da oficina ocorreu na Escola Estadual de Ensino Fundamental e Médio Senador Rui Carneiro situada em Mamanguape/PB. Para realizar a matrícula, os alunos não poderiam ter tido nenhuma experiência prévia com programação anteriormente. Conforme apresentado na figura 34, as aulas ocorreram no laboratório de informática da própria escola e contou com a participação de 13 alunos, sendo 8 meninas e 5 meninos na faixa etária entre 15 a 17 anos.

Figura 34 - Os alunos da oficina

Fonte: Elaborado pelo autor.

A oficina de introdução à programação realizada teve o intuído de comportar um teste para avaliar habilidades do Pensamento Computacional manifestadas através da Ensino de programação introdutória com o Code.org. O teste foi construído através de uma revisão bibliográfica de questões alusivas a um conjunto de

habilidades compreendidas como: Algoritmo, Depuração, Reconhecimento de Padrões e Raciocínio Lógico.

As questões reunidas foram incorporadas a um questionário para aplicação em dois momentos da oficina. No primeiro momento (pré-teste) o questionário foi entregue aos alunos antes da iniciação das aulas. Já no segundo momento (pós-teste), após a conclusão da oficina, os alunos foram submetidos a outro questionário contendo as mesmas questões. A comparação entre as respostas dos alunos em ambos momentos (pré-teste e pós-teste) serviriam para avaliar a manifestação das habilidades citadas anteriormente. Sendo que, o melhoramento das respostas no momento (pós-teste) denotaria capacitações adquiridas a partir da experiência de programação obtida na oficina.

Os questionários utilizados para aplicação dos testes durante a oficina se encontram na seção de apêndices. As questões se dividem em objetivas e subjetivas e, como mencionadas anteriormente, compreendem questões acerca de Algoritmo, Depuração, Reconhecimento de Padrões e Raciocínio Lógico.

As questões de Algoritmo requeriam o descrever de situações presentes nas rotinas de qualquer pessoa. Em outras palavras, o que se pretendia com a avaliação das respostas era identificar melhorias na forma como os alunos transcreveriam um algoritmo composto por um sequenciamento de passos necessários para alcançar a resolução de um dado problema.

O quesito Depuração pedia aos alunos o acompanhamento do lançar de dados entre dois jogadores em um jogo de tabuleiro. Os alunos deveriam responder quando requisitado a localização dos jogadores no tabuleiro. A questão avaliava o nível de acompanhamento dos alunos na tarefa de percorrer passos na busca de erros, algo essencialmente presente na prática de depurar códigos na programação.

As questões objetivas de Reconhecimento de padrões buscavam avaliar se os alunos conseguiriam reconhecer a existência de padrões nas figuras exibidas. Para isso os alunos teriam que indicar quais figuras seriam as próximas que iriam compor o sequenciamento do quadro de figuras apresentados.

As questões relativas ao Raciocínio Lógico visavam avaliar se a lógica dos alunos sofreria algum tipo de modificação ou progresso, após as atividades de programação. As subseções abaixo apresentam a análise e consideração do autor a respeito dos resultados obtidos após a realização dos testes na oficina de programação.

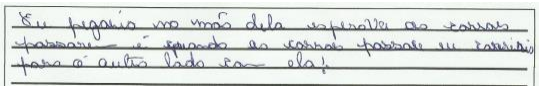
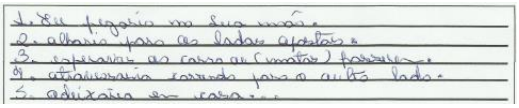
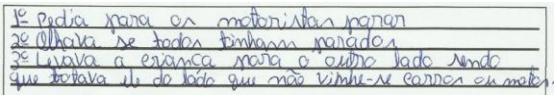
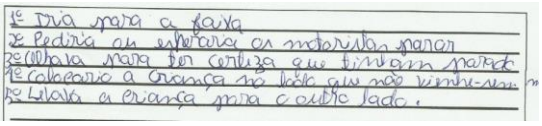
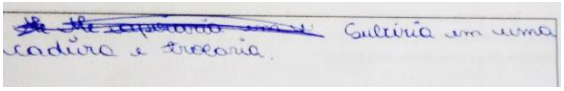
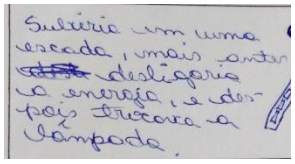
5.1.2 Algoritmo

Ao avaliarmos os resultados da forma em que se apresentaram não podemos chegar a uma conclusão definitiva para compreender se os alunos aprimoraram a capacidade relativa a construção de algoritmos.

Na comparação entre as respostas subjetivas (1 e 3) identificamos que somente três respostas apresentaram melhorias no descrever da situação. As demais respostas não apresentaram variação significativa, prevalecendo descrições praticamente idênticas em ambos os testes. A questão objetiva (2) também não apresentou variações significativas permanecendo iguais na maioria dos casos.

O quadro 1 a seguir demonstra os três casos identificados que denotariam melhorias nas respostas dos alunos no pós-teste.

Quadro 2 - Variações identificadas nas respostas de algoritmo

	Resposta do questionário pré-teste	**Resposta do questionário pós-teste**
A		
B		
C		

Fonte: Elaborado pelo autor.

Vale ressaltar que as demais respostas que não apresentam variações não foram apresentadas neste documento.

Portanto, como não foi possível chegar a algum tipo de conclusão a esse respeito lançamos algumas hipóteses que poderiam indicar os motivos que impediram o alcance de conclusões relativas ao desenvolvimento da capacidade dos alunos em criar algoritmos.

Hipótese 1 – A forma como a questão abordou não deixou claro que eles deveriam descrever com maiores detalhes possíveis, ou mesmos os alunos podem

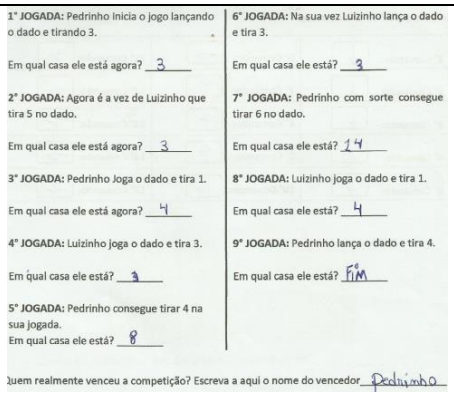
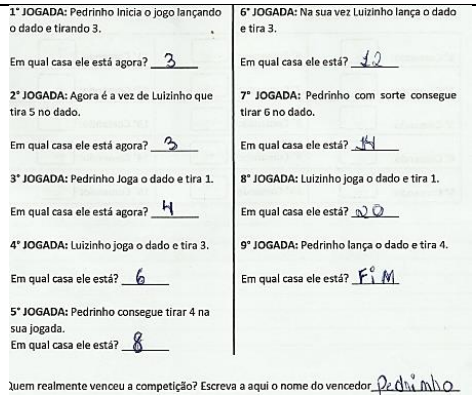
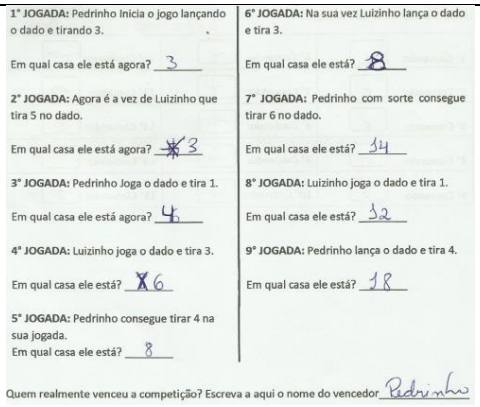
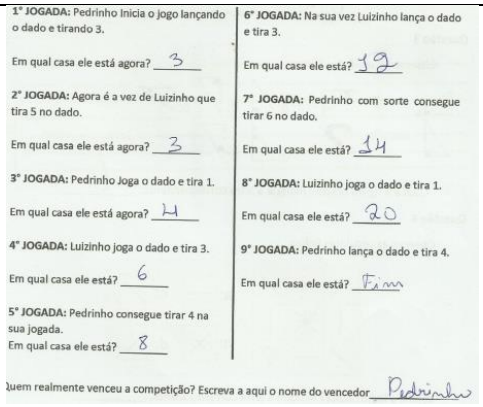
ter optado por não procurar descrever com maior riqueza de detalhes em ambos momentos.

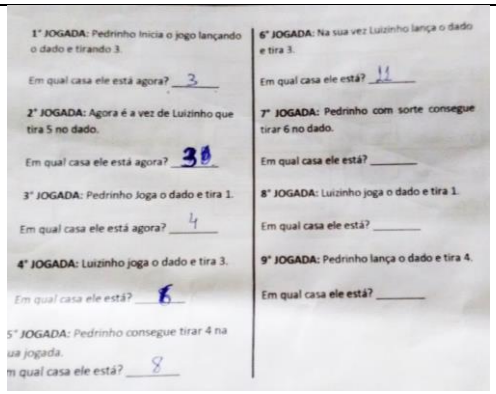
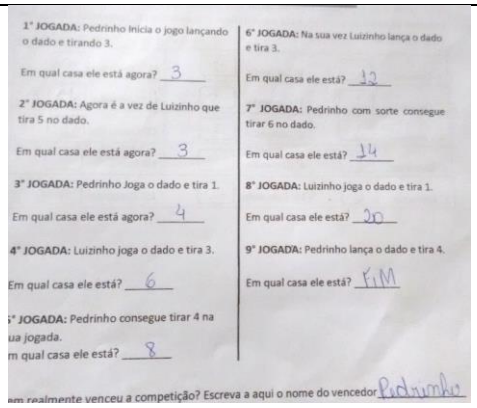
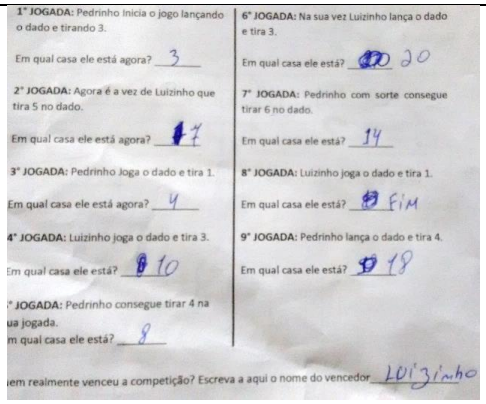
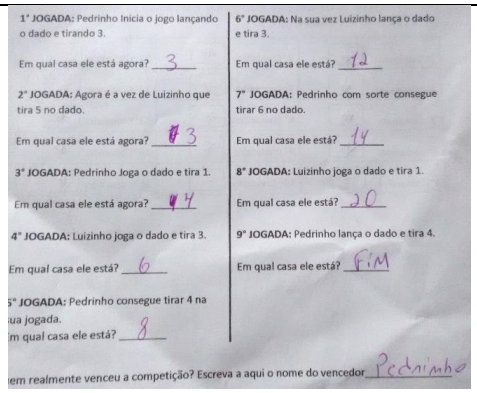
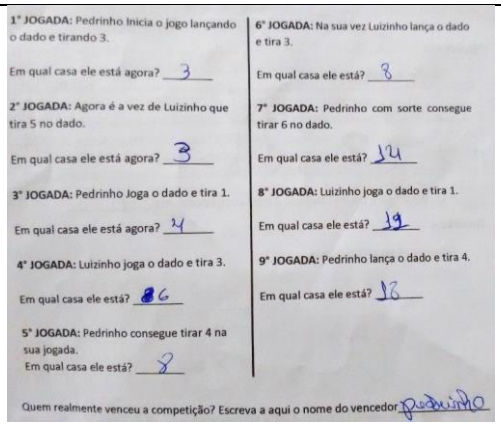
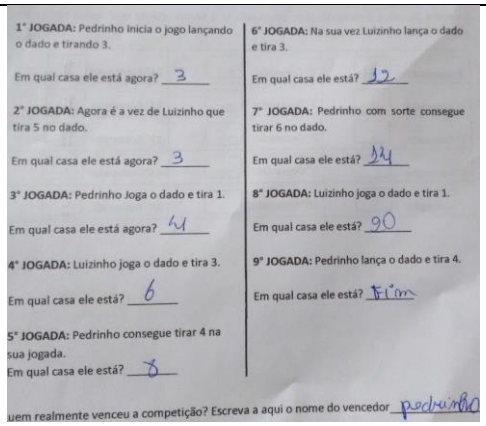
Hipótese 2 – O curto espaço de tempo entre o pré-teste e pós-teste teria contribuído para reprodução das respostas no segundo teste. Nesse caso, os alunos recordados da resposta anterior estariam predispostos a responderam novamente da mesma maneira.

5.1.3 Depuração

Das 13 respostas avaliadas, identificamos que 5 demonstraram melhoras no acompanhamento das jogadas enunciadas no problema da questão. O que indicaria que a prática de programação nas atividades do Code.org contribui para o aprimoramento da capacidade de depurar dos alunos, ou seja, realizar o acompanhamento linear de uma sequência de procedimentos para identificar e realizar correções. O quadro 2 apresenta essas respostas identificadas na análise.

Quadro 3 - Variações identificadas nas respostas de depuração

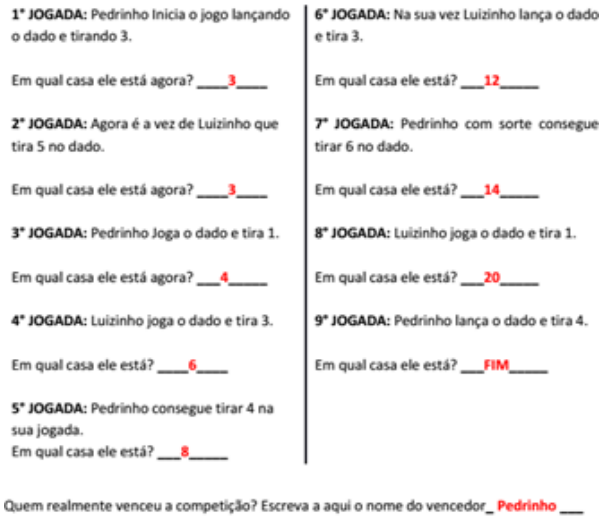
	Resposta do questionário pré-teste	**Resposta do questionário pós-teste**
A		
B		

C		
D		
E		

Fonte: Elaborado pelo autor.

A figura 35 apresenta o gabarito consultado para realizar as comparações das respostas dos alunos no pré-teste e no pós-teste.

Figura 35 - Gabarito da questão de Depuração



Fonte: Elaborado pelo autor.

Outras 5 respostas comparadas não apresentaram variações significativas, visto pela característica assertiva em todas elas. Pois, todas foram classificadas como corretas em ambos os momentos pré-teste e pós-teste. O que podemos atribuir que esse conjunto de alunos já tinham um certo aprimoramento dessa habilidade.

As 3 respostas restantes em ambos os testes não apresentaram variações, e foram classificadas como incorretas em ambos os momentos. Nesse caso podemos considerar que esse conjunto de alunos não desenvolveu em nenhum aspecto a habilidade relativa a depuração.

Dessa forma, através da avaliação dos resultados levantamos indícios que um conjunto de alunos desenvolveu a habilidade relativa a Depuração. Outro conjunto de alunos, por ter respondido a questão corretamente nos testes, denotam a presença dessa habilidade. Já os 3 alunos restantes que não apresentaram melhorias nas respostas podemos concluir que as práticas de programação não possibilitaram nesse grupo o desenvolvimento dessa habilidade.

5.1.4 Reconhecimento de Padrões e Lógica

Em nossas análises comparativas não identificamos variações significativas que indicassem de alguma forma que os alunos apresentaram avanços relativos a essas capacidades. As respostas se dividem entre grupos de alunos que acertaram questões no pré-teste e mantiveram no pós-teste e o outro grupo de alunos de forma contrária também se manteve com respostas erradas nos dois testes.

Dessa forma apenas podemos levantar hipóteses que sirvam como indícios para compreender os motivos que impossibilitaram chegar as devidas conclusões.

Hipótese 1 – As questões escolhidas para compor o questionário não foram responsáveis por fazer os alunos utilizarem realmente tais capacidades para resolução.

Hipótese 2 – O curto espaço de tempo entre o pré-teste e pós-teste teria contribuído para reprodução das respostas no segundo teste. Nesse caso, os alunos recordados da resposta anterior estariam predispostos a responderam novamente da mesma maneira.

5.1.5 Conclusão

Por meio dos instrumentos de coleta de dados selecionados (pré e pós testes) não foi possível levantar indícios de quais habilidades do pensamento computacional foram estimulados nos alunos. Entretanto, após o estudo de caso das habilidades trabalhadas no code.org, seria adequado construir um novo instrumento de coleta de dados baseado nas evidências das habilidades mapeadas pelo framework de Brennan e Resnick (2012).

No que se refere ao teste, as questões compostas precisam ser reformuladas. Devem ser melhor apropriadas para captar exatamente a habilidade investigada, para isso essas questões devem estar associadas a cada tipo de habilidades requerida ao aluno nas etapas frequentadas por ele na plataforma Code.org. O uso de questões pertinentes e bem elaboradas poderia contribuir para chegar as devidas conclusões sobre o desenvolvimento dessas habilidades investigadas.

Em resumo, o instrumento de coleta de dados adotado nesse experimento não possibilitou levantar tais indícios, o que remete a construção de um novo instrumento de coleta de dados reformulado e ancorado nas habilidades mapeadas pelo framework de Brennan e Resnick (2012) e nos resultados do presente estudo de caso realizado no Code.org.

5.2 CONSIDERAÇÕES FINAIS

Conforme mencionado anteriormente próximos trabalhos poderão compor testes que utilizem um novo instrumento de coleta validado. Além disso, ainda há lacunas acerca da identificação mais abrangente dessas habilidades e competências no contato com o Code.org, o que demanda pesquisas que utilizem novas abordagens

de investigação além da análise da plataforma realizada nesse trabalho. Dessa forma também deverão ser investigados os fatores pertinentes a práticas de programação e perspectivas computacionais dos alunos no desempenhar de suas atividades.

Portanto, o presente trabalho de pesquisa de maneira geral atingiu os objetivos inicialmente pretendidos. Conclui-se que o produto decorrente dessa pesquisa que ao longo de meses foi disposto nessas páginas poderá ser proveitoso a toda uma comunidade acadêmica por transmitir uma compreensão acerca da relação entre dois temas notoriamente importantes para a formação das crianças e jovens do hoje e do amanhã.

REFERÊNCIAS

BARANAUSKAS, Maria Cecília Calani *et al*. Uma taxonomia para ambientes de aprendizado baseados no computador. **Valente, JA O computador na sociedade do conhecimento.** Campinas, SP: UNICAMP/NIED, 1999.

BARCELOS, Thiago *et al*. **Relações entre o pensamento computacional e a Matemática: uma Revisão Sistemática da Literatura.** In: Anais dos Workshops do Congresso Brasileiro de Informática na Educação. 2015. p. 1369.

BARR, Valerie; STEPHENSON, Chris. **Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community?**. Acm In roads, v. 2, n. 1, p. 48-54, 2011.

BLIKSTEIN, Paulo. O pensamento computacional e a reinvenção do computador na educação. 2008.Disponível em <http://www.blikstein.com/paulo/documents/online/ol_pensamento_computacional.html> Acessado em: 10 jan. 2016.

BONOTTO, Gabriele; FELICETTI, Vera Lucia. **Habilidades e competências na prática docente: perspectivas a partir de situações-problema.** Educação Por Escrito, v. 5, n. 1, p. 17-29, 2014.

BRENNAN, Karen; RESNICK, Mitchel. New frameworks for studying and assessing the development of computational thinking. In: **Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada**. 2012.

CODE.ORG. Fundação sem fins lucrativos dedicada ao crescimento do ensino de programação de computadores. Disponível em: <https://code.org>. Acesso em: 30 abr. 2016.

SILVEIRA JÚNIOR[1], Garibaldi *et al*. **Análise da ferramenta de programação visual blockly como recurso educacional no ensino de programação**.

DANTAS, Ricardo Fidelis; DA COSTA, Francisco Eudes Almeida. **CODE: O ensino de linguagens de programação educativas como ferramentas de ensino/aprendizagem**. In: Simpósio Hipertexto e Tecnologias na Educação, 5, 2013, Recife. Disponível em: / < https://issuu.com/simposiohipertexto/docs/programacao-simposio-hipertexto2013>. Acesso em: 21 Mai 2016.

CARVALHO, Márcio Luiz Bunte; CHAIMOWICZ, Luiz; MORO, Mirella M. **Pensamento computacional no Ensino Médio Mineiro.** In: Workshop de Educação em Computação, Congresso da Sociedade Brasileira de Computação. 2013.

FRANÇA, Rozelma Soares; AMARAL, Haroldo José Costa. Proposta Metodológica de Ensino e Avaliação para o Desenvolvimento do pensamento computacional com o Uso do Scratch. In: **Anais do Workshop de Informática na Escola**. 2013. p. 179.

OLIVEIRA, Maxwell Ferreira. **Metodologia científica: um manual para a realização de pesquisas em Administração**. 2011.

FERREIRA, Aurélio Buarque de Holanda. **Mini Aurélio Século XXI Escolar: o minidicionário da língua portuguesa**. 5ª ed. rev. e ampliada. Rio de Janeiro: Editora Nova Fronteira, 2001.

FERREIRA, Aurélio Buarque de Holanda. **Dicionário Aurélio**. Disponível em: <https://dicionariodoaurelio.com/habilidade>. Acesso em 23, janeiro de 2016.

FINCHER, Sally *et al*. Comparing alice, greenfoot & scratch. In: **Proceedings of the 41st ACM technical symposium on Computer science education**. ACM, 2010. p. 192-193.

FONSECA, J. J. S. **Metodologia da pesquisa Científica.** Fortaleza: UEC, 2002. Apostila. Disponível em <http://www.ia.ufrrj.br/ppgea/conteudo/conteudo-2012> Acessado em: 20 jan. 2016.

GADOTTI, Moacir. A escola e o professor: Paulo Freire e a paixão de ensinar. **Produção de terceiros sobre Paulo Freire; Série Prefácios**, 2007.

GERHARDT, Tatiana Engel; SILVEIRA, Denise Tolfo. **Métodos de pesquisa**. PLAGEDER, 2009.

GIL, Antonio Carlos. Como elaborar projetos de pesquisa. **São Paulo**, v. 5, p. 61, 2002.

ISTE – International Society for Technology in Education; CSTA - Computer Science Teachers Association; NSF - National Science Foundation. **Computational thinking: leadership toolkit**. First Edition, 2011.

KELLEHER, C.; PAUSCH, R**. Lowering the Barriers to Programming: A Taxonomy of Programming Environments and languages for Novic Programmers**. In ACM Computing Surveys, v.37, n. 2,83-137, 2005.

LAKATOS, Eva Maria; MARCONI, Marina de Andrade. Fundamentos da metodologia científica. In: **Fundamentos da metodologia científica**. . 5. Ed. São Paulo. Altas, 2003.

MALONEY, J., RESNICK, M., RUSK, N., SILVERMAN, B., and EASTMOND, E. 2010. **The scratch programming language and environment.** ACM Trans. Comput. Educ. 10, 4, Article 16 (November 2010), 15 pages. DOI = 10.1145/1868358.1868363. http://doi.acm.org/10.1145/1868358.1868363

MANNILA, Linda *et al*. **Computational thinking in k-9 education**. In:Proceedings of the Working Group Reports of the 2014 on Innovation & Technology in Computer Science Education Conference. ACM, 2014. p. 1-29.

NUNES, Daltro José. Ciência da computaçao na educaçao básica. **Jornal da Ciência**, v. 9, 2011.Disponível em <http://www.adufrgs.org.br/artigos/ciencia-da-computacao-na-educacao-basica/> Acessado em: 08 fev. 2016.

NRC10. **Report of a Workshop on The Scope and Nature of Computational Thinking Committee for the Workshops on Computational Thinking; National Research Council**, 2009. Disponível em: http://www.nap.edu/catalog/12840.html. Acesso em: 03 jan.2016.

PHILLIPS, Pat. Computational Thinking: a problem-solving tool for every classroom. **Communications of the CSTA**, v. 3, n. 6, p. 12-16, 2009.

PRODANOV, Cleber Cristiano; DE FREITAS, Ernani Cesar. **Metodologia do Trabalho Científico: Métodos e Técnicas da Pesquisa e do Trabalho Acadêmico-2ª Edição**. Editora Feevale, 2013.

SCAICO, Pasqueline Dantas *et al*. Ensino de Programação no Ensino Médio: Uma Abordagem Orientada ao Design com a linguagem Scratch. **Revista Brasileira de Informática na Educação**, v. 21, n. 02, p. 92, 2013.

SELLTIZ, Claire. **Métodos de pesquisa nas relações sociais**. EPU, 1974.

SOUSA *et al.* 2015. **O ENSINO DE PROGRAMAÇÃO PARA CRIANÇAS DA REDE PÚBLICA DE CAMPINA GRANDE.** Disponível em < http://www.editorarealize.com.br/revistas/conedu/trabalhos/TRABALHO_EV045_MD4_SA13_ID222_09092015104347.pdf > Acessado em: 08 abr. 2016.

TAROUCO, Liane Margarida Rockenbach *et al*. Jogos educacionais.**CINTED, UFRGS**, 2004.

VALENTE, José Armando. Mudanças na sociedade, mudanças na educação: o fazer e o compreender. **O computador na sociedade do conhecimento**, v. 1, p. 29-48, 1999.

VALENTE, josé Armando. **O uso inteligente do computador na educação.** Revista Pátio, ano I, n. 1, p. 19-21, mai/jul, 1997.

WING, Jeannette M. Computational thinking. **Communications of the ACM**, v. 49, n. 3, p. 33-35, 2006.

WING, Jeannette M. Computationalthinkingandthinkingaboutcomputing.**PhilosophicalTransactionsofthe Royal Societyof London A: Mathematical, PhysicalandEngineeringSciences**, v. 366, n. 1881, p. 3717-3725, 2008.

APÊNDICES

APÊNDICE A - FORMULÁRIO PRÉ-TESTE

UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE CIÊNCIAS APLICADAS E EDUCAÇÃO
LICENCIATURA EM CIENCIA DA COMPUTAÇÃO
CAMPUS IV - LITORAL NORTE

Questionário para identificar habilidades e competências do pensamento computacional

Este questionário tem como objetivo identificar habilidades e competências referentes ao pensamento computacional nos alunos participantes das oficinas de Introdução à programação ministradas pelos bolsistas do PIBID Licenciatura em Ciência da Computação.

Aluno: _____

SEQUÊNCIA

Questão 1

Como você faria para trocar uma lâmpada em um ambiente alto? Descreva todos os passos.

Questão 2

As etapas abaixo são referentes ao preparo de um bolo. Reorganize a etapas descritas enumerando na ordem correta de execução.

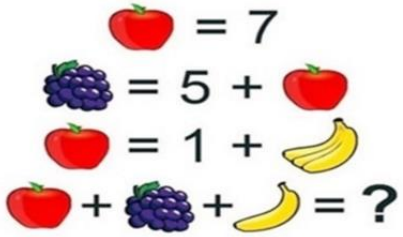
( ) Levar o bolo ao forno
( ) Bater a massa
( ) Colocar cobertura
( ) Inserir os ingredientes
( ) Ligar o forno
( ) Reunir os ingredientes
( ) Desligar o forno
( ) Servir o bolo

Questão 3

Como você guiaria uma criança na travessia de uma rua? Seu objetivo é ajudá-la a passar para o outro lado da rua com segurança. Abaixo descreva os passos necessários numerando-os.

RACIOCÍNIO LÓGICO

Questão 1



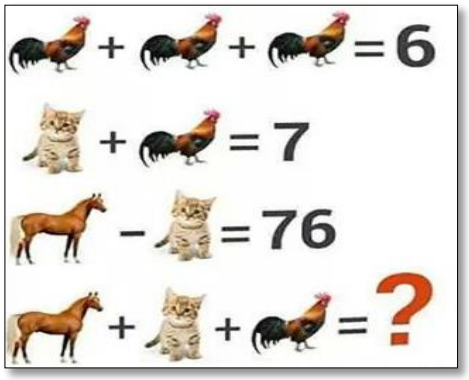
Qual o resultado oculto? _____

Questão 2

A sequência **1, 4, 10, 22, 46, 94, ...,** obedece a uma regra lógica. O termo dessa série subsequente ao número 94 é:

A. 112 ( )
B. 130 ( )
C. 190 ( )
D. 215 ( )
E. 165 ( )

Questão 3

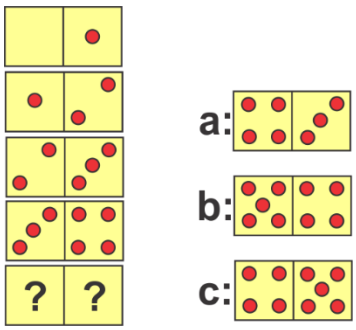


Qual o resultado oculto? _____

RECONHECIMENTO DE PADRÕES

Questão 1

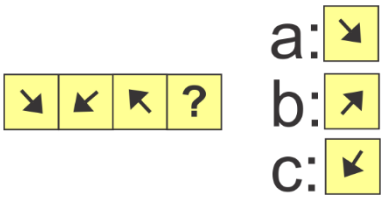
Observe as seguintes peças de Dominó. Note o lado esquerdo de cada peça, e em seguida, o seu lado direito.



Qual é a peça oculta? Escreva a letra correspondente. _____

Questão 2

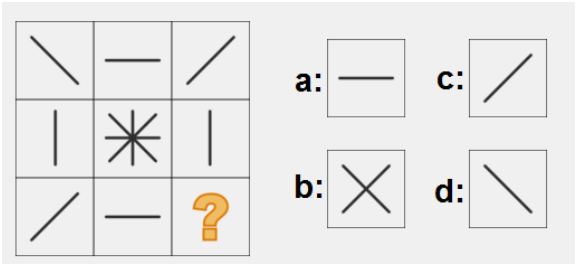
Observe as seguintes setas.



Qual é a seta oculta? Indique a letra correspondente. _____

Questão 3

Observe as seguintes linhas.

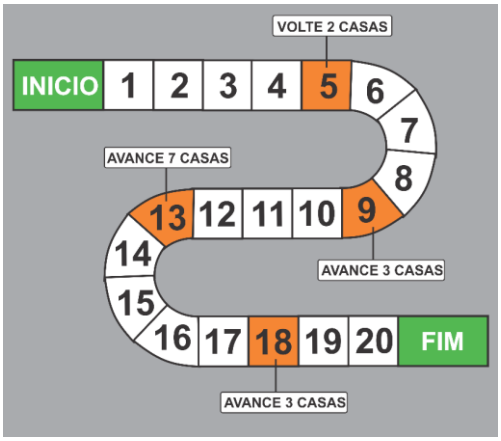


Qual é a linha oculta? Indique a letra correspondente. _____

DEPURAÇÃO

Questão 1

Pedrinho e Luizinho disputaram em um jogo de tabuleiro para ver quem chegaria primeiro ao fim. A regra do Jogo é clara: cada jogador, por vez, lança um único dado para obter o valor equivalente ao número de casas que irá percorrer no tabuleiro. Pedrinho venceu a disputa, mas Luizinho não concorda com o resultado dizendo que ele errou na contagem das casas. Ajude-os a descobrir quem é o verdadeiro ganhador.



1° JOGADA: Pedrinho Inicia o jogo lançando o dado e tirando 3.

Em qual casa ele está agora? _____

2° JOGADA: Agora é a vez de Luizinho que tira 5 no dado.

Em qual casa ele está agora? _____

3° JOGADA: Pedrinho Joga o dado e tira 1.

Em qual casa ele está agora? _____

4° JOGADA: Luizinho joga o dado e tira 3.

Em qual casa ele está? _____

5° JOGADA: Pedrinho consegue tirar 4 na sua jogada.
Em qual casa ele está? _____

6° JOGADA: Na sua vez Luizinho lança o dado e tira 3.

Em qual casa ele está? _____

7° JOGADA: Pedrinho com sorte consegue tirar 6 no dado.

Em qual casa ele está? _____

8° JOGADA: Luizinho joga o dado e tira 1.

Em qual casa ele está? _____

9° JOGADA: Pedrinho lança o dado e tira 4.

Em qual casa ele está? _____

Quem realmente venceu a competição? Escreva a aqui o nome do vencedor_____

APÊNDICE B - FORMULÁRIO PÓS-TESTE

UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE CIÊNCIAS APLICADAS E EDUCAÇÃO
LICENCIATURA EM CIENCIA DA COMPUTAÇÃO
CAMPUS IV - LITORAL NORTE

Questionário para identificar habilidades e competências do pensamento computacional

Este questionário tem como objetivo identificar habilidades e competências referentes ao pensamento computacional nos alunos participantes das oficinas de Introdução à programação ministradas pelos bolsistas do PIBID Licenciatura em Ciência da Computação.

Aluno: _____

SEQUÊNCIA

Questão 1

Como você faria para trocar uma lâmpada em um ambiente alto? Descreva todos os passos.

Questão 2

As etapas abaixo são referentes ao preparo de um bolo. Reorganize a etapas descritas enumerando na ordem correta de execução.

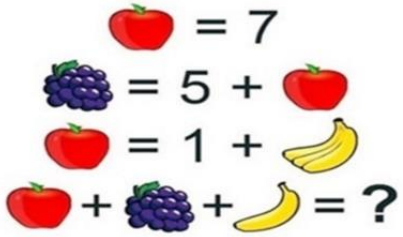
( ) Levar o bolo ao forno
( ) Bater a massa
( ) Colocar cobertura
( ) Inserir os ingredientes
( ) Ligar o forno
( ) Reunir os ingredientes
( ) Desligar o forno
( ) Servir o bolo

Questão 3

Como você guiaria uma criança na travessia de uma rua? Seu objetivo é ajudá-la a passar para o outro lado da rua com segurança. Abaixo descreva os passos necessários numerando-os.

RACIOCÍNIO LÓGICO

Questão 1



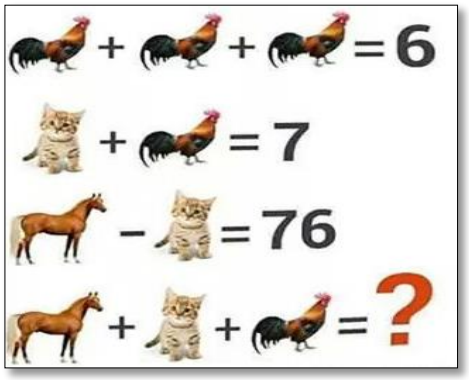
Qual o resultado oculto? _____

Questão 2

A sequência **1, 4, 10, 22, 46, 94, ...,** obedece a uma regra lógica. O termo dessa série subsequente ao número 94 é:

F. 112 ( )
G. 130 ( )
H. 190 ( )
I. 215 ( )
J. 165 ( )

Questão 3

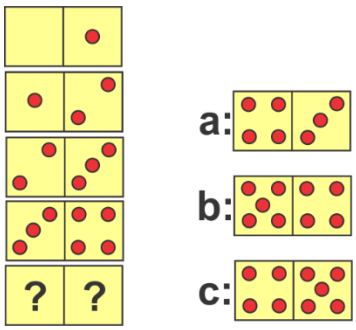


Qual o resultado oculto? _____

RECONHECIMENTO DE PADRÕES

Questão 1

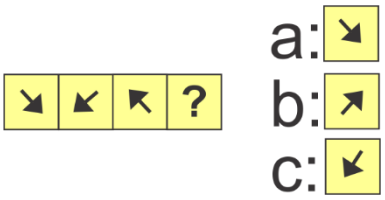
Observe as seguintes peças de Dominó. Note o lado esquerdo de cada peça, e em seguida, o seu lado direito.



Qual é a peça oculta? Escreva a letra correspondente. _____

Questão 2

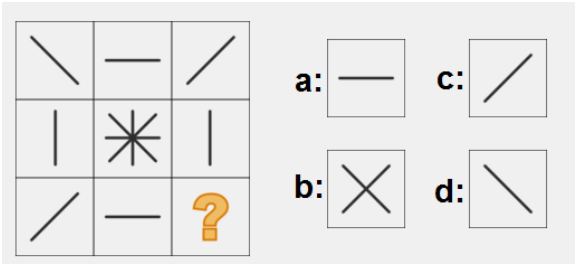
Observe as seguintes setas.



Qual é a seta oculta? Indique a letra correspondente. _____

Questão 3

Observe as seguintes linhas.

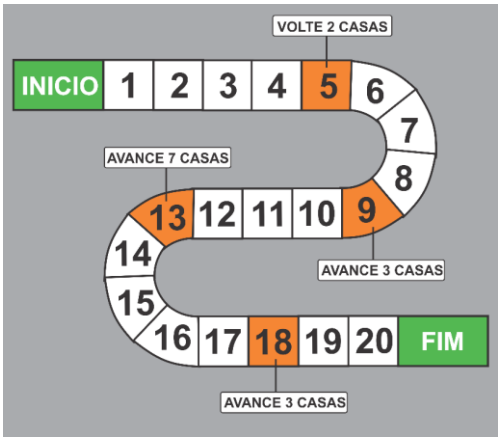


Qual é a linha oculta? Indique a letra correspondente. _____

DEPURAÇÃO

Questão 1

Pedrinho e Luizinho disputaram em um jogo de tabuleiro para ver quem chegaria primeiro ao fim. A regra do Jogo é clara: cada jogador, por vez, lança um único dado para obter o valor equivalente ao número de casas que irá percorrer no tabuleiro. Pedrinho venceu a disputa, mas Luizinho não concorda com o resultado dizendo que ele errou na contagem das casas. Ajude-os a descobrir quem é o verdadeiro ganhador.



1° JOGADA: Pedrinho Inicia o jogo lançando o dado e tirando 3.

Em qual casa ele está agora? _____

2° JOGADA: Agora é a vez de Luizinho que tira 5 no dado.

Em qual casa ele está agora? _____

3° JOGADA: Pedrinho Joga o dado e tira 1.

Em qual casa ele está agora? _____

4° JOGADA: Luizinho joga o dado e tira 3.

Em qual casa ele está? _____

5° JOGADA: Pedrinho consegue tirar 4 na sua jogada.
Em qual casa ele está? _____

6° JOGADA: Na sua vez Luizinho lança o dado e tira 3.

Em qual casa ele está? _____

7° JOGADA: Pedrinho com sorte consegue tirar 6 no dado.

Em qual casa ele está? _____

8° JOGADA: Luizinho joga o dado e tira 1.

Em qual casa ele está? _____

9° JOGADA: Pedrinho lança o dado e tira 4.

Em qual casa ele está? _____

Quem realmente venceu a competição? Escreva a aqui o nome do vencedor_____