REVISTA TÓPICOS
ISSN 2965-6672 | Qualis A2

CÓDIGOS DE DETECÇÃO DE ERROS NA EDUCAÇÃO BÁSICA POR MEIO DE UMA PROPOSTA DIDÁTICA COM DÍGITOS VERIFICADORES DO CPF E PROGRAMAÇÃO SCRATCH

ERROR-DETECTION CODES IN BASIC EDUCATION THROUGH A DIDACTIC PROPOSAL USING CPF CHECK DIGITS AND SCRATCH PROGRAMMING

Ciências Exatas e da Terra • 20/11/2026

REGISTRO DOI: 10.70773/revistatopicos/794952515

Rayanne Auxiliadora de Oliveira Matos[1]

Murillo Alves Macêdo[2]

Myrian Cristina Sousa Pacheco[3]

Oscar Oliveira Vieira[4]

RESUMO

Códigos de detecção de erros são utilizados em diferentes sistemas de identificação e transmissão de informações, mas seus princípios matemáticos nem sempre são explorados na Educação Básica. Este artigo apresenta uma proposta didática que utiliza os dígitos verificadores do Cadastro de Pessoas Físicas (CPF) para articular conteúdos matemáticos, resolução de problemas, pensamento computacional e programação em blocos. A sequência parte de uma situação cotidiana, conduz os estudantes à investigação de somas ponderadas e restos de divisões por 11 e avança para a formalização de um algoritmo, sua representação em fluxograma e sua implementação no Scratch. Além de descrever o cálculo, discute-se que a validação matemática não comprova a existência nem a situação cadastral de um CPF e não detecta necessariamente toda alteração possível. A fundamentação mobiliza resolução de problemas, investigação, argumentação e tecnologias digitais como recursos para explorar ideias matemáticas. Trata-se de uma proposta didática teoricamente fundamentada para aplicação na Educação Básica.

Palavras-chave: códigos de detecção de erros; dígitos verificadores; CPF; Scratch; Educação Matemática.

ABSTRACT

Error-detection codes are used in identification and information-transmission systems, yet their mathematical principles are not always explored in Basic Education. This paper presents a didactic proposal using the check digits of the Brazilian Individual Taxpayer Registry (CPF) to connect mathematics, problem solving, computational thinking, and block-based programming. Students investigate weighted sums and remainders modulo 11, formalize an algorithm, represent it in a flowchart, and implement it in Scratch.

The paper also discusses the limits of the method: mathematical validation does not establish whether a CPF has been assigned or whether its registration status is regular, and it does not detect every possible alteration. The proposal draws on problem solving, inquiry, mathematical argumentation, and digital technologies as resources for exploring mathematical ideas. It is presented as a didactic proposal rather than as a report of an empirically evaluated intervention.
Keywords: error-detection codes; check digits; CPF; Scratch; Mathematics Education.

1. INTRODUÇÃO

Sequências numéricas são utilizadas em documentos, cadastros, operações comerciais e sistemas de informação. Em muitos casos, alguns algarismos são calculados a partir dos demais para permitir a verificação de determinadas propriedades da sequência. Esses algarismos, chamados dígitos verificadores, oferecem uma oportunidade para discutir como regras matemáticas podem contribuir para a confiabilidade das informações e ser traduzidas em procedimentos executáveis.

O Cadastro de Pessoas Físicas (CPF) é um contexto familiar para muitos estudantes brasileiros. Seu número possui onze algarismos, nos quais os dois últimos são calculados por meio de somas ponderadas e restos de divisões por 11. A pergunta “como um sistema pode verificar se uma sequência foi digitada de acordo com determinada regra?” transforma um objeto cotidiano em problema matemático.

Este artigo apresenta uma proposta didática para investigar os dígitos verificadores do CPF e construir, no *Scratch*, um programa que confira a consistência matemática de uma sequência numérica. A proposta não é ensinar programação como atividade isolada nem reduzir a Matemática à execução de cálculos. Busca-se organizar uma experiência em que os estudantes formulem hipóteses, calculem, identifiquem regularidades, justifiquem procedimentos e testem conclusões.

O texto apresenta os fundamentos matemáticos e didáticos da proposta, explica o cálculo dos dígitos verificadores do CPF, descreve uma sequência de atividades e discute suas possibilidades e limitações pedagógicas.

2. FUNDAMENTAÇÃO MATEMÁTICA E DIDÁTICA

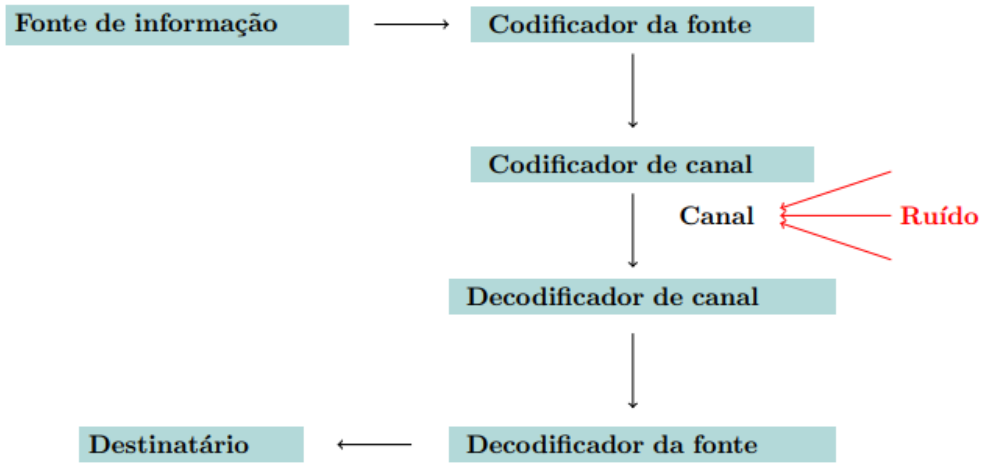
2.1. Conceitos Básicos Sobre Teoria dos Códigos

Segundo Hefez e Villela (2008), os códigos detectores e corretores de erros fazem parte do nosso cotidiano, estando presentes, por exemplo, sempre que fazemos uso de informações digitalizadas. Durante a transmissão ou armazenamento de dados, podem ocorrer interferências que podem modificar a mensagem original, o que pode fazer com que a mensagem enviada seja diferente. Um código detector e corretor de erros é, em essência, um modo organizado de acrescentar algum dado adicional a cada informação que se queira transmitir ou armazenar, que permita, ao recuperar a informação, detectar e corrigir erros (quando possível). A Teoria dos Códigos estuda justamente técnicas que tentam detectar e corrigir tais erros.

A mensagem que será enviada é a **fonte de informação.** Ela deve passar pelo **codificador da fonte,** que codifica a mensagem original

de acordo com o meio no qual será transmitida. Em seguida, são adicionadas as informações redundantes, aqui temos o **código de canal.** A mensagem será transmitida por algum canal (radiofrequência, canal de micro-ondas, cabo, circuito integrado digital, etc.). Nesse momento podem ocorrer ruídos que podem modificar a mensagem enviada. A mensagem modificada pelo ruído é recebida. O **decodificador de canal** detecta e corrige o erro (quando possível). A mensagem passa pelo **decodificador da fonte,** realizando o processo inverso do codificador da fonte e então chega ao destinatário. Esse procedimento pode ser esquematizado conforme mostra a Figura 1.

Figura 1. Esquema de codificação/decodificação



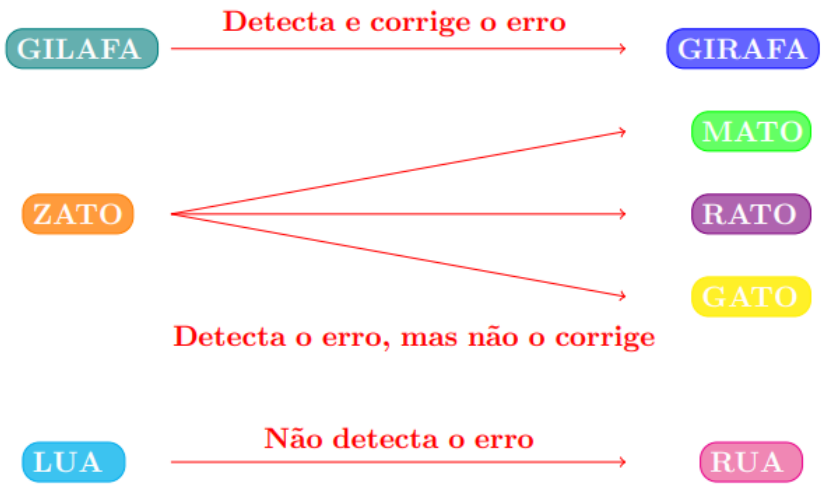
Fonte: Adaptado de Lidl e Niederreitr (1994), p. 306.

Vale destacar que detectar e corrigir erros são tarefas independentes. Detectar significa identificar que a sequência recebida não satisfaz a regra adotada; corrigir significa determinar qual era a informação original.

Um exemplo de código detector e corretor de erros é o idioma. Quando utilizamos aplicativos de envio de mensagens de texto, ao digitar a palavra **GILAFA** imediatamente ele consegue detectar o erro, pois essa palavra não pertence ao idioma. E ainda mais, ele

conclui que a palavra que se pretendia escrever era **GIRAFA,** pois é a palavra do idioma mais próxima da palavra recebida. Neste caso, foi possível detectar e corrigir o erro. No entanto, se a palavra digitada for **ZATO,** o erro é detectado, mas não é possível realizar a correção, pois existem várias palavras do idioma igualmente próximas a ela, conforme podemos ver na Figura 2. Poderá ainda ocorrer a situação onde não é possível detectar o erro. Se a intenção era digitar a palavra **RUA** e ao invés disso for digitada a palavra **LUA,** o erro passará despercebido, pois a palavra **LUA** é uma palavra do idioma (Caldeira; Matos, 2021).

Figura 2. Detecção e correção de erros



Fonte: Caldeira e Matos (2023).

Outro exemplo onde aparecem códigos detectores e corretores de erros são os dígitos verificadores. Eles são utilizados em sequências numéricas que exigem integridade e validação rápida, como documentos de identificação e transações financeiras, como por exemplo, em documentos oficiais (CPF, CNPJ e RG no Brasil), dados bancários (número de agências e contas correntes/poupanças), pagamentos (códigos de barras de boletos bancários e faturas de concessionárias), códigos de barras comerciais (como EAN e UPC),

entre outras situações. Neste artigo, será abordada, em particular, a estrutura matemática dos dígitos verificadores do CPF e nas possibilidades didáticas decorrentes de sua análise.

No caso dos dígitos verificadores do Cadastro de Pessoas Físicas (CPF), a verificação consiste em analisar se os algarismos informados obedecem às regras matemáticas estabelecidas para sua determinação. Quando a sequência não satisfaz essas regras, é possível identificar uma inconsistência. Entretanto, quando a sequência é considerada consistente, isso não comprova que o CPF tenha sido efetivamente atribuído a uma pessoa, que esteja em situação cadastral regular ou que não tenha ocorrido qualquer alteração nos algarismos. Assim, o mecanismo deve ser compreendido como um procedimento de verificação matemática, cujo alcance depende das propriedades da regra utilizada.

2.2. O CPF Como Contexto de Investigação Matemática

O Cadastro de Pessoas Física (CPF) desempenha hoje um papel importante na vida do brasileiro. O CPF é um banco de dados gerenciado pela Secretaria da Receita Federal do Brasil no qual podem se inscrever voluntária ou obrigatoriamente brasileiros e estrangeiros, independentemente de idade. Cada inscrito é unicamente identificado por um número de inscrição de onze dígitos decimais. (Clubes de Matemática da Obmep, 2020)

Inicialmente utilizado para identificar o cadastrado junto à Receita Federal para fins tributários, atualmente, o CPF ultrapassou o âmbito tributário e passou a fazer parte de diversas situações no cotidiano da população brasileira e essa presença torna o CPF um contexto particularmente interessante para o ensino de Matemática, uma vez

que um objeto familiar aos estudantes pode ser utilizado para investigar a estrutura numérica do documento e, especialmente, a Matemática envolvida na construção de seus dígitos verificadores.

É justamente essa característica - por trás de uma sequência aparentemente comum de onze algarismos existe um procedimento matemático capaz de verificar determinadas propriedades dessa sequência - que nos interessa neste trabalho. A partir dessa observação, podemos transformar um elemento cotidiano em objeto de investigação matemática e, posteriormente, em um algoritmo implementado no Scratch. A familiaridade com um contexto pode favorecer a formulação de perguntas, mas não produz aprendizagem por si só. É necessário transformá-lo em uma tarefa que convide os estudantes a observar, conjecturar, testar e justificar. A pergunta inicial sugerida é: “Os dois últimos algarismos de um CPF são escolhidos livremente ou dependem dos anteriores? Como poderíamos investigar isso?”.

2.3. Resolução de Problemas, Investigação e Argumentação

A resolução de problemas pode organizar o ensino de Matemática ao favorecer a mobilização de conhecimentos prévios, a elaboração de estratégias e a discussão das justificativas que sustentam uma solução. Onuchic e Allevato (2011) abordam a resolução de problemas como uma perspectiva para o ensino-aprendizagem-avaliação de Matemática, na qual o problema é o ponto de partida, para que os estudantes possam participar ativamente da construção de significados, fazendo conexões entre diferentes ramos da matemática, desenvolvendo novos conceitos e novos conteúdos, enquanto o professor orienta e sistematiza os conhecimentos desenvolvidos.

Nesta proposta, a investigação parte da seguinte questão: os dois últimos algarismos de um CPF são escolhidos livremente ou dependem dos algarismos anteriores? A partir desse questionamento, os estudantes são convidados a observar exemplos fictícios, realizar cálculos, comparar resultados e formular hipóteses sobre as regras utilizadas na determinação dos dígitos verificadores. Após essa exploração inicial, a sistematização matemática permite relacionar as regularidades observadas às somas ponderadas e aos restos das divisões por 11.

O processo investigativo não se encerra na identificação de uma regra. Os estudantes também precisam explicar os procedimentos realizados, comparar estratégias e justificar por que cada etapa do cálculo é necessária. A passagem dos exemplos particulares para uma regra geral constitui uma oportunidade para desenvolver a argumentação matemática e distinguir uma regularidade observada de uma justificativa que explique seu funcionamento.

A investigação prossegue com a elaboração e a testagem de um algoritmo. Os estudantes podem analisar sequências fictícias que atendam ou não às regras dos dígitos verificadores, investigar possíveis falhas e comparar os resultados obtidos manualmente com as respostas produzidas pelo programa. Nesse processo, é importante compreender que o funcionamento correto em alguns testes não constitui, por si só, uma demonstração de que o algoritmo funciona para todas as entradas possíveis. A análise dos resultados e o reconhecimento dos limites do procedimento integram, portanto, a aprendizagem proposta.

2.4. Tecnologias Digitais e Pensamento Computacional

O uso de tecnologias digitais no ensino de Matemática deve estar associado a objetivos de aprendizagem definidos, de modo que os recursos tecnológicos contribuam para a exploração, a representação e a compreensão dos conceitos estudados. Borba e Penteado (2010) discutem as relações entre informática e Educação Matemática, destacando as possibilidades que as tecnologias oferecem para reorganizar a atividade matemática.

De acordo com Zorzo, Raabe e Brackmann (2021), os conhecimentos em Computação são tão importantes para a vida na sociedade contemporânea, quanto os conhecimentos de Matemática, Física, Química, dentre outras.

Dentro dessa temática, Papert afirma que:

> *[...]a tecnologia pode apoiar uma mega mudança na educação tão ampla quanto a que vimos na medicina, porém em um processo diretamente oposto ao que conduziu às mudanças na medicina moderna. A medicina mudou, tornando-se cada vez mais técnica em sua natureza; na educação, a mudança virá pela utilização de meios técnicos para eliminar a natureza técnica da aprendizagem na Escola. (Papert, 1994, p. 34).*

As ideias de Papert (1994) se conectam diretamente com o conceito de Pensamento Computacional, que foi definido por Wing (2006, p. 33) como “uma capacidade fundamental para qualquer um, e não apenas para os cientistas informáticos” e envolve “resolver problemas, conceber sistemas e compreender o comportamento humano, tirando partido dos conceitos que são fundamentais para a Ciência Informática”.

Nesse sentido, o computador pode ser visto como uma ferramenta que possibilita a construção do conhecimento, em vez de apenas transmiti-lo. O Pensamento Computacional, por exemplo, pode ser desenvolvido a partir da interação com a máquina, permitindo que os alunos explorem, experimentem e criem soluções para problemas.

Segundo pesquisas recentes (Code.Org, 2015; Liukas, 2015; Grover; Pea, 2013; Csizmadia *et. al,* 2015), o Pensamento Computacional pode ser dividido em quatro pilares principais:

- Decomposição: Dividir problemas complexos em partes menores e mais fáceis de gerenciar.

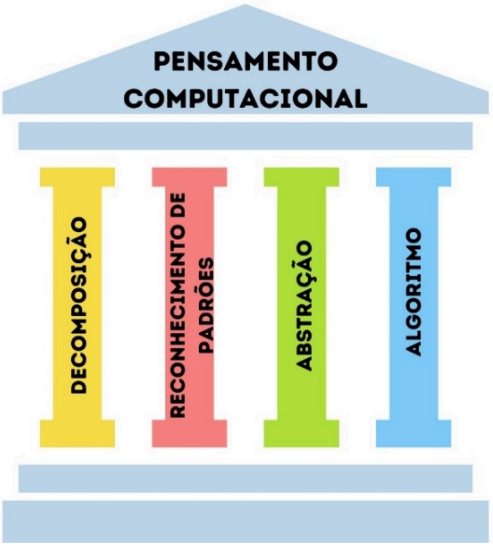
- Reconhecimento de Padrões: Identificar padrões ou tendências repetidas em dados ou processos.

- Abstração: Focar nos aspectos essenciais de um problema, ignorando detalhes irrelevantes e criando modelos simplificados.

- Algoritmo: Desenvolver instruções passo a passo para resolver um problema, geralmente através da programação, mas também em outras áreas.

Na Figura 3, apresentamos um esquema dos quatro pilares que sustentam o Pensamento Computacional.

Figura 3. Os quatro pilares do Pensamento Computacional



Fonte: Elaborada pelos autores (2026).

No contexto desta atividade, esse pensamento é desenvolvido por meio da decomposição do problema em etapas, da identificação de regularidades nos cálculos, da seleção das informações necessárias, da organização de instruções e da verificação dos resultados produzidos pelo programa.

A decomposição aparece quando o cálculo dos dois dígitos verificadores é organizado em etapas distintas. O reconhecimento de padrões é mobilizado na observação dos pesos utilizados nas somas ponderadas. A abstração envolve identificar quais dados são necessários para realizar os cálculos e quais informações podem ser desconsideradas. A elaboração do algoritmo consiste em organizar as operações em uma sequência lógica, enquanto a depuração ocorre quando os estudantes identificam e corrigem erros nas instruções implementadas. Por sua vez, a testagem permite examinar o comportamento do programa diante de diferentes entradas e avaliar se os resultados são coerentes com a regra matemática.

A Base Nacional Comum Curricular (BNCC) e o complemento de Computação à BNCC fornecem referências para o trabalho com

resolução de problemas, algoritmos e tecnologias digitais na Educação Básica (BRASIL, 2018; BRASIL, 2022). A proposta dialoga com essas orientações ao articular raciocínio matemático, representação algorítmica e programação. Entretanto, a simples utilização de um ambiente de programação não garante a aprendizagem: é necessário que os estudantes compreendam as regras que estão implementando, expliquem suas decisões e revisem suas soluções com base nos resultados obtidos.

A relação entre pensamento matemático e computacional não é automática: programar exige decisões, mas a aprendizagem depende da qualidade da tarefa, da mediação docente e das oportunidades de explicar e revisar a solução. A programação torna-se matematicamente significativa quando os estudantes compreendem a regra que implementam e relacionam os comandos ao raciocínio utilizado.

Dentre as várias ferramentas desenvolvidas para promover o Pensamento Computacional, a mais utilizada é o ambiente Scratch. Mais do que uma ferramenta, ele é um catalisador para a criatividade e a inovação, permitindo que as novas gerações explorem o potencial da tecnologia para dar vida às suas ideias.

O *Scratch* é uma linguagem de programação visual baseada em blocos, desenvolvida pelo grupo Lifelong Kindergarten, do MIT Media Lab. Embora seu desenvolvimento tenha começado em 2003, a plataforma foi lançada publicamente em 2007, com o propósito de tornar a programação mais acessível, criativa e atrativa, especialmente para crianças e jovens (Resnick *et al.*, 2009). Desde então, passou por diversas atualizações, entre as quais se destaca a

versão 3.0, lançada em 2019, que ampliou seus recursos e possibilidades de criação.

A interface intuitiva do *Scratch* permite desenvolver projetos interativos, como jogos, animações e histórias, por meio do encaixe de blocos que representam comandos de programação. Essas características contribuíram para sua disseminação em contextos educacionais, nos quais a ferramenta pode favorecer a aprendizagem de conceitos de programação e a construção de projetos autorais (Maloney *et al.*, 2010).

O *Scratch* foi concebido para favorecer uma aprendizagem na qual os estudantes participem ativamente da elaboração de projetos que tenham significado para eles. Essa perspectiva valoriza a exploração, a experimentação e a criação como elementos do processo educativo. Ao contrário das abordagens tradicionais que separam a rigidez científica do pensamento criativo, Papert (1994) defende que a tecnologia deve servir como um suporte para a imaginação infantil. Ao programar, os estudantes não apenas exercitam conceitos matemáticos, mas utilizam o computador como um recurso de expressão artística e de concretização das suas fantasias, integrando a afetividade, o brincar e o raciocínio lógico no próprio processo de aprendizagem.

Essa reflexão contribui para compreender o potencial educativo do *Scratch*: ao elaborar histórias, jogos e animações, os estudantes podem explorar situações imaginárias, tomar decisões e atribuir significados às próprias criações. Ao mesmo tempo, precisam organizar comandos, estabelecer regras e lidar com os resultados de suas escolhas. Assim, a dimensão lúdica da ferramenta pode ser

articulada ao desenvolvimento do raciocínio lógico, da criatividade e de estratégias para a resolução de problemas.

No contexto escolar, Barros (2020) destaca contribuições do *Scratch* para o engajamento dos estudantes e para a compreensão de conceitos matemáticos e de programação. A elaboração de projetos próprios favorece uma postura ativa diante da aprendizagem, pois os estudantes precisam formular ideias, experimentar soluções, identificar dificuldades e revisar suas construções. Nessa perspectiva, o recurso tecnológico não se limita à apresentação de conteúdos: constitui um ambiente no qual os conhecimentos podem ser mobilizados na criação e na resolução de problemas.

Além das possibilidades relacionadas à programação, o *Scratch* pode contribuir para o desenvolvimento de habilidades matemáticas. A elaboração de um algoritmo exige compreender o problema proposto, identificar as informações relevantes e organizar uma sequência de procedimentos capaz de produzir o resultado esperado. Esse processo pode favorecer a construção de modelos, a análise de relações e a compreensão dos conceitos envolvidos (Raabe; Zorzo; Blikstein, 2020).

No presente trabalho, essas potencialidades são exploradas por meio da construção de um programa para verificar os dígitos verificadores do Cadastro de Pessoas Físicas (CPF). A atividade articula conceitos matemáticos, como operações aritméticas, divisibilidade e congruências, à elaboração de algoritmos e à programação em blocos. Dessa forma, o *Scratch* é utilizado como recurso para investigar uma aplicação da Matemática em situações cotidianas, tornando explícitas as relações entre o procedimento matemático e sua implementação computacional.

3. A MATEMÁTICA DOS DÍGITOS VERIFICADORES

O cálculo dos dígitos verificadores do CPF pode ser interpretado como um procedimento aritmético baseado na congruência módulo 11. Essa interpretação permite não apenas reproduzir o algoritmo, mas também investigar matematicamente algumas de suas propriedades e compreender que tipos de erros podem ou não ser detectados.

3.1. Cálculo dos Dígitos Verificadores

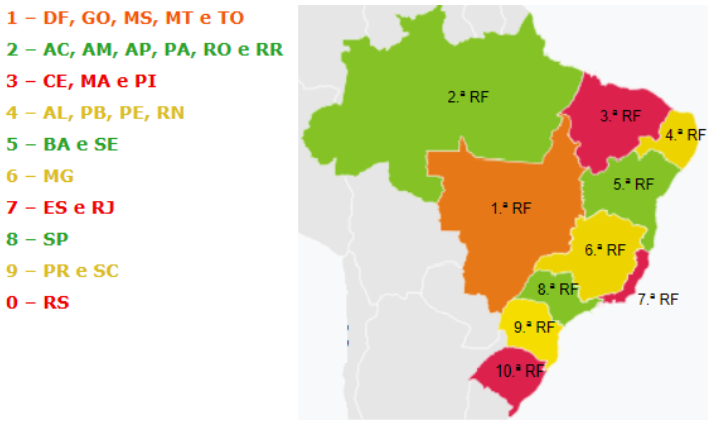
O CPF é apresentado por meio de uma sequência de onze algarismoss, usualmente representada na forma

$$a_1\,a_2\,a_3\,a_4\,a_5\,a_6\,a_7\,a_8\,a_9 \text{ - } d_1\,d_2\ .$$

Cada um dos onze dígitos obedece às seguintes regras:

- Os primeiros oito dígitos $a_1\,a_2\,a_3\,a_4\,a_5\,a_6\,a_7\,a_8$, formam o número-base definido pela Receita Federal no memento da inscrição;
- O nono dígito a_9 , define a Região Fiscal responsável pela inscrição (ver Figura 4);
- O penúltimo, d_1 , é o dígito verificador dos nove primeiros;
- O último, d_2 , é o dígito verificador dos nove anteriores a ele.

Figura 4. Número das regiões fiscais



Fonte: Clubes de Matemática da OBMEP (2020).

No caso da 10ª Região Fiscal (Rio Grande do Sul), o algarismo zero é utilizado como nono dígito. Podem existir casos específicos em que esse nono dígito não esteja de acordo com os determinados acima. Definidos os dígitos que formam o número-base e o dígito relativo à região fiscal, falta definir os dígitos verificadores. É aqui que a matemática entra. De maneira geral, Dígitos Verificadores (DV) são dígitos incorporados a números para possibilitar a detecção de erros de digitação, são as informações redundantes inseridas na informação inicial com o objetivo de detectar e corrigir erros. Vale ressaltar que uma vez detectado o erro, nem sempre será possível corrigi-lo. Isto depende do tipo de código corretor de erros utilizado.

Vamos entender agora como são definidos os DV. Considere um número de CPF representado inicialmente pelos nove algarismos

$$a_1\, a_2\, a_3\, a_4\, a_5\, a_6\, a_7\, a_8\, a_9\,,$$

aos quais serão acrescentados os dígitos verificadores d_1 e d_2. Assim o CPF completo pode ser escrito como

$$a_1\, a_2\, a_3\, a_4\, a_5\, a_6\, a_7\, a_8\, a_9 \text{ - } d_1\, d_2\,.$$

Para determinar d_1, associa-se aos nove primeiros algarismos a sequência de pesos

$$10; 9; 8; 7; 6; 5; 4; 3; 2\,.$$

Define-se, então

$$S_1 = 10 \times a_1 + 9 \times a_2 + 8 \times a_3 + 7 \times a_4 + 6 \times a_5 + 5 \times a_6 + 4 \times a_7 + 3 \times a_8 + 2 \times a_9\,.$$

Se r_1 é o resto da divisão de S_1 por 11, então:

$$S_1 \equiv r_1 \ (mod\ 11),\ 0 \leq r_1 \leq 10\,.$$

O primeiro dígito verificador é definido por:

$$d_1 = \begin{cases} 0, & se\ r_1 = 0\ ou\ r_1 = 1 \\ 11 - r_1, & se\ 2 \leq r_1 \leq 10 \end{cases}$$

Note que se:

$d_1 = 11 - r_1$, com $2 \leq r_1 \leq 10$,

então:

$$S_1 + d_1 \equiv r_1 + (11 - r_1) \equiv 0\ (mod\ 11)\,.$$

Ou seja,

$$S_1 + d_1 \equiv 0\ (mod\ 11)\,.$$

Essa relação mostra que, nos casos em que $r_1 \neq 0$ ou $r_1 \neq 1$, o dígito verificador é justamente o número que deve ser acrescentado à soma ponderada para que o resultado seja múltiplo de 11.

Nos casos $r_1 = 0$ e $r_1 = 1$, a regra estabelece $d_1 = 0$. Para $r_1 = 0$, ainda temos:

$$S_1 + d_1 \equiv 0 \ (mod\ 11),$$

enquanto, para $r_1 = 1$,

$$S_1 + d_1 \equiv 1 \ (mod\ 11).$$

Essa exceção é importante e terá consequências na análise de erros.

O segundo dígito verificador é obtido por um procedimento análogo, mas agora o primeiro dígito verificador participa do cálculo. Utilizam-se os pesos

$$10; 9; 8; 7; 6; 5; 4; 3; 2$$

sobre os algarismos

$$a_2\, a_3\, a_4\, a_5\, a_6\, a_7\, a_8\, a_9\, d_1\ .$$

Assim, definimos S_2 como:

$$S_2 = 10 \times a_2 + 9 \times a_3 + 8 \times a_4 + 7 \times a_5 + 6 \times a_6 + 5 \times a_7 + 4 \times a_8 + 3 \times a_9 + 2 \times d_1\ .$$

Se r_2 representa o resto da divisão de S_2 por 11, então:

$$S_2 \equiv r_2 \ (mod\ 11),$$

e define-se

$$d_2 = \begin{cases} 0, & se\ r_2 = 0\ ou\ r_2 = 1 \\ 11 - r_2, & se\ 2 \leq r_2 \leq 10 \end{cases}$$

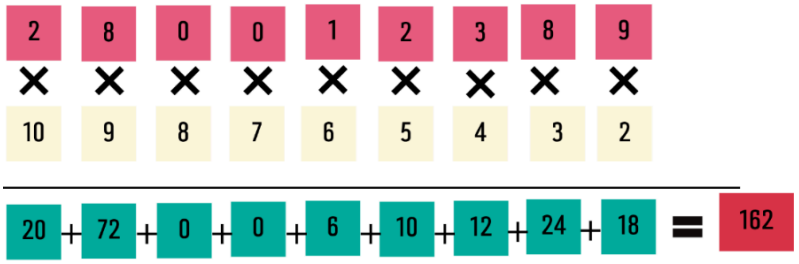
Novamente, quando $2 \leq r_2 \leq 10$,

$$S_2 + d_2 \equiv 0 \ (mod\ 11)$$

Exemplo 1 - Para entender a regra, veja o cálculo dos dígitos verificadores para o CPF de número $280.012.389$ - XY .

- Cálculo do primeiro dígito verificador: Calculamos a primeira soma ponderada

Figura 5. Cálculo do primeiro dígito verificador

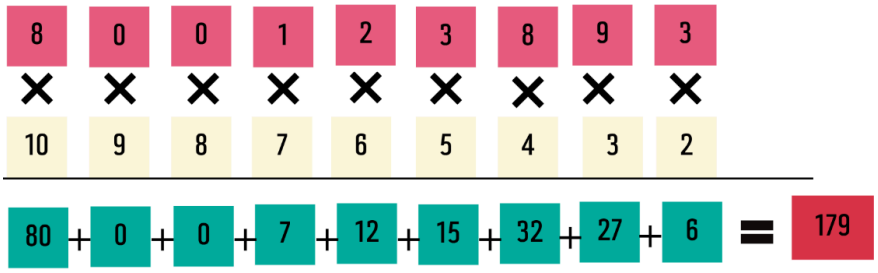


Fonte: Elaborada pelos autores (2026).

Como $162 = 11 \times 14 + 8$, isto é, $162 \equiv 8\ (mod\ 11)$, então o primeiro dígito verificador será $X = 11 - 8 = 3$.

- Cálculo do segundo dígito verificador: Calculamos a segunda soma ponderada

Figura 6. Cálculo do segundo dígito verificador



Fonte: Elaborada pelos autores (2026).

Como $179 = 11 \times 16 + 3$, isto é, $179 \equiv 3\ (mod\ 11)$, então o segundo dígito verificador será $Y = 11 - 3 = 8$.

Pronto! Já temos o CPF completo $280.012.389$ - 38 .

Observe que, no cálculo do segundo dígito verificador, o primeiro algarismo da sequência original pode ser omitido da soma utilizada para determinar o resto módulo 11. Isso ocorre porque, no procedimento usual, esse algarismo é multiplicado por 11, de modo que sua contribuição é múltipla de 11 e não altera o resto da divisão por 11. Assim, é possível utilizar os algarismos da segunda à nona posição e o primeiro dígito verificador, associados aos pesos de 10 a 2, respectivamente.

Portanto, o cálculo dos dois dígitos verificadores pode ser visto como aplicação sucessiva de uma mesma regra aritmética baseada no resto da divisão por 11. O aspecto mais importante, entretanto, não é apenas encontrar os dois algarismos. É perceber que existe uma regra matemática que relaciona os algarismos da sequência. É justamente essa regra que será posteriormente transformada em algoritmo.

3.2. Investigando a Detecção de Erros

Uma questão natural para uma atividade investigativa é: **Se um algarismo de um CPF for digitado incorretamente, o algoritmo necessariamente identificará o erro?**

A resposta é **NÃO.** O algoritmo detecta muitos erros, mas não todos. Essa conclusão pode ser obtida diretamente a partir das congruências utilizadas no cálculo.

Considere uma alteração em um dos nove primeiros algarismos. Suponha que

$$a_i \longrightarrow a_i + \delta_i,$$

com $\delta_i \neq 0$.

Se w_i é o peso associado à posição i, a nova soma será:

$$S_1^{'} = S_1 + w_i \delta_i .$$

Consequentemente,

$$S_1^{'} \equiv S_1 + w_i \delta_i \ (mod\ 11).$$

Como $w_i \not\equiv 0\ mod(11)$, uma alteração não nula do algarismo provoca, necessariamente, uma alteração no resto módulo 11. Isso significa que o resto muda. Entretanto, o dígito verificador pode permanecer igual, pois os restos 0 e 1, produzem ambos o dígito verificador 0.

Por exemplo, considere os nove primeiros algarismos

$$100000001.$$

O primeiro resto é

$$S_1 \equiv 1\ (mod\ 11),$$

e, portanto, $d_1 = 0$.

Se o primeiro algarismo 1 for alterado para 2, obtemos

$$200000001.$$

Nesse caso,

$$S_1^{'} \equiv 0\ (mod\ 11),$$

e novamente $d_1^{'} = 0$.

Como a alteração ocorreu no primeiro algarismo, a soma utilizada para calcular o segundo dígito não é modificada. Consequentemente, o segundo dígito também permanece inalterado. Assim, temos uma situação em que $100000001-08$ e $200000001-08$ satisfazem o mesmo procedimento de verificação, embora o primeiro algarismo tenha sido alterado.

Esse exemplo é particularmente interessante para a sala de aula porque mostra que uma afirmação aparentemente intuitiva de que o dígito verificador detecta qualquer erro não é verdadeira.

3.3. Erros nos Próprios Dígitos Verificadores

Há situações em que a detecção de erros é imediata. Se os nove primeiros algarismos permanecerem inalterados e apenas d_1 for modificado, o valor calculado para o primeiro dígito continuará sendo o original. Portanto, o valor informado não coincidirá com o valor calculado, e o erro será detectado. O mesmo ocorre se apenas d_2 for alterado. Assim, para os erros isolados nos próprios dígitos verificadores, temos que qualquer alteração isolada em d_1 ou d_2 é detectada. A situação se torna mais interessante quando um ou mais dos nove primeiros algarismos também são modificados, pois nesse caso os dígitos verificadores podem ser recalculados e, em determinadas situações, coincidir novamente com os valores armazenados.

3.4. Erros por Troca de Posições

Outro erro frequente de digitação ocorre quando dois algarismos são trocados de posição. Suponha que a_i e a_j sejam permutados.

Se os pesos correspondentes forem w_i e w_j, a alteração produzida na primeira soma ponderada é

$$w_i a_j + w_j a_i .$$

Reorganizando,

$$(w_i - w_j)(a_j - a_i) .$$

Logo,

$$S_1^{'} \equiv S_1 + (w_i - w_j)(a_j - a_i) \ (mod\ 11) .$$

Essa expressão permite investigar quando uma troca será detectada. Se

$$(w_i - w_j)(a_j - a_i) \neq 0 \ (mod\ 11),$$

o resto da soma será alterado. Porém, novamente, isso não garante sozinho a detecção, pois pode ocorrer a passagem de um resto 0 para um resto 1, ou de 1 para 0, mantendo o mesmo dígito verificador. Além disso, como o segundo dígito é calculado a partir de uma segunda soma ponderada, uma troca pode ser detectada nessa segunda etapa mesmo quando não é identificada pela primeira.

Por exemplo, o procedimento do CPF atribui dígitos verificadores 00 aos nove primeiros algarismos

$$123456797 .$$

Se os dois primeiros algarismos forem trocados, obtemos

Aplicando novamente o algoritmo, os dígitos verificadores permanecem 00. Portanto, do ponto de vista exclusivamente desse procedimento, a troca não é detectada. Esse exemplo reforça uma ideia importante para a proposta didática: um mecanismo de detecção de erros não precisa detectar todos os erros possíveis para ser matematicamente útil. Sua eficiência depende das propriedades da regra adotada e dos tipos de erros considerados.

4. PROPOSTA DE SEQUÊNCIA DIDÁTICA

A sequência didática proposta destina-se prioritariamente a estudantes do Ensino Médio, podendo ser adaptada aos anos finais do Ensino Fundamental de acordo com os conhecimentos prévios da turma. Estima-se a duração de nove a dez aulas de 50 minutos, distribuídas entre atividades de investigação matemática, cálculo manual, elaboração de algoritmo, programação em blocos e análise dos resultados. O planejamento pode ser ajustado conforme o ritmo de aprendizagem dos estudantes e a disponibilidade de recursos tecnológicos.

As atividades podem ser realizadas individualmente ou em pequenos grupos, de modo a favorecer a comparação de estratégias, a discussão dos procedimentos e a construção coletiva de soluções. Para a implementação no Scratch, recomenda-se o uso de computadores com acesso ao ambiente de programação ou à internet, admitindo-se o compartilhamento dos equipamentos quando necessário. Todos os exemplos numéricos utilizados nas atividades devem ser fictícios, sem a coleta ou a divulgação de números de CPF reais. O tempo deve ser ajustado aos conhecimentos prévios da turma.

Quadro 1. Etapas da sequência didática proposta

Etapa	Objetivo	Atividade central	Produção	Tempo
1. Problematização	Levantar hipóteses.	Discutir como identificar uma digitação incorreta em sequências fictícias.	Hipóteses iniciais registradas.	1 aula
2. Cálculo manual	Compreender pesos e restos.	Calcular o primeiro dígito e comparar estratégias.	Cálculos comentados.	1–2 aulas
3. Generalização	Sistematizar a regra.	Calcular o segundo dígito e registrar os passos.	Regra geral escrita pelos estudantes.	1 aula
4. Investigar erros	Analisar limites.	Alterar algarismos, trocar posições e comparar os dígitos.	Registro dos testes e das conclusões.	1–2 aulas
5. Algoritmo	Ordenar instruções.	Produzir pseudocódigo e fluxograma.	Pseudocódigo e fluxograma.	1 aula
6. Scratch	Implementar a regra.	Criar variáveis, repetições, operadores e condicionais.	Programa funcional e registros de depuração.	2 aulas
7. Testes	Verificar e comunicar.	Testar entradas e explicar conclusões.	Registro de testes e explicação final.	1 aula

Fonte: Elaborado pelos autores (2026).

A avaliação pode ocorrer ao longo de toda a sequência didática, considerando tanto os procedimentos utilizados quanto os resultados obtidos. Na etapa inicial, podem ser observadas as hipóteses formuladas pelos estudantes; durante os cálculos, a compreensão das operações e a justificativa das estratégias; na elaboração do algoritmo, a organização lógica das instruções; e na programação, a correspondência entre a regra matemática e os comandos implementados. Os registros de testes, a identificação e a correção de erros e a explicação dos limites do programa também constituem evidências de aprendizagem. Recomenda-se combinar a observação do trabalho em grupo com a análise das produções individuais, de modo a identificar dificuldades específicas e orientar intervenções do professor.

4.1. Articulação da Sequência Didática com a BNCC

A sequência didática proposta articula conhecimentos matemáticos e computacionais por meio da investigação dos dígitos verificadores do Cadastro de Pessoas Físicas (CPF). Ao partir de uma situação-problema, os estudantes formulam hipóteses, realizam cálculos, identificam regularidades, organizam procedimentos e avaliam os resultados obtidos. A proposta busca, assim, integrar a compreensão matemática à elaboração de algoritmos e à programação em blocos, sem reduzir a atividade à execução de comandos.

No que se refere às competências gerais da Base Nacional Comum Curricular (BNCC), a proposta relaciona-se à Competência Geral 2, ao estimular a curiosidade intelectual, a investigação, a reflexão e a análise crítica; à Competência Geral 4, ao mobilizar diferentes formas

de representação e comunicação, como registros matemáticos, pseudocódigo, fluxogramas e explicações orais ou escritas; e à Competência Geral 5, ao envolver a utilização e a criação de soluções com tecnologias digitais de maneira crítica e reflexiva (Brasil, 2018). Essas competências são mobilizadas ao longo da investigação, da elaboração do algoritmo, da implementação no Scratch e da discussão dos limites do programa.

Na área de Matemática e suas Tecnologias, destaca-se a habilidade EM13MAT315, que propõe investigar e registrar, por meio de um fluxograma, quando possível, um algoritmo que resolva um problema (Brasil, 2018). Essa habilidade apresenta relação direta com a quinta etapa da sequência, na qual os estudantes organizam o procedimento de cálculo dos dígitos verificadores em pseudocódigo e fluxograma, antes de implementá-lo no ambiente de programação. A proposta também dialoga com as Competências Específicas 1 e 5 da área, na medida em que envolve a interpretação de uma situação-problema por meio de conceitos e procedimentos matemáticos e a investigação de relações matemáticas com apoio de tecnologias digitais.

A proposta também se relaciona ao complemento à BNCC referente à Computação, aprovado pelo Conselho Nacional de Educação em 2022 (Brasil, 2022). No Ensino Médio, a habilidade EM13CO01 pode ser mobilizada caso os estudantes reutilizem e adaptem soluções ou estruturas de programação já conhecidas. A habilidade EM13CO02 relaciona-se ao refinamento progressivo da solução, desde a compreensão do problema e a elaboração do algoritmo até sua implementação e depuração no Scratch. Já a habilidade EM13CO05 permite discutir os limites da automatização, especialmente ao analisar que a consistência matemática de uma sequência não

comprova a existência de um CPF nem sua situação cadastral e que o mecanismo não detecta necessariamente todas as alterações possíveis.

A articulação com a BNCC deve ser compreendida como uma possibilidade de desenvolvimento das aprendizagens previstas, e não como uma garantia automática de sua realização. A aprendizagem depende da participação dos estudantes, da mediação docente e das oportunidades de formular hipóteses, justificar procedimentos, testar soluções e revisar os resultados. Por isso, as produções previstas na sequência — cálculos, registros investigativos, pseudocódigo, fluxograma, programa e análise dos testes — podem servir como evidências para acompanhar o desenvolvimento das aprendizagens matemáticas e computacionais.

Dessa maneira, a sequência didática promove uma abordagem interdisciplinar, na qual a Matemática constitui a base para a compreensão do procedimento de cálculo, enquanto a Computação oferece recursos para sua sistematização e automatização. O *Scratch* funciona, nesse contexto, como ambiente de experimentação e construção, favorecendo a participação ativa dos estudantes e a compreensão das relações entre algoritmo, representação matemática e programa computacional.

4.2. Problematização e Exploração

O professor pode iniciar perguntando: “Se alguém digitar incorretamente um algarismo de um identificador, como o sistema poderia perceber?”. Os estudantes apresentam hipóteses antes de conhecer a regra. Por privacidade, devem ser usados apenas

números inventados para a atividade; ninguém deve informar ou inserir o próprio CPF.

Em duplas ou pequenos grupos, os estudantes registram algarismos, pesos, produtos, soma e resto. O objetivo não é apenas obter o resultado, mas identificar a sequência de operações e explicar por que ela pode ser repetida. Em seguida, alteram um algarismo ou trocam duas posições e registram se os dígitos calculados mudam. Uma tabela pode incluir sequência inicial, alteração realizada, dígitos calculados, detecção e justificativa.

4.3. Do Cálculo Ao Algoritmo

Antes de abrir o Scratch, a turma deve descrever o procedimento em linguagem natural e depois em pseudocódigo. Uma organização possível é:

1. Receber uma sequência de onze algarismos.

2. Verificar se a entrada tem onze algarismos e contém somente dígitos.

3. Calcular a primeira soma ponderada S_1 usando os nove primeiros algarismos e os pesos 10, 9, 8, 7, 6, 5, 4, 3 e 2.

4. Calcular o resto r_1 da divisão de S_1 por 11 e determinar o primeiro dígito verificador d_1, atribuindo $d_1 = 0$ quando $r_1 = 0$ ou $r_1 = 1$, e $d_1 = 11 - r_1$ nos demais casos.

5. Calcular a segunda soma ponderada S_2 com os algarismos da segunda à nona posição e o primeiro dígito calculado e os pesos 10, 9, 8, 7, 6, 5, 4, 3 e 2.

6. Calcular o resto r_2 da divisão de S_2 por 11 e determinar o segundo dígito verificador d_2, aplicando a mesma regra: atribuindo $d_2 = 0$ quando $r_2 = 0$ ou $r_2 = 1$, e $d_2 = 11 - r_2$ nos demais casos.

7. Informar se a sequência é consistente com a regra matemática.

No fluxograma, devem ser diferenciadas entrada/saída, processamento e decisão. Losangos podem representar as condições “o resto é 0 ou 1?” e “os dígitos calculados coincidem com os informados?”. É necessário verificar o formato antes do cálculo para que entradas inválidas não sejam processadas como se fossem sequências corretas.

Assim, a utilização de uma representação intermediária entre a Matemática e o programa pode favorecer a compreensão da estrutura do algoritmo. O estudante precisa decidir quais informações devem ser armazenadas, quais operações devem ser repetidas e em que momentos determinadas decisões precisam ser tomadas. Esta passagem evidencia que programar não consiste simplesmente em escrever comandos. Antes de programar, é necessário compreender e organizar a estratégia de resolução do problema. Compreendido o modo como são obtidos os dígitos verificadores do CPF, torna-se possível organizar esse procedimento em uma sequência de instruções executáveis. Essa mudança de representação é importante, pois o cálculo, antes realizado manualmente, passa a ser descrito passo a passo, de forma que cada ação possa ser reconhecida e posteriormente programada. No procedimento manual, o estudante identifica os algarismos, realiza as multiplicações pelos pesos correspondentes, soma os resultados e calcula o resto da divisão por 11. A partir desse resultado, aplica a

regra que determina o dígito verificador. Quando essas ações são colocadas em uma ordem definida, com início, desenvolvimento e resultado esperado, elas assumem a forma de um algoritmo. Essa organização exige mais do que repetir operações. E preciso compreender o que está sendo calculado, por que determinada operação ocorre em certo momento e de que maneira o resultado de uma etapa interfere na etapa seguinte. Esse aspecto é especialmente relevante antes da programação, porque o computador executa aquilo que foi previamente estruturado pelo usuário.

No *Scratch*, essa estrutura pode ser representada por meio de blocos que desempenham funções diferentes dentro do programa. Algumas informações precisam ser armazenadas, outras devem ser atualizadas ao longo da execução e determinadas ações precisam ser repetidas ou condicionadas a um resultado. Por esse motivo, aparecem no projeto elementos como variáveis, operadores, estruturas de repetição e comandos condicionais. Ao passar do cálculo escrito para a programação em blocos, o estudante torna mais visível a lógica envolvida na resolução. A sequência das operações, a alteração dos valores e as condições necessárias para chegar a uma resposta deixam de ficar apenas subentendidas e passam a fazer parte da própria estrutura do programa. E a partir dessa organização que se pode analisar, de maneira mais detalhada, como o procedimento matemático foi representado no *Scratch.*

4.4. Implementação no Scratch: Uma Progressão Didática

A implementação do algoritmo no Scratch foi estruturada em duas etapas progressivas, permitindo que a passagem do procedimento manual para a programação em blocos acompanhe o nível de

familiaridade dos estudantes com a ferramenta e com a lógica de programação.

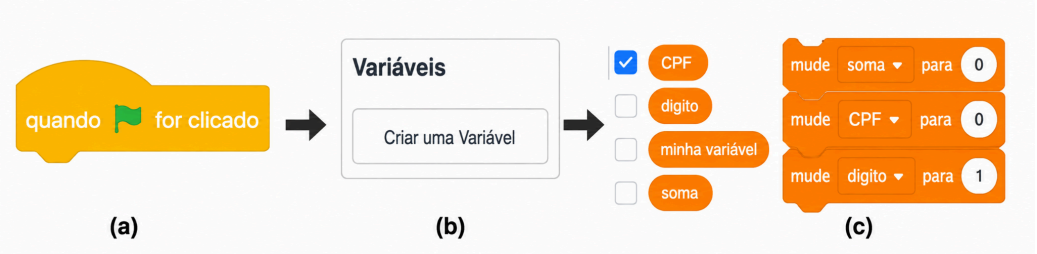
4.4.1. Nível Inicial: Leitura de Dados, Variáveis e Estrutura Básica

Em um primeiro momento, para turmas sem experiência prévia com programação, a atividade prioriza a compreensão dos comandos fundamentais e da estrutura do fluxo de dados: entrada, processamento e saída.

Nesta primeira etapa, o programa é apresentado como um protótipo didático destinado à aprendizagem das estruturas básicas de programação, como entrada de dados, utilização de variáveis, repetição e estruturas condicionais. Essa versão não constitui ainda um verificador completo dos dígitos do CPF. A implementação integral da regra matemática é desenvolvida na etapa seguinte, quando os estudantes articulam os procedimentos aritméticos estudados com sua representação algorítmica.

O início da programação ocorre com o bloco “quando a bandeira verde for clicada”, responsável por acionar a execução do algoritmo. Em seguida, são definidas as variáveis CPF, DÍGITO e SOMA, que recebem valores iniciais antes do processamento. Como se observa na Figura 7, essa etapa reúne o acionamento do programa, a criação das variáveis e a atribuição dos seus valores iniciais. Esse procedimento estabelece um estado inicial conhecido para o algoritmo e organiza as informações que serão modificadas durante sua execução.

Figura 7. Inicialização do programa e definição das variáveis no Scratch



Fonte: Elaborado pelos autores (2026).

Cada variável desempenha uma função própria dentro do programa. A variável CPF é utilizada para armazenar os algarismos manipulados durante a execução; DÍGITO indica a posição do algarismo que está sendo analisado; e SOMA conserva os valores obtidos durante o processamento. Essa distribuição permite separar diferentes funções dentro do algoritmo e acompanhar como cada informação se altera ao longo da execução.

Na etapa seguinte, o programa precisa receber o dado que será processado. Para isso, utiliza-se o bloco de pergunta do Scratch, que solicita ao usuário que digite o CPF e mantém a informação fornecida disponível por meio do bloco resposta. A Figura 8 apresenta essa passagem da pergunta genérica disponível no ambiente para a pergunta adaptada à proposta. A partir dessa entrada, o algoritmo deixa de trabalhar apenas com valores previamente definidos e passa a processar uma informação fornecida pelo próprio usuário.

Figura 8. Entrada de dados por meio do bloco de pergunta no Scratch

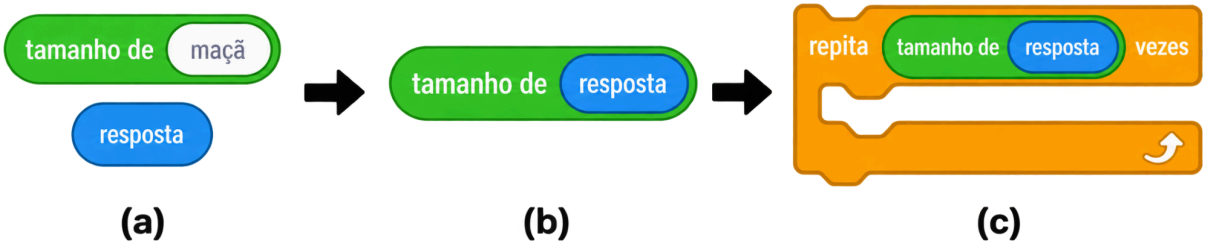


Fonte: Elaborado pelos autores (2026).

Essa interação é importante porque estabelece uma sequência bastante comum na construção de programas: entrada de dados, processamento e produção de um resultado. No Scratch, essas etapas são visualmente organizadas pela combinação entre blocos, o que permite acompanhar o fluxo da execução e as relações estabelecidas entre os comandos. Sápiras, Dalla Vecchia e Maltempi (2015) destacam justamente que, nesse ambiente, as ações precisam ser explicitadas e organizadas por meio de blocos conectados, fazendo com que o usuário represente sua estratégia de resolução na própria estrutura do programa.

Após a entrada do CPF, torna-se necessário definir quantas vezes o conjunto de instruções deverá ser executado. Inicialmente, o Scratch apresenta o bloco de repetição com um número fixo de ciclos. Entretanto, para que a quantidade de repetições dependa da informação fornecida pelo usuário, utiliza-se o operador “tamanho de [resposta]”. Como mostra a Figura 9, o valor fixo é substituído pelo tamanho da resposta, formando a estrutura “repita [tamanho de resposta] vezes”.

Figura 9. Construção da estrutura de repetição a partir do tamanho da resposta



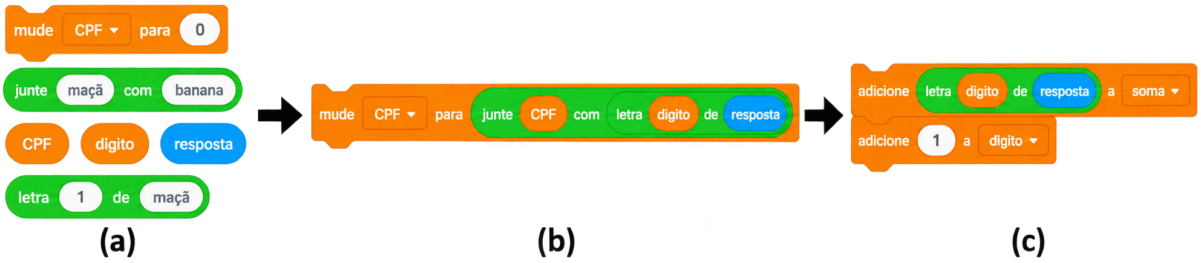
Fonte: Elaborado pelos autores (2026)

Essa alteração faz com que o número de execuções seja determinado pelo próprio dado de entrada. Assim, em vez de repetir uma sequência uma quantidade previamente estabelecida de vezes,

o algoritmo passa a considerar o número de caracteres presentes na resposta fornecida pelo usuário. A estrutura repita constitui, portanto, um laço de repetição responsável por controlar a execução sucessiva das instruções inseridas em seu interior. Estruturas desse tipo são relevantes no Scratch por permitirem que uma mesma sequência de comandos seja reutilizada de forma controlada, evitando sua reprodução manual várias vezes.

No interior da repetição, o programa percorre sucessivamente os algarismos informados pelo usuário. Para isso, a variável DÍGITO é utilizada como referência para indicar qual posição da resposta está sendo analisada. Como se observa na Figura 10, o bloco “letra [DÍGITO] de [resposta]” permite recuperar um elemento da entrada de acordo com a posição indicada pela variável.

Figura 10. Processamento sequencial dos algarismos no interior da estrutura de repetição



Fonte: Elaborado pelos autores (2026).

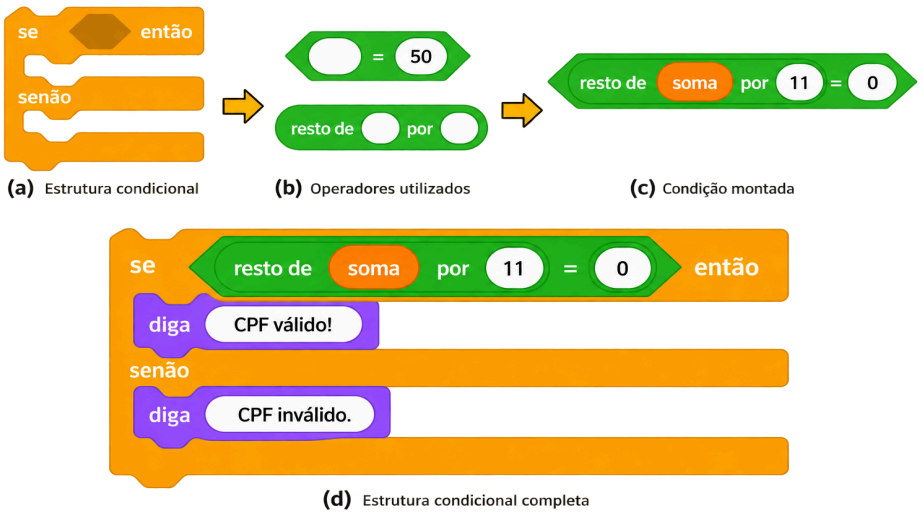
A cada passagem pelo laço, o algarismo correspondente é incorporado às operações do programa. Paralelamente, a variável DÍGITO é incrementada em uma unidade, fazendo com que, na repetição seguinte, uma nova posição da resposta seja acessada. Dessa forma, a leitura ocorre sequencialmente, sem a necessidade de criar um comando específico para cada algarismo.

Ainda no interior da repetição, a variável SOMA é utilizada para acumular os valores processados. Em vez de substituir o conteúdo armazenado anteriormente, o programa acrescenta um novo valor a cada ciclo. Esse mecanismo permite conservar resultados parciais enquanto a repetição ocorre. A própria Figura 10 mostra essa articulação entre a seleção do algarismo, sua utilização no processamento e a atualização das variáveis.

Do ponto de vista computacional, observa-se nessa etapa uma combinação entre entrada de dados, indexação, repetição, atualização de variáveis e acumulação de resultados. Essas estruturas não aparecem isoladamente: elas se articulam para produzir o comportamento esperado do algoritmo. Sápiras, Dalla Vecchia e Maltempi (2015) ressaltam que elementos lógico-computacionais presentes no Scratch podem atuar conjuntamente na construção e experimentação de soluções, o que aproxima a programação do processo de resolução de problemas.

Depois de concluída essa etapa de processamento, o algoritmo precisa tomar uma decisão a partir do resultado obtido. Para isso, utiliza-se uma estrutura condicional, representada no Scratch pelo bloco “se... então... senão”. Como se observa na Figura 11, a condição é construída a partir de operadores que comparam o resultado calculado com um critério previamente definido, direcionando a execução para respostas diferentes conforme o resultado da verificação. Com isso, o fluxo do programa deixa de ser exclusivamente sequencial e passa a admitir diferentes caminhos de execução.

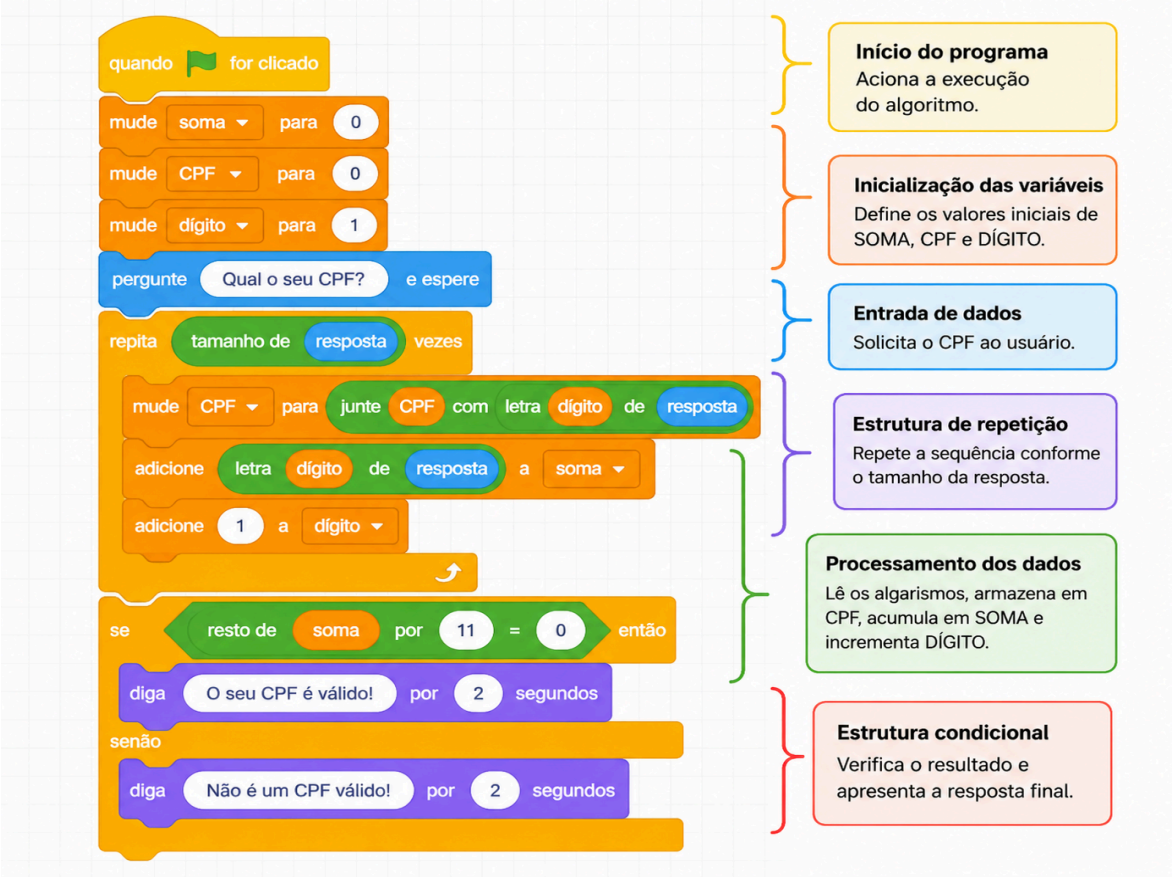
Figura 11. Construção da estrutura condicional para verificação do resultado



Fonte: Elaborado pelos autores no Scratch (2026).

Ao reunir todas as etapas apresentadas ao longo desta seção, obtém-se a programação completa desenvolvida no Scratch. Como se observa na Figura 12, o algoritmo integra, em uma única sequência, a inicialização das variáveis, a entrada de dados, a estrutura de repetição, o processamento dos algarismos e a condição responsável por direcionar a resposta final. Essa visualização permite compreender de forma conjunta como os diferentes blocos se articulam e como cada estrutura assume uma função específica dentro do fluxo do programa.

Figura 12. Programação completa do algoritmo desenvolvido no Scratch



Fonte: Elaborado pelos autores no Scratch (2026).

É importante destacar que essa primeira implementação possui caráter introdutório e não reproduz o procedimento matemático de verificação dos dígitos do CPF. Sua finalidade é explorar conceitos básicos de programação, como entrada de dados, variáveis, estruturas de repetição e condicionais. A condição utilizada para produzir a resposta final verifica apenas uma propriedade da soma dos algarismos, não sendo suficiente para determinar a consistência dos dígitos verificadores. Por esse motivo, o programa deve ser compreendido como um protótipo didático, cuja limitação constitui ponto de partida para a construção da versão avançada, baseada nas somas ponderadas e nas regras de cálculo dos dois dígitos verificadores.

4.4.2. Nível Avançado: Modularização, Listas e Somas Ponderadas

A versão avançada do programa amplia o protótipo inicial ao implementar o procedimento matemático de verificação dos dígitos

do CPF por meio de estruturas de dados, variáveis, repetições, condicionais e blocos personalizados. A solução é organizada em três procedimentos principais — Contagem de dígitos, Verificação de dígitos e Juntar dígitos —, coordenados pelo algoritmo principal.

4.4.2.1. Estrutura de Variáveis, Lista e Procedimentos

A organização do projeto, apresentada na Figura 13, envolve variáveis para armazenar a sequência informada, os dígitos verificadores calculados, as somas ponderadas, os restos das divisões e o índice utilizado nas repetições. A lista Dígitos armazena individualmente os algarismos da sequência, permitindo acessá-los nas posições necessárias durante os cálculos.

Os blocos personalizados permitem dividir o programa em etapas com funções específicas, facilitando sua compreensão, manutenção e depuração. Essa organização também favorece a discussão sobre decomposição de problemas e modularização, aspectos relevantes do pensamento computacional.

Figura 13. Variáveis, lista e procedimentos criados no ambiente Scratch



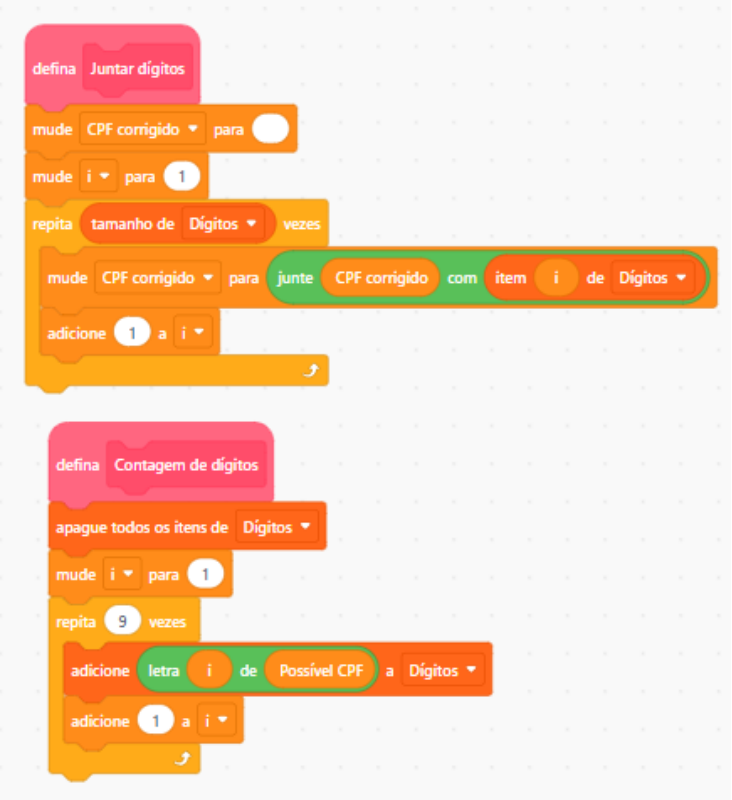
Fonte: Elaborado pelos autores (2026).

4.4.2.2. Procedimentos de Organização e Recomposição dos Dados

O procedimento Contagem de dígitos, apresentado na Figura 14, extrai os nove algarismos iniciais da sequência informada e os armazena na lista Dígitos. Para isso, utiliza uma estrutura de repetição que percorre as posições da entrada e adiciona cada caractere à lista, preservando a ordem original.

Após o cálculo dos dígitos verificadores, o procedimento Juntar dígitos percorre os elementos da lista e concatena seus valores em uma única sequência, armazenada na variável CPF corrigido. Essa sequência é utilizada na etapa final de comparação com a entrada fornecida pelo usuário.

Figura 14. Procedimentos 'Contagem de dígitos' e 'Juntar dígitos' no Scratch.



Fonte: Elaborado pelos autores (2026).

4.4.2.3. Implementação dos Cálculos dos Dígitos Verificadores

O procedimento Verificação de dígitos, ilustrado na Figura 15, reúne as operações necessárias para calcular os dois dígitos verificadores. As somas ponderadas são implementadas por meio de estruturas de repetição, nas quais os algarismos armazenados na lista são multiplicados pelos pesos correspondentes e os produtos são acumulados em uma variável.

Ao término de cada repetição, o programa calcula o resto da divisão por 11 e aplica a regra matemática apresentada anteriormente para determinar o dígito verificador correspondente. O primeiro dígito calculado é acrescentado à lista, de modo que possa ser utilizado na etapa seguinte. Após o cálculo do segundo dígito, ambos são reunidos aos nove algarismos iniciais, permitindo reconstruir a sequência para a comparação final.

A implementação exige atenção à inicialização das variáveis e à atualização do índice em cada repetição. Também é necessário garantir que o segundo laço percorra as posições corretas da lista e acumule os produtos antes do cálculo do resto. Esses aspectos são importantes para que os blocos executem o procedimento matemático descrito nas seções anteriores.

Figura 15. Procedimento 'Verificação de Dígitos' completo no Scratch



Fonte: Elaborado pelos autores (2026).

4.4.2.4. Fluxo de Controle e Validação Principal

O algoritmo principal, apresentado na Figura 16, coordena a execução do programa. Inicialmente, solicita ao usuário uma sequência e verifica se ela contém exatamente onze caracteres, todos numéricos. Caso a entrada não atenda a essas condições, o programa deve informar que ela é inválida e encerrar a execução antes dos cálculos. Essa verificação também exige que a entrada seja tratada como texto, a fim de preservar eventuais zeros à esquerda.

Figura 16. Algoritmo principal de validação e controle no Scratch



Fonte: Elaborado pelos autores (2026).

Quando a entrada está no formato esperado, o algoritmo aciona os procedimentos de extração e cálculo dos dígitos verificadores e, em seguida, reconstrói a sequência. Por fim, compara a sequência reconstruída com a informada pelo usuário. Se ambas coincidirem, apresenta a mensagem “CPF existe”; caso contrário, informa que a

sequência é inconsistente com a regra dos dígitos verificadores, apresentando a mensagem “CPF não existe”.

É importante ressaltar que a consistência matemática não comprova a existência de um CPF nem sua situação cadastral perante a Receita Federal. O programa verifica apenas a correspondência entre os dígitos informados e aqueles calculados segundo o procedimento implementado.

A passagem do protótipo inicial para a versão avançada permite explorar a decomposição de problemas, a organização de dados, a construção de procedimentos e a depuração de programas. Essa abordagem pode ser relacionada às habilidades EM13CO02, referente ao refinamento e à organização de soluções computacionais, e EM13CO05, relacionada à análise dos limites da computação. A habilidade EM13CO01 também pode ser mobilizada quando os estudantes reutilizam e adaptam estruturas ou soluções previamente desenvolvidas (Brasil, 2022).

Exemplo 2: Ao alterar o primeiro dígito verificador da sequência utilizada no Exemplo 1, o programa identifica a inconsistência entre o dígito informado e o valor calculado, conforme pode ser visto na Figura 17.

Figura 17. Exemplo aplicado no Scratch



Fonte: Elaborada pelos autores (2026).

Dessa forma, a implementação no Scratch constitui uma etapa de tradução e teste do procedimento matemático, possibilitando aos estudantes analisar a correspondência entre os cálculos realizados, as instruções programadas e os resultados apresentados[5].

4.5. Testes e Avaliação

A etapa de testes tem como objetivo verificar a correspondência entre o procedimento matemático, o algoritmo elaborado e sua implementação no Scratch. Para isso, recomenda-se utilizar sequências fictícias, cujos dígitos verificadores possam ser calculados manualmente, evitando o uso de dados pessoais reais.

Os testes devem contemplar diferentes situações: sequências matematicamente consistentes com a regra dos dígitos verificadores; sequências com um ou ambos os dígitos verificadores alterados; entradas com quantidade incorreta de caracteres; e entradas que contenham letras ou símbolos não numéricos. Também podem ser investigadas alterações nos algarismos iniciais, com o objetivo de analisar quais erros são detectados pelo procedimento e discutir suas limitações.

A realização dos testes deve ser acompanhada da comparação entre os resultados obtidos manualmente e aqueles apresentados pelo programa. Quando houver divergência, os estudantes podem revisar os cálculos, verificar as condições lógicas e examinar as estruturas de repetição, identificando possíveis erros na implementação.

A avaliação pode considerar a compreensão do procedimento matemático, a organização e a justificativa dos cálculos, a elaboração do algoritmo ou fluxograma, a correspondência entre as regras matemáticas e os blocos do Scratch, a capacidade de testar e depurar o programa e a clareza na comunicação dos resultados. Dessa forma, a avaliação contempla não apenas a resposta produzida pelo programa, mas também o processo de raciocínio desenvolvido pelos estudantes.

Por se tratar de uma proposta didática, e não de um relato de intervenção empírica, esses critérios constituem possibilidades de acompanhamento da aprendizagem. Sua aplicação em sala de aula poderá fornecer elementos para analisar as dificuldades encontradas pelos estudantes e aperfeiçoar a sequência didática.

4.6. Discussão das Possibilidades Didáticas

A principal contribuição da proposta está na articulação entre diferentes representações de um mesmo problema: a situação contextualizada, os cálculos matemáticos, a formulação de uma regra geral, o algoritmo, o fluxograma e o programa computacional. A transição entre essas representações exige que os estudantes identifiquem informações relevantes, organizem procedimentos e compreendam como as operações e condições lógicas se relacionam na construção de uma solução.

A atividade oferece possibilidades para explorar conteúdos como multiplicação, divisão com resto, reconhecimento de regularidades, generalização e linguagem algébrica. Conforme a escolaridade e os conhecimentos prévios da turma, também pode favorecer a introdução ou o aprofundamento das congruências módulo 11. Nos anos finais do Ensino Fundamental, o professor pode priorizar a investigação de padrões, os cálculos e a organização de procedimentos; no Ensino Médio, pode ampliar a discussão por meio da formalização matemática e da análise dos limites do mecanismo de detecção de erros.

A utilização do Scratch permite examinar a relação entre uma regra matemática e sua implementação computacional. Ao testar o programa e analisar os resultados, os estudantes podem identificar erros de programação, revisar procedimentos e compreender que a execução correta de um código não garante, por si só, a adequação da solução ao problema proposto. Nesse sentido, a depuração constitui uma oportunidade para retomar o raciocínio matemático e avaliar a correspondência entre os cálculos e as instruções programadas.

Outra possibilidade didática consiste em discutir os limites dos dígitos verificadores. Embora o mecanismo permita detectar determinadas alterações em sequências numéricas, nem todos os erros são necessariamente identificados. Além disso, a consistência matemática da sequência não comprova a existência de um CPF nem sua situação cadastral. Essa discussão contribui para uma compreensão crítica do alcance das soluções computacionais. Também é possível ampliar a discussão para outras sequências numéricas que utilizam os dígitos verificadores, explorando outras regras matemáticas e as implementando no Scratch.

Por se tratar de uma proposta didática ainda não avaliada empiricamente, as contribuições apresentadas correspondem a possibilidades fundamentadas na articulação entre investigação matemática, resolução de problemas e programação. A aplicação da sequência em turmas da Educação Básica poderá fornecer elementos para analisar as dificuldades dos estudantes, as estratégias utilizadas e as adaptações necessárias ao contexto escolar.

5. CONSIDERAÇÕES FINAIS

Este trabalho apresentou uma proposta de sequência didática para abordar a detecção de erros por meio dos dígitos verificadores do Cadastro de Pessoas Físicas (CPF), articulando conhecimentos matemáticos, pensamento computacional e programação em blocos. A proposta permite explorar somas ponderadas, divisões com resto, congruências módulo 11 e condições lógicas, relacionando a formalização matemática à elaboração de algoritmos e à sua implementação no Scratch.

A sequência didática foi organizada de modo a favorecer a investigação, a formulação de hipóteses, a realização de cálculos, a elaboração de procedimentos e a análise dos resultados obtidos. Nesse contexto, o Scratch é utilizado como recurso para implementar e testar uma regra matemática, e não como finalidade isolada. A identificação e a correção de erros na programação também oferecem oportunidades para retomar os conceitos envolvidos e examinar a coerência entre o procedimento matemático e o código desenvolvido.

Outro aspecto relevante é a discussão das limitações dos dígitos verificadores. Embora o mecanismo permita detectar determinadas alterações em sequências numéricas, ele não identifica necessariamente todos os erros. Além disso, a consistência matemática de uma sequência não comprova a existência de um CPF nem sua situação cadastral. A abordagem dessas limitações contribui para uma compreensão mais crítica das possibilidades e do alcance das soluções computacionais.

Por se tratar de uma proposta didática ainda não avaliada empiricamente, as contribuições apresentadas correspondem a possibilidades pedagógicas fundamentadas na articulação entre resolução de problemas, investigação matemática e tecnologias digitais. Como perspectiva de continuidade, propõe-se a aplicação da sequência em turmas da Educação Básica, acompanhada da análise das produções dos estudantes, das dificuldades encontradas durante a implementação dos algoritmos e das estratégias utilizadas na resolução dos problemas. Esses registros poderão subsidiar o aperfeiçoamento da proposta e investigações futuras sobre as relações entre aprendizagem matemática e pensamento computacional.

REFERÊNCIAS BIBLIOGRÁFICAS

BARROS, Taiser Tadeu Teixeira. **Formação em Pensamento Computacional utilizando Scratch para Professores de Matemática e Informática da Educação Fundamental.** 2020. Tese (Doutorado em Informática na Educação) - Universidade Federal do Rio Grande do Sul, Porto Alegre, 2020. Disponível em: https://lume.ufrgs.br/bitstream/handle/10183/219412/001124028.pdf?sequence=1&isAllowed=y. Acesso em: 22 set. 2026.

BORBA, Marcelo de Carvalho; PENTEADO, Miriam Godoy. **Informática e educação matemática.** 4. ed. Belo Horizonte: Autêntica, 2010.

BRASIL. Ministério da Educação. **Base Nacional Comum Curricular.** Brasília: MEC, 2018. Disponível em: https://www.gov.br/mec/pt-br/escola-em-tempo-integral/BNCC_EI_EF_110518_versaofinal.pdf. Acesso em: 12 out. 2026.

BRASIL. Ministério da Educação. **Computação: Complemento à BNCC.** Brasília: MEC, 2022. Disponível em: http://portal.mec.gov.br/docman/fevereiro-2022-pdf/236791-anexo-ao-parecer-cneceb-n-2-2022-bncc-computacao/file. Acesso em: 12 out. 2026.

CALDEIRA, Jhone.; MATOS, Rayanne Auxiliadora de Oliveira. Códigos Corretores de Erros e uma breve introdução aos códigos de Hamming. **REVISTA FOCO,** *[S. l.]*, v. 16, n. 6, 2023. DOI: 10.54751/revistafoco.v16n6-098. Disponível em: https://ojs.focopublicacoes.com.br/foco/article/view/2392. Acesso em: 12 out. 2026.

CLUBES DE MATEMÁTICA DA OBMEP. Sala de atividades: a Matemática nos documentos. 2020. Disponível em: https://clubes.obmep.org.br/blog/sala-de-atividades-a-matematica-nos-documentos/. Acesso em: 22 set. 2026.

CSIZMADIA, Andrew.; CURZON, Paul.; DORLING, Mark; HUMPHREYS, Simon; NG, Thomas; SELBY, Cynthia; WOOLLARD, John. **Computational thinking:** A guide for teachers. 2015. Computing At School (CAS). Disponível em:

https://eprints.soton.ac.uk/424545/1/150818_Computational_Thinking_1_.pdf. Acesso em: 20 de maio 2024.

HEFEZ, Abramo; VILLELA, Maria Lúcia Torres. **Códigos corretores de erros.** Rio de Janeiro: IMPA, 2008)

LIDL, Rudolf; NIEDERREITER, Harald. **Introduction to finite fields and their applications.** Cambridge University Press, 1994.

MALONEY, John; RESNICK, Mitchel; RUSK, Natalie; SILVERMAN, Brian; EASTMOND, Evelyn. The Scratch Programming Language and Environment. ACM Transactions on Computing **Education,** v. 10, n. 4, 2010.

ONUCHIC, Lourdes de la Rosa; ALLEVATO, Norma Suely Gomes. Pesquisa em resolução de problemas: caminhos, avanços e novas perspectivas. Bolema, Rio Claro, v. 25, n. 41, p. 73–98, 2011.

PAPERT, Seymour. **A Máquina das Crianças:** repensando a escola na era da informática. Trad. Sandra Costa. Porto Alegre: Artes Médicas, 1994.

RAABE, André; ZORZO, Avelino F.; BLIKSTEIN, Paulo. **Computação na educação básica:** fundamentos e experiências. Penso Editora, 2020.

RESNICK, Mitchel; Maloney, Jonh; MONROY-HERNÁNDEZ, Andrés; RUSK, Natalie; EASTMOND, Evelyn; BRENNAN, Karen, MILLNER, Amon; ROSENBAUM, Eric; SILVER, Jay, SILVERMAN, Brian; KAFAI, Yasmin. Scratch: Programming for All. **Communications of the ACM,** v. 52, n. 11, 2009.

SÁPIRAS, Fernanda Schuck; DALLA VECCHIA, Rodrigo; MALTEMPI, Marcus Vinicius. Utilização do Scratch em sala de aula. Educação Matemática Pesquisa, São Paulo, v. 17, n. 5, p. 973–988, 2015.

WING, Jeannette M. Computational thinking. Communications of the ACM, New York, v. 49, n. 3, p. 33–35, 2006. DOI: https://doi.org/10.1145/1118178.1118215.

ZORZO, Avelino Francisco; RAABE, Andree Luis Alice; BRACKMANN, Christian. Computação: o vetor de transformação da sociedade. **arXiv e-prints,** p. arXiv: 2106.11419, 2021.

---

[1] Licenciada em Matemática pela Universidade Católica de Goiás (2005–2008). Possui especialização em Matemática pelo Instituto Federal de Goiás (IFG) (2017–2019), mestrado profissional em Matemática em Rede Nacional (PROFMAT) pela Universidade Federal de Goiás (UFG) (2019–2021) e especialização em Robótica Educacional e suas Tecnologias no Ensino de Matemática pela Universidade Federal de Catalão (UFCAT) (2023–2024). Atua nas áreas de matemática, ensino de matemática, educação matemática, robótica educacional, tecnologias educacionais, metodologias de ensino, formação docente, práticas pedagógicas e inovação no ensino de matemática. Currículo Lattes: http://lattes.cnpq.br/6330687095233102. ORCID: https://orcid.org/0009-0009-3672-5410. E-mail: acesse o artigo original para visualizar o e-mail.

[2] Professor efetivo da rede estadual de Goiás (SEDUC-GO) e da rede municipal de Senador Canedo (SEMEC). É mestre em Matemática pela UFG, especialista em Robótica Educacional (UFCAT) e licenciado em Física (PUC Goiás) e Matemática (FIAR), com foco em

metodologias ativas e tecnologias educacionais. E-mail: acesse o artigo original para visualizar o e-mail.

[3] Professora de Rede Estadual de Goiás. Possui especialização em Matemática pelo Instituto Federal de Goiás (IFG). Licenciada em Matemática pela Universidade Católica de Goiás (UCG). E-mail: acesse o artigo original para visualizar o e-mail

[4] Estudante do 9º ano do Ensino Fundamental, Escola São Francisco de Assis, Senador Canedo, GO, Brasil. E-mail: acesse o artigo original para visualizar o e-mail

[5] O projeto completo desenvolvido no Scratch está disponível para acesso, teste e remixagem em: https://scratch.mit.edu/projects/1391719095/