SERVIÇO PÚBLICO FEDERAL
UNIVERSIDADE FEDERAL DO PARÁ
CAMPUS UNIVERSITÁRIO DE BRAGANÇA
FACULDADE DE MATEMÁTICA

LUIS HENRIQUE DA SILVA REIS

UMA IMPLEMENTAÇÃO EM PSEUDOLINGUAGEM PARA A CIFRA DE HILL

BRAGANÇA-PA
2023

LUIS HENRIQUE DA SILVA REIS

UMA IMPLEMENTAÇÃO EM PSEUDOLINGUAGEM PARA A CIFRA DE HILL

Trabalho de Conclusão de Curso apresentado à Universidade Federal do Pará, como parte dos requisitos necessários para obtenção do Título de Licenciado Pleno em Matemática.

Prof. MSc. Nelson Ned Nascimento Lacerda

BRAGANÇA-PA
2023

Dados Internacionais de Catalogação na Publicação (CIP) de acordo com ISBD
Sistema de Bibliotecas da Universidade Federal do Pará
Gerada automaticamente pelo módulo Ficat, mediante os dados fornecidos pelo(a) autor(a)

S586i Silva Reis, Luis Henrique da.
Uma implementação em pseudolinguagem para a cifra de Hill / Luis Henrique da Silva Reis. — 2023.
73 f. : il. color.

Orientador(a): Prof. Me. Nelson Ned Nascimento Lacerda
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal do Pará, Campus Universitário de Bragança, Faculdade de Matemática, Bragança, 2023.

1. Matrizes e determinantes. 2. Lógica de programação. 3. Algoritmos. 4. Cifra de Hill. 5. VisuAlg. I. Título.

CDD 510.28553

LUIS HENRIQUE DA SILVA REIS

UMA IMPLEMENTAÇÃO EM PSEUDOLINGUAGEM PARA A CIFRA DE HILL

Trabalho de Conclusão de Curso apresentado à Universidade Federal do Pará, como parte dos requisitos necessários para obtenção do Título de Licenciado Pleno em Matemática.

Bragança, 8 de fevereiro de 2023

BANCA EXAMINADORA

Nelson Ned Nascimento Lacerda

Prof. MSc. Nelson Ned Nascimento Lacerda
Orientador – UFPA

Marly dos Anjos Nunes

Prof. Dra. Marly dos Anjos Nunes
Examinadora Interna – UFPA

Edson Jorge de Matos

Prof. Dr. Edson Jorge de Matos
Examinador Interno – UFPA

AGRADECIMENTOS

Aos familiares; mãe Sra. Elza Correa e irmão Danilo Correa, pelo enorme apoio ao longo desta jornada acadêmica, pois "Para as coisas que são necessárias para a vida, o homem é auxiliado pela família da qual é parte". São Tomás de Aquino.

A UFPA, campus Bragança e FAMAT pelo período de aprendizagem, ministrado pelos exímios professores que a compõe, além disso aos amigos da turma matemática 2017 em especial ao amigo Wallace Almeida, por "Idem velle, idem nolle". São Tomás de Aquino.

Ao orientador Prof. MSe. Nelson Lacerda por ter cedido seu tempo e dedicação na elaboração deste trabalho.

A banca examinadora, Prof. Dra. Marly Nunes e Prof. Dr. Edson Matos, por aceitarem o convite.

“A matemática é a linguagem das conexões que não vemos entre objetos, mas para usar e aplicar essa linguagem devemos ser capazes de apreciar, sentir, de tomar posse do que é oculto, do inconsciente”.

(Ada Lovelace).

RESUMO

Este trabalho apresenta uma aplicação em pseudolinguagem para a cifra de Hill no Programa VisuAlg. Para tal, norteado através de revisões bibliográficas, de natureza aplicada, concatenamos conceitos matemáticos da álgebra linear tais como; matrizes e determinantes, bem como teoria dos números; algoritmos de Euclides e aritmética modular, junto a criptografia, algoritmos e lógica de programação. Assim foi desenvolvido um algoritmo com 800 linhas de códigos para cifrar e decifrar mensagens, embora o número de caracteres seja limitado, este não afeta os resultados. Esta pesquisa surgiu em razão do autor possuir conhecimentos na área lógico programacional, então o interesse em aplicar conhecimentos matemáticos em lógica de programação, com finalidade em contribuir para a matemática alinhada a área de programação.

Palavras-chave: Matrizes e determinantes; Lógica de programação; Algoritmos; Cifra de Hill; VisuAlg.

ABSTRACT

This work presents a pseudolanguage application for the Hill cipher in the VisuAlg Program. To this end, guided by bibliographic reviews, of an applied nature, we concatenate mathematical concepts of linear algebra such as; matrices and determinants, as well as number theory; Euclidean algorithms and modular arithmetic, along with cryptography, algorithms and programming logic. Thus, an algorithm with 800 lines of codes was developed to encrypt and decrypt messages, although the number of characters is limited, this does not affect the results. This research arose because the author has knowledge in the programming logic area, so the interest in applying mathematical knowledge in programming logic, with the purpose of contributing to mathematics aligned with the programming area.

keywords: Matrices and Determinants; Programming Logic; Algorithms; Hill's Cipher; VisuAlg.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A matemática e ciência da computação, compartilham o mesmo conceito sobre algoritmos; uma sequência de instruções finitas e bem definidas cujo o objetivo é resolver um determinado problema. Na área matemática, um dos algoritmos mais antigo e famoso fora desenvolvido por Euclides a (300 a.C.) voltado para o cálculo do MDC, no qual também é objeto de estudo deste trabalho. A origem da palavra algoritmo remete a Al Khowarizmi [1]

Já na computação, Ada Lovelace fora a precursora do que hoje é conhecido como programa de computador (MARTINS, 2016). Ada sempre esteve próxima aos projetos de Charles Babbage. Após o sucesso da máquina diferencial em 1822, no qual tinha como objetivo realizar cálculos de polinômios de forma mecânica, Charles cria o projeto da maquina analítica, no qual assegurava desenvolver uma variedade de operações matemáticas complexas.

Na Itália de 1842, Charles participa de um seminário, no qual expõe seus resultados sobre a nova máquina analítica. O seminário fora transcrito em francês no formato de artigo. Após a publicação, por intermédio de Charles, Ada é convidada traduzir o artigo para o inglês e adicionar notas autorais, assim concluindo em 1843. Devido as notas, o artigo vai além de uma simples tradução, logo fora publicado em periódicos matemáticos, com destaque para as notas assinadas sob as iniciais AAL(Augusta Ada Lovelace). Ada as classificou alfabeticamente de A a G, sendo esta última conhecida como o primeiro programa (algoritmo) de computador, sendo um algoritmo no qual computava os números de Bernoulli. Além do algoritmo, Ada presume que a máquina não só poderia computar números mas também criar imagens.

A criptografia é mais um ramo da ciência que se faz presente neste trabalho através da matemática e algoritmos. Sua finalidade é possibilitar a transferência segura de dados pessoais. As criptografias foram classificadas em fortes e fracas, sendo a cifra de Hill estabelecida como fraca, pois devido aos atuais recursos computacionais será fácil quebra-la, em contra partida o método criptográfico RSA encontra-se no status de impossível de ser quebrado, considerando os recursos tecnológicos atuais.

Esta implementação objetiva presentar um algoritmo feito no programa Visualg para o método criptográfico cifra de Hill. Para tal, com base em referenciais bibliográficos pretende-se compreender conceitos sobre matrizes, determinantes e teoria dos números, bem como a utilização da pseudolinguagem visualg e conceitos sobre o método criptográfico cifra de Hill, como também desenvolver o raciocínio lógico programacional por meio da matemática, visto que para aprender uma linguagem de programação, se fez necessário compreender os conceitos de lógica de programação. .

O presente trabalho está organizado em capítulos de 1 a 8, no qual o capítulo 1 é a presente introdução. Já o capítulo 2 fora destinado ao estudo da álgebra linear, bem como seus conceitos de matrizes e determinantes. O capítulo 3 se faz presente para o estudo de teoria dos números, no mais, conceitos sobre Euclides, congruência e recíproco modular. Para o capítulo 4 abordaremos alguns conceitos sobre algoritmos e lógica de programação. Já no capítulo 5 será explanado os conceitos do programa Visualg, além disso faremos alguns algoritmos para se obter o produto de uma matriz, bem como seu determinante e o mais importante, um algoritmo para se obter o recíproco de um inteiro positivo. No capítulo 6 faremos um breve resumo sobre criptografia e alguns métodos criptográficos. Para o capítulo 7 especificaremos o processo de criptografar e descriptografar a cifra de Hill, seguido da implementação no programa Visualg. O capítulo 8 trata das considerações finais, no qual abordaremos discussões e resultados a respeito deste trabalho.

[1] Matemático, astrônomo, astrólogo, geógrafo e escritor persa século IX.

2 MATRIZES E DETERMINANTES

Neste capítulo iremos explorar algumas definições sobre matrizes e suas operações, bem como determinantes (STEINBRUCH, 1987) , (ANTON; RORRES, 2012)

Na história da matemática os conceitos de álgebra linear iniciou-se com os sistemas de equações lineares, determinantes e então matrizes. A origem dos sistemas lineares datam de 300 a.C. escritas em tabletas de argila babilônicas. Já na China em 200 a.C. no livro *Nove Capítulos sobre a Arte Matemática* consta que fora desenvolvido um método de representar os coeficientes dos sistemas lineares através de bastões da bambu em um tabuleiro, na cor vermelho para positivos e preto para negativo.

O estudo dos determinantes surgiu no Japão em 1683 por Seki Kowa, em seus estudos Seki compreende a dinâmica dos determinantes de 2° a 5° ordem. Em 1693, Gottfried Leibniz em carta enviada ao Marquês de L'Hôpital apresenta seus estudos para a solução geral de um sistema linear com três equações e duas incógnitas, onde chegou-se a uma equação que determinaria a solução do sistema.

2.1 Matriz Elementar

Definição 1. *Uma matriz m×n (m por n) é uma arranjo retangular de números, dispostos em m linhas e n colunas:*

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \\ a_{m1} & a_{m2} & a_{m3} & \dots & a_{mn} \end{bmatrix}$$

Cada elemento da matriz A é indicado por a_{ij}. No qual os índices i e j são indicados respectivamente por linha e coluna. Também a matriz A pode ser representada por $A = [a_{ij}]$, com i variando de 1 a m, $i = 1, 2, 3, 4, ..., m$ e j variando de 1 a n, $j = 1, 2, 3, 4..., n$. Portanto, sendo a matriz A de ordem m por n, esta é representada por $A_{(m,n)}$. Ou seja, em uma matriz com 4 linhas e 3 colunas tem-se $A_{(4,3)}$ diz-se; matriz de ordem quatro por três.

Exemplo 1. Sejam as matrizes:

$$A = \begin{bmatrix} 3 & 5 & -1 & \sqrt{25} & \frac{3}{7} \\ 4 & 2 & 8 & 0 & 2 \end{bmatrix}, B = \begin{bmatrix} 5 & -1 \\ 3 & 9 \\ 8 & 0 \\ \frac{7}{8} & 2 \end{bmatrix}, C = \begin{bmatrix} 9 \\ 1 \\ 6 \end{bmatrix}$$

Logo, temos uma matriz A de ordem $i = 2$ e $j = 5$, ou seja, $A_{(2\times5)}$. E de forma análoga, a matriz $B_{(4\times2)}$ e $C_{(3\times1)}$.

2.2 Matrizes Especiais

2.2.1 Matriz Quadrada

Quando o número de linhas é equivalente ao de colunas, tem-se um matriz quadrada de ordem (n×n).

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \cdots & \dots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix}$$

2.2.2 Matriz Identidade

Definição 2. *Seja uma matriz quadrada $A = [a_{ij}]$ onde $a_{ij} = 1$ se $i = j$, $a_{ij} = 0$ se $i \neq j$ é denominada matriz identidade, denotada por I_n.*

Exemplo 2. Veja.

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

2.3 Operações com Matrizes

2.3.1 Adição

Definição 3. *Se $A = [a_{ij}]$ e $B = [b_{ij}]$ são de ordem (m×n), a soma de A e B é uma matriz C_{ij} de ordem $(m \times n)$ tal que:*

$$c_{ij} = a_{ij} + b_{ij}$$

Exemplo 3. Seja $A = \begin{bmatrix} 7 & -5 & 2 \\ 3 & -1 & 8 \end{bmatrix}$ e $B = \begin{bmatrix} 0 & 9 & -4 \\ 5 & 3 & 1 \end{bmatrix}$. Encontrar C, a partir de A + B.

$$A + B = \begin{bmatrix} 7+0 & -5+9 & 2+(-4) \\ 3+5 & -1+3 & 8+1 \end{bmatrix} = \begin{bmatrix} 7 & 4 & -2 \\ 8 & 2 & 9 \end{bmatrix} = C$$

2.3.2 Subtração

Definição 4. *Se $A = [a_{ij}]$ e $B = [b_{ij}]$ são de ordem (m×n), a subtração de A e B é uma matriz C_{ij} de ordem $(m \times n)$ tal que:*

$$c_{ij} = a_{ij} - b_{ij}$$

Exemplo 4. Seja $A = \begin{bmatrix} 2 & -9 & 1 \\ -3 & 8 & 4 \end{bmatrix}$ e $B = \begin{bmatrix} 5 & -7 & 8 \\ -6 & 2 & 3 \end{bmatrix}$. Encontrar C, apartir de A − B.

$$A + B = \begin{bmatrix} 2-5 & -9-(-7) & 1-8 \\ -3-(-6) & 8-2 & 4-3 \end{bmatrix} = \begin{bmatrix} -3 & -2 & -7 \\ 3 & 6 & 1 \end{bmatrix} = C$$

2.3.3 Multiplicação por um Escalar

Definição 5. *Se λ é um escalar, o produto de uma matriz $A = [a_{ij}]$ por um escalar resultará em $B = [b_{ij}]$ tal que:*

$$b_{ij} = \lambda a_{ij}$$

Exemplo 5. Seja $\lambda = -4$ e $A = \begin{bmatrix} 8 & -2 & 3 \\ -4 & 7 & 1 \\ 5 & -3 & 9 \end{bmatrix}$. Fazer $\lambda \cdot A$.

$$\lambda A = \begin{bmatrix} 8(-4) & -2(-4) & 3(-4) \\ -4(-4) & 7(-4) & 1(-4) \\ 5(-4) & -3(-4) & 9(-4) \end{bmatrix} = \begin{bmatrix} -32 & -8 & -12 \\ 16 & -28 & -4 \\ -20 & 12 & -36 \end{bmatrix}$$

2.3.4 Multiplicação

Definição 6. *Se $A = [a_{ij}]$ de ordem($m \times p$) e $B = [b_{ij}]$ de ordem ($p \times n$), então o produto de A e B, denotado por AB, é $C = [c_{ij}]$ de ordem ($m \times n$), tal que:*

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \ldots a_{ip}b_{pj} = \sum_{k=1}^{p} a_{ik}b_{kj}$$

Exemplo 6. Seja $A = \begin{bmatrix} 7 & -2 & 9 \\ 4 & 5 & -6 \\ 5 & -3 & 9 \end{bmatrix}$ e $B = \begin{bmatrix} 4 & -5 \\ -9 & 8 \\ 1 & 3 \end{bmatrix}$.

$$A \cdot B = \begin{bmatrix} 7 \cdot 4 + (-2) \cdot (-9) + 9 \cdot 1 & 7 \cdot (-5) + (-2) \cdot 8 + 9 \cdot 3 \\ 4 \cdot 4 + 5 \cdot (-9) + (-6) \cdot 1 & 4 \cdot (-5) + \cdot 8 + (-6) \cdot 3 \end{bmatrix} = \begin{bmatrix} 55 & -24 \\ -35 & 2 \end{bmatrix}$$

2.4 Determinante de uma Matriz

Definição 7. *Seja a matriz $A = [a_{ij}]$ uma matriz ($n \times n$). Definimos o determinante de A, denotado por $det(A)$;*

$$det(A) = \sum (\pm) a_{1_{j_1}} a_{2_{j_2}} \cdots a_{n_{j_n}}$$

Para uma matriz de 2º ordem, seu determinante será dado por;

$$det(A) = a_{11}a_{22} - a_{12}a_{21}$$

Exemplo 7. Seja a matriz. $A = \begin{bmatrix} 12 & 5 \\ 4 & 3 \end{bmatrix}$ calcule seu determinante.

Substituindo os elementos da matriz A em $det(A) = a_{11}a_{22} - a_{12}a_{21}$,
temos,

$$det(A) = 12 \cdot 3 - 5 \cdot 4 = 16$$

2.5 Matriz Inversa

Definição 8. *Uma matriz A (n×n) é invertível se existir uma matriz B (n×n) tal que:*

$$AB = BA = I_n.$$

A matriz B é chamada a inversa de A. Caso não existir uma matriz B, então dizemos que A não possui inversa.

Teorema 1. *Se uma matriz possui inversa, essa inversa é única*

Demostração. Sejam B e C inversas de A. Então,

$$BA = AC = I_n,$$

portanto

$$B = BI_n = B(AC) = (BA)C = I_nC = C.$$

■

Caso a inversa de A exista, denotaremos por A^{-1}. Assim,

$$AA^{-1} = A^{-1}A = I_n.$$

Exemplo 8. Seja $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$. Se existir, encontrar sua inversa.

Para encontrar A^{-1}, suponha que

$$A^{-1} = \begin{bmatrix} a & b \\ c & d \end{bmatrix},$$

assim temos

$$AA^{-1} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix} = I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

vem que

$$\begin{bmatrix} a+2c & b+2d \\ 3a+4c & 3b+4d \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Igualando os coeficientes correspondentes das matrizes acima, obtemos o seguinte conjunto de equações lineares.

$$a + 2c = 1 \qquad b + 2d = 0$$

$$e$$

$$3a + 4c = 0 \qquad 3b + 4d = 1.$$

E são soluções do sistema: $a = -2,\ c = \dfrac{3}{2},\ b = 1$ e $d = -\dfrac{1}{2}$. Além disso, como a seguinte igualdade é valida

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} -2 & 1 \\ \frac{3}{2} & -\frac{1}{2} \end{bmatrix},$$

isso quer dizer que também satisfaz

$$\begin{bmatrix} -2 & 1 \\ \frac{3}{2} & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Portanto, podemos concluir que A é invertível, assim

$$A^{-1} = \begin{bmatrix} -2 & 1 \\ \frac{1}{2} & -\frac{1}{2} \end{bmatrix}.$$

Teorema 2. *A matriz* $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ *é invertível se, e só se,* $(ad - bc) \neq 0$, *em* $A^{-1} = \frac{1}{ad-bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$.

Demostração. Para isto vamos provar que $AA^{-1} = A^{-1}A = I$.

Mostrar que $AA^{-1} = I$.

$$AA^{-1} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix},$$

obtemos

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} \frac{-d}{bc-ad} & \frac{b}{bc-ad} \\ \frac{c}{bc-ad} & \frac{-a}{bc-ad} \end{bmatrix},$$

onde

$$\begin{bmatrix} a\left(\frac{-d}{bc-ad}\right) + b\left(\frac{c}{bc-ad}\right) & a\left(\frac{b}{bc-ad}\right) + b\left(\frac{-a}{bc-ad}\right) \\ c\left(\frac{-d}{bc-ad}\right) + d\left(\frac{c}{bc-ad}\right) & c\left(\frac{b}{bc-ad}\right) + d\left(\frac{-a}{bc-ad}\right) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I.$$

Analogamente, para $A^{-1}A = I$

$$A^{-1}A = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix},$$

tem-se

$$\begin{bmatrix} \frac{-d}{bc-ad} & \frac{b}{bc-ad} \\ \frac{c}{bc-ad} & \frac{-a}{bc-ad} \end{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix},$$

logo

$$\begin{bmatrix} \left(\frac{-d}{bc-ad}\right)a + \left(\frac{b}{bc-ad}\right)c & \left(\frac{-d}{bc-ad}\right)b + \left(\frac{b}{bc-ad}\right)d \\ \left(\frac{c}{bc-ad}\right)a + \left(\frac{-a}{bc-ad}\right)c & \left(\frac{c}{bc-ad}\right)b + \left(\frac{-a}{bc-ad}\right)d \end{bmatrix},$$

portanto

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I.$$

■

Exemplo 9. Seja a matriz $A = \begin{bmatrix} 13 & 5 \\ 7 & 3 \end{bmatrix}$, calcular sua inversa.

A princípio, vamos avaliar se $det(A) = (ad - bc) \neq 0$.

$$det(A) = (13 \cdot 3 - 5 \cdot 7) = 4.$$

Então,

$$A^{-1} = \frac{1}{4}\begin{bmatrix} 3 & -5 \\ -7 & 13 \end{bmatrix} = \begin{bmatrix} \frac{3}{4} & -\frac{5}{4} \\ -\frac{7}{4} & \frac{13}{4} \end{bmatrix}.$$

Conclui-se que

$$AA^{-1} = \begin{bmatrix} 13 & 5 \\ 7 & 3 \end{bmatrix}\begin{bmatrix} \frac{3}{4} & -\frac{5}{4} \\ -\frac{7}{4} & \frac{13}{4} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

3 TEORIA DOS NÚMEROS

Neste capítulo iremos abordar algumas da relevantes contribuições de Euclides, assim como Friedrich Gauss para a aritmética modular segundo (ALENCAR, 1981) e (BEZERRA, 2018).

3.1 Algumas Contribuições de Euclides

3.1.1 Algoritmo da Divisão

Definição 9 (Principio da boa ordenação)**.** *Todo conjunto não vazio A de inteiros não negativos possui o elemento mínimo, ou seja, todo subconjunto não vazio A do conjunto* $\mathbb{Z}_+ = \{0, 1, 2, 3, \cdots\}$ *de inteiros não negativos* $(\varnothing \neq A \subset \mathbb{Z}_+)$ *possui elemento mínimo,tal que*

$$(\forall A \subset \mathbb{Z}_+,\ A \neq \varnothing) \Longrightarrow \exists\ min\ em\ A$$

Exemplo 10. O conjunto $A = \{2, 4, 6, 8, ...\}$ dos inteiros positivos pares é subconjunto não vazio de

$$\mathbb{Z}_+(\varnothing \neq A \subset \mathbb{Z}_+).$$

portanto, pelo princípio da boa ordenação, A possui elemento mínimo, onde $(minA = 2)$.

Teorema 3 (Algoritmo da divisão)**.** *Se a e b são dois inteiros, com* $b > 0$*, então existem e são únicos os inteiros q e r que satisfazem as condições:*

$$a = bq + r \quad e \quad 0 \leq r < b.$$

Demostração. Seja S o conjunto de todos os inteiros não negativos da forma $a - bx$, com $x \in \mathbb{Z}$, ou seja,

$$S = \{a - bx \mid x \in \mathbb{Z}, a - bx \geq 0\}.$$

O conjunto S não é vazio $(S \neq \varnothing)$, pois, com b > 0, temos b ≥ 1, logo para x = - | a | temos,

$$a - bx = a + b \mid a \mid \geq a + \mid a \mid \geq 0.$$

Logo, pelo princípio da boa ordenação, existe o elemento mínimo r de S tal que;

$$r \geq 0\ e\ r = a - ba \quad ou \quad a = bq + r,\ com\ q \in \mathbb{Z},$$

além disso, temos $r < b$, pois, caso $r \geq b$, teríamos,

$$0 \leq r - b = a - bq - b = a - b(q + 1) < r,$$

ou seja, r não seria o elemento mínimo de S.

Em relação a unicidade de q e r. Suponhamos que existem dois outros inteiros q_1 e r_1 tais que,

$$a = bq + r \quad e \quad 0 \leq r_1 < b.$$

Segue-se que

$$bq_1 + r_1 = bq + r \Longrightarrow r_1 - r = (q - q_1)b \Longrightarrow b \mid (r_1 - r),$$

além disso

$$-b < -r \leq 0 \quad e \quad 0 \leq r_1 < b,$$

implica em

$$-b < r_1 - r < b, \mid r_1 - r \mid < b.$$

Veja que

$$b \mid (r_1 - r) \; e \mid r_1 - r \mid < b,$$

por consequência

$$r_1 - r = 0,$$

como $b \neq 0$, temos que

$$q - q_1 = 0.$$

Portanto

$$r_1 = r \;\; e \;\; q_1 = q.$$

■

3.1.2 Algoritmo de Euclides

Lema 1. *Se $a = bq + r$, então o mdc $(a, b) = mdc\ (b, r)$.*

Demostração. Se

$$mdc\ (a, b) = d,$$

então

$$d \mid a \;\; e \;\; d \mid b,$$

isso implica que

$$d \mid (a - bq) \;\; ou \;\; d \mid r,$$

assim, d é divisor comum de b e r, logo

$$d \mid b \;\; e \;\; d \mid r.$$

Por outro lado, sendo c um divisor comum de b e r, temos

$$c \mid b \;\; e \;\; c \mid r,$$

então,

$$c \mid (bq + r) \;\; ou \;\; c \mid a.$$

isto é, c é um divisor comum de a e b, logo implica,

$$c \leq d.$$

Portanto

$$mdc(b, r) = d.$$

■

Definição 10 (Algoritmo de Euclides)**.** *Processo para o cálculo do mdc de dois inteiros positivos a e b.*

Figura 1: Dispositivo de Euclides

	q_1	q_2	q_3	q_n		q_{n+1}
a	b	r_1	r_2	$\cdots$	r_{n-1}	r_n
r_1	r_2	r_3	r_4		0	

Fonte: Própria do Autor

Sejam a e b dois inteiros não conjuntamente nulos ($a \neq 0$ *ou* $b \neq 0$), cujo o mdc se deseja determinar. De imediato:
(1) se $a \neq 0$, então $mdc(a, 0) =\mid a \mid$

(2) se $a \neq 0$, então $mdc(a, a) =\mid a \mid$

(3) se $b \mid a$, então $mdc(a, b) =\mid b \mid$

Além disso, sendo $mdc(a, b) = mdc(\mid a \mid, \mid b \mid)$, a determinação do $mdc(a, b)$ reduz-se ao caso em que a e b são inteiros positivos distintos, com $a > b$, tais que b não divide a, ou seja, $a > b > 0$ e $b \nmid a$. Neste caso, a aplicação repetida do algoritmo da divisão gera as igualdades:

$$a = bq_1 + r_1, \quad 0 < r_1 < b$$

$$b = r_1q_2 + r_2, \quad 0 < r_2 < r_1$$

$$r_1 = r_2q_3 + r_3, \quad 0 < r_3 < r_2$$

$$r_2 = r_3q_4 + r_4, \quad 0 < r_4 < r_3,$$

como os restos $r_1, r_2, r_3, r_4, ...$ são inteiros positivos tais que

$$b > r_1 > r_2 > r_3 > r_4 > ...$$

de modo que, existem apenas $b - 1$ inteiros positivos menores que b, o que implica em uma divisão cujo resto $r_{n+1} = 0$, assim

$$r_{n-2} = r_{n-1}q_n + r_n, \quad 0 < r_n < r_{n-1}$$

$$r_{n-1} = r_nr_{n+1} + r_{n+1}, \quad r_{n+1} = 0$$

o resto anterior a $r_n = 0$ é o $mdc(a, b)$, ou seja, o $mdc(a, b) = r_n$, visto que, pelo lema 1, temos:

$$mdc(a, b) = mdc(b, r_1) = mdc(r_1, r_2) = ... = mdc(r_{n-2}, r_{n-1}) = mdc(r_{n-1}, r_n) = r_n$$

3.1.3 Algoritmo de Euclides Estendido

Extensão do Algoritmo de Euclides, no qual determina o mdc entre dois inteiros positivos, bem como para obter o recíproco modular (COUTINHO, 2007), (HEFEZ, 2006).

Definição 11 (Algoritmo de Euclides Estendido)**.** *Sejam a e b inteiros positivos, se $mdc(a,b) = r_n$, então $\exists$ x e y tais que.*

$$x \cdot a + y \cdot b = r_n$$

Observação 1. Este algoritmo é uma combinação linear de a e b.

Exemplo 11. Seja $a = 281$ e $b = 26$, encontrar o $mdc(a,b)$ e sua expressão como combinação linear de a e b.

Pelo algoritmo euclidiano,

$$281 = 26 \cdot 10 + 21 \tag{1}$$

$$26 = 21 \cdot 1 + 5 \tag{2}$$

$$21 = 5 \cdot 4 + 1 \tag{3}$$

$$5 = 1 \cdot 5 + 0 \tag{4}$$

Tomando a equação (3), nota-se que $r_n = 1$, ou seja $mdc(281, 26) = 1$.

Utilizando o algoritmo euclidiano estendido, além disso operando somente com equações cujo $r_n \neq 0$ e isolando os termos r_n, temos

$$1 = 21 - 5 \cdot 4 \tag{5}$$

$$5 = 26 - 21 \cdot 1 \tag{6}$$

$$21 = 281 - 26 \cdot 10 \tag{7}$$

Partindo de

$$1 = 21 - 5 \cdot 4$$

Substituindo (6) em (5) temos

$$1 = 21 - (26 - 21 \cdot 1)4 \tag{8}$$

Aplicando a distributiva

$$1 = 21 - 26 \cdot 4 + 21 \cdot 1 \cdot 4 \tag{9}$$

Aplicando a associativa

$$1 = 21 \cdot 5 - 26 \cdot 4 \tag{10}$$

Substituindo (7) em (10), tem-se

$$1 = (281 - 26 \cdot 10)5 - 26 \cdot 4 \tag{11}$$

Aplicando a distributiva

$$1 = 281 \cdot 5 - 26 \cdot 10 \cdot 5 - 26 \cdot 4 \tag{12}$$

Aplicando a associativa

$$1 = 281 \cdot 5 - 26 \cdot 54 \tag{13}$$

Portanto $x = 5$ e $y = -54$

3.2 Aritmética Modular

Além da teoria de Euclides visto acima, neste trabalho iremos necessitar de definições da aritmética modular.

3.2.1 Congruência

Definição 12. *Sejam a, b e $m \in \mathbb{Z}$, com m positivo fixo, Diz-se que a é congruente a b módulo m se e somente se m divide a diferença $a - b$, tal que $a - b = km$, ou seja,*

$$a \equiv b(mod\ m)$$

Exemplo 12. Tome os números 7 e 11 $\in \mathbb{Z}$, além disso 2 e 3 $\in \mathbb{Z}$. Se 7 e 11 forem divididos por 2, em ambos obtém-se resto igual a 1, no entanto ao serem divididos por 3 obtém-se respectivamente restos 1 e 2, assim

$$7 \equiv 11(mod\ 2),$$

no entanto

$$7 \not\equiv 11(mod\ 3).$$

Teorema 4. *Dois inteiros a e b são congruentes módulo m se e somente se a e b deixam o mesmo resto quando divididos por m.*

Demostração. ($\Longrightarrow$) Suponhamos que $a \equiv b(mod\ m)$.

Por definição

$$a - b = km, \qquad k \in \mathbb{Z}.$$

Seja r o resto da divisão de b por m, pelo algoritmo da divisão

$$b = mq + r, \quad onde \quad 0 \leq r < m$$

assim

$$a = km + b = km + mq + r = (k + q)m + r.$$

Em outros termos, r é o resto da divisão de a por m, onde os inteiros a e b divididos por m deixam o mesmo resto r.

($\Longleftarrow$) Analogamente, suponhamos a e b divididos por m deixam o mesmo resto r. Logo, podemos assumir que

$$a = mq_1 + r \ \text{ e } \ b = mq_2 + r, \ \text{ onde } \ 0 \leq r < m.$$

Portanto

$$a - b = (q_1 - q_2)m \Longrightarrow m \mid (a - b) \Longrightarrow a \equiv b(mod\ m).$$

■

3.2.2 Congruência Linear

Definição 13. *Seja a, b, x e $m \in \mathbb{Z}$, há congruência se*

$$a \cdot x \equiv b \ (mod\ m). \tag{14}$$

Dessa forma, todo inteiro x_0 tal que

$$ax_0 \equiv b \ (mod\ m),$$

é solução da congruência linear (14) se e somente se

$$m|(ax_0 - b),$$

isto é, caso exista um inteiro y_0 tal que

$$ax_0 - b = my_0.$$

Se x_0 é uma solução de (14), então todos os inteiros $x_0 + km$, sendo k um inteiro qualquer, ou seja,

$$..., x_0 - 2m,\ x_0 - m,\ x_0 + m,\ x_0 + 2m, ...$$

também são soluções de (14), visto que

$$a(x_0 + km) \equiv ax_0 \equiv b(mod\ m),$$

ou seja,

$$x_0 \equiv x_1 \ (mod\ m),$$

Exemplo 13. Encontrar a solução para a congruência linear $21x \equiv 43 \ (mod\ 26)$,

Pelo algoritmo estendido temos

$$26 = 21 \cdot 1 + 5$$

$$21 = 5 \cdot 4 + 1$$

$$5 = 1 \cdot 5 + 0$$

Isolando os restos

$$1 = 21 - 5 \cdot 4$$

$$5 = 26 - 21 \cdot 1$$

Substituindo os termos

$$1 = 21 - (26 - 21 \cdot 1)4$$

$$1 = 21 - 26 \cdot 4 + 21 \cdot 1 \cdot 4$$

$$1 = 21 - 26 \cdot 4$$

Multiplicando b, ou seja 43 em ambos os membros

$$43 = 21 \cdot 5 \cdot 43 - 26 \cdot 4 \cdot 43$$

$$43 = 21 \cdot 215 - 26 \cdot 172$$

Logo $x = 215$ e $y = -172$

Portanto

$$21 \cdot 215 \equiv 43 \ (mod\ 26)$$

Vimos que $x_0 = 215$ é uma solução, e por consequência todo inteiro $215 + 26k$ também. Variando $k = -10, -9, 8, 1, 2$, temos

$$..., \ -45, \ -19, \ 7, \ 241, \ 267, \ ...$$

3.3 Recíproco ou Inverso Multiplicativo Modular

Definição 14. *Seja a e $m \in \mathbb{Z}^+$, $\exists\ a^{-1}$, tal que,*

$$a \cdot a^{-1} \equiv 1 \ (mod\ m)$$

Observação 2. Se $a^{-1} < 0$, então aplicar $a^{-1} \equiv x \ (mod\ m)$, pois $a, m \in \mathbb{Z}^+$.

Observação 3. Se (mod m) for convencionalmente pequeno, basta variar x em

$$a \cdot x \equiv 1 \ (mod\ m)$$

Exemplo 14. Seja $a = 89$ e $m = 26 \in \mathbb{Z}$ Encontrar a^{-1}(inverso) de $a \ (mod\ m)$.
Utilizando o algoritmo estendido de Euclides, iremos obter a^{-1}, tal que $mdc = m \cdot y + a \cdot a^{-1}$.

$$89 = 26 \cdot 3 + 11$$

$$26 = 11 \cdot 2 + 4$$

$$11 = 4 \cdot 2 + 3$$

$$4 = 3 \cdot 1 + 1$$

$$3 = 1 \cdot 3 + 0$$

Isolando os restos temos

$$1 = 4 - 3 \cdot 1$$

$$3 = 11 - 4 \cdot 2$$

$$4 = 26 - 11 \cdot 2$$

$$11 = 89 - 26 \cdot 3$$

Realizando as devidas substituições tem-se

$$1 = 4 - 3 \cdot 1$$

$$1 = 4 - (11 - 4 \cdot 2)1$$

$$1 = 4 - 11 \cdot 1 + 4 \cdot 2 \cdot 1$$

$$1 = 4 \cdot 3 - 11 \cdot 1$$

$$1 = (26 - 11 \cdot 2)3 - 11 \cdot 1$$

$$1 = 26 \cdot 3 - 11 \cdot 2 \cdot 3 - 11 \cdot 1$$

$$1 = 26 \cdot 3 - 11 \cdot 7$$

$$1 = 26 \cdot 3 - (89 - 26 \cdot 3)7$$

$$1 = 26 \cdot 3 - 89 \cdot 7 + 26 \cdot 3 \cdot 7$$

$$1 = 26 \cdot 24 - 89 \cdot 7$$

Portanto $y = 24$ e $a^{-1} = -7$
Como $a^{-1} < 0$, então, aplicar $a^{-1} \equiv x \ (mod\ m)$ logo,

$$-7 \equiv \ (mod\ 26) \Rightarrow 19$$

Sendo verdade que

$$-7 \equiv 19 \ (mod\ 26)$$

Portanto

$$a^{-1} = 19$$

Exemplo 15. Seja $a = 7$ e $m = 15$. Encontrar o recíproco de $(7\ mod\ 15)$.

Variando x de 1 a 15

$7 \cdot 1 \equiv 1(mod\ 15) \Rightarrow 7 \not\equiv 1\ mod\ 15$

$7 \cdot 5 \equiv 1(mod\ 15) \Rightarrow 35 \not\equiv 1\ mod\ 15$

$\vdots$

$7 \cdot 13 \equiv 1(mod\ 15) \Rightarrow 91 \equiv 1\ mod\ 15$

Portanto, 13 é o recíproco de 7 (mod 15).

4 ALGORITMOS E LÓGICA DE PROGRAMAÇÃO

Um algoritmo é uma sequência lógica e finita de passos bem definidos, tendo como intenção concluir um objetivo. O próprio método de Euclides para o MDC segue uma sequencia lógica bem definida e finita de passo cujo o objetivo é determinar o MDC, tanto que chama-se algoritmo de Euclides. Além do mais podem ser representados através de três formas: descrição narrativa, fluxograma ou pseudocódigo.

A receita de bolo na figura 2 é um algoritmo representado por descrição narrativa, pois os ingredientes estão escritos na forma corrente e seguem uma lógica de passos que são finitos e bem definidos cujo o objetivo é ter um bolo assado, que por sua vez pode ser parte de um objetivo maior, do qual poderá ser deliciar-se com o mesmo. Logo mediante novos objetivos, os algoritmos são passíveis de mudanças,

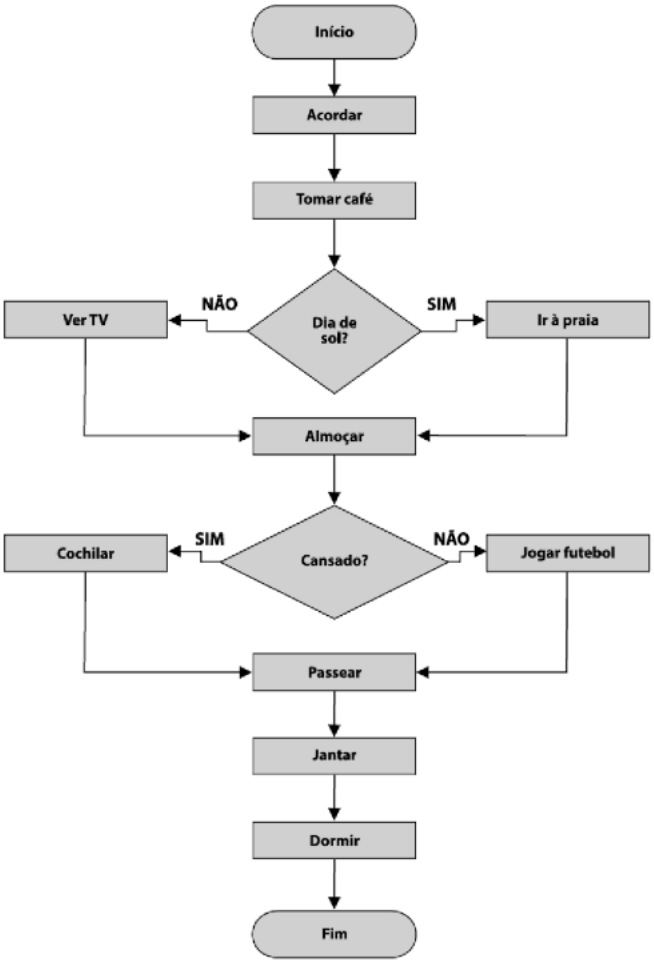
Figura 2: Receita de Bolo

Fonte: Pinterest 2023

O mesmo algoritmo da receita poderá ser representado através de um fluxograma, para tal é necessário que os símbolos sejam padronizados e bem definidos, onde os passos serão escritos dentro de uma figura geométrica. De modo análogo, observe a figura 3 e o modelo para representar a rotina de uma pessoa.

Esta representação tem como vantagem a visualização gráfica do processo. Este era o principal instrumento (junto a papel e caneta) para programadores esquematizar projetos, bem como no ensino de lógica de programação.

Figura 3: Fluxograma



Fonte: Forlogic 2016

Outra maneira para representar um algoritmo é através de pseudocódigo, onde de forma genérica e utilizando uma linguagem simples e nativa pode-se escrever um algoritmo. Observe a figura 4, as informações estão dispostas em variáveis.

Figura 4: Pseudocódigo Para Nota

```
algoritmo "Nota"
// Função :
// Autor :
// Data : 05/09/2022
// Seção de Declarações
var
   nota1, nota2, media: Real
inicio
      EscrevaL("------------------------")
      EscrevaL("   ESCOLA EUCLIDIANA   ")
      EscrevaL("------------------------")
      Escreva("Primeira Nota: ")
      Leia(nota1)
      Escreva("Segunda Nota: ")
      Leia(nota2)
      media <- (nota1 + nota2)/2
      EscrevaL("------------------------")
      EscrevaL(" MEDIA: ", media:3:1)
      Se (media >= 7) entao
         EscrevaL(" ALUNO APROVADO ")
      senao
         EscrevaL(" ALUNO REPROVADO ")
      FimSe
      EscrevaL("------------------------")
fimalgoritmo
```

Fonte: Própria do Autor

Todas as linguagens de programação compartilham entre si das mesmas regras e conceitos lógicos programacionais, tais como;

- Sequência lógica
- Condicionais simples e compostas
- Variáveis simples e compostas
- Variáveis compostas; vetoriais e matriciais
- Laços de repetições
- Procedimentos
- Rotinas

No qual serão abordadas no capítulo seguinte. Desse modo o encadeamento lógico das regras e conceitos é chamado de logica de programação. Que por sua vez é um algoritmo, ou seja, todo código de programação é um algoritmo, mas nem todo algoritmo é um código de programação.

A linguagem de programação objetiva a criação de algo que interaja com o computador; um programa, software, aplicativo, site web. A exemplo Word, Excel, Google. Já a pseudolinguagem não produz programas de computador, apenas cria, edita e compila a lógica dos processos, que por sua vez são chamados de pseudocódigos. Em resumo as única semelhanças entre linguagem de programação e pseudolinguagem são os conceitos programacionais, bem como o método de escrita estruturado. Já o termo pseudocódigo faz referência ao código criado por uma pseudolinguagem (FUNDAÇÃO BRADESCO, 2022).

5 PSEUDOLINGUAGEM VISUALG

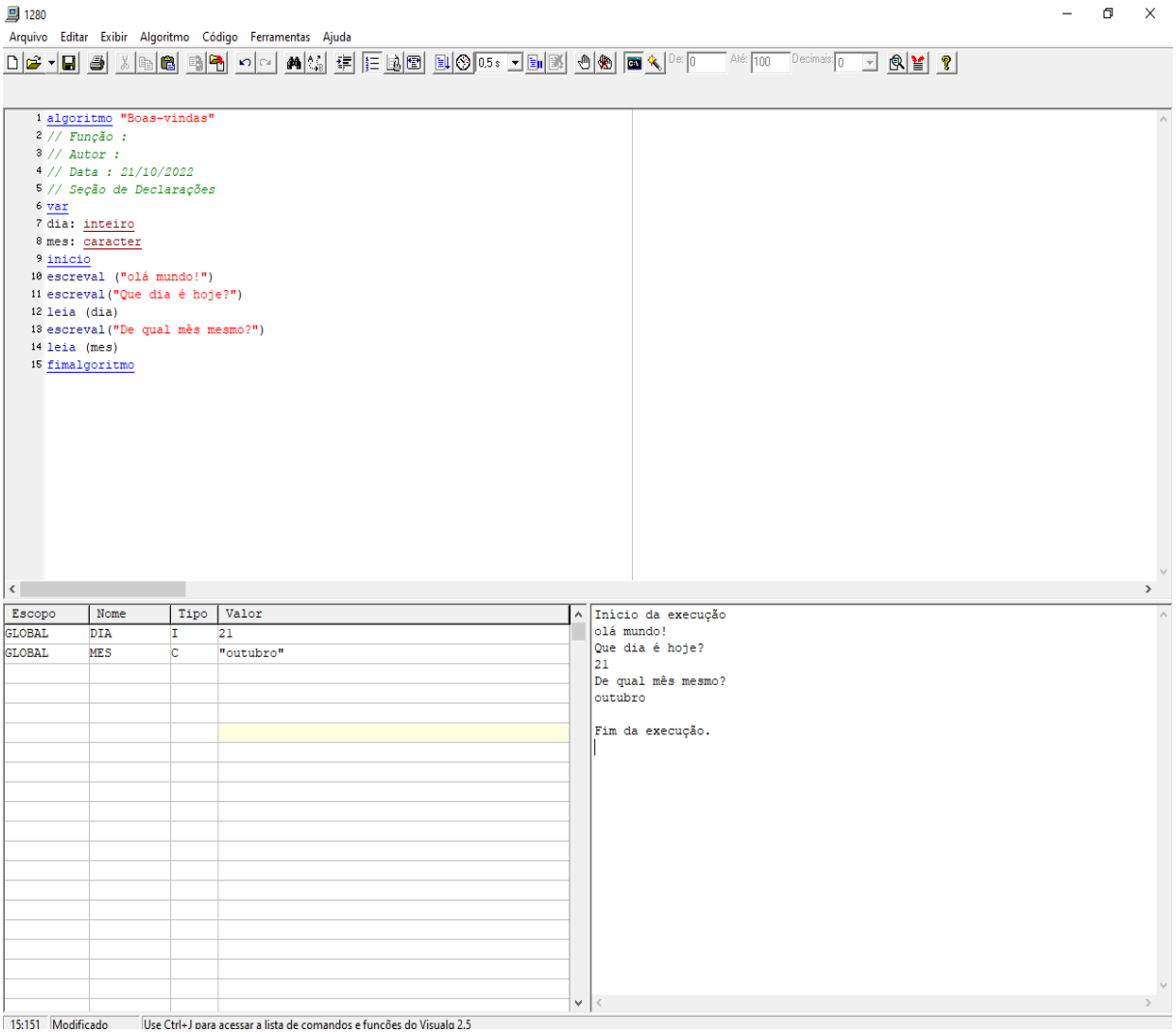
Neste capítulo abordaremos alguns conceitos relevantes do programa VisuAlg. Ressaltando que para este trabalho foi utilizado a versão 2.5, pois até o presente momento a versão atual 3.0 encontra-se com vários erros. Ademais norteado por (MANZANO; OLIVEIRA, 2019) e (GUANABARA, 2014).

Desenvolvido em 1996 pelo professor Cláudio Morgado e tendo novas atualizações com Carlos Nicolodi. Utilizado amplamente por faculdades do Brasil, pois é de distribuição grátis e domínio público, além de ser um executável de apenas 1,12 MB. Tem por finalidade auxiliar os alunos na aprendizagem de lógica de programação, substituindo a utilização de fluxogramas, papel e caneta. Aproximado esteticamente e conceitualmente a uma linguagem de programação, suas funções permitem criar, editar e interpretar algoritmos baseado em português estruturado[2], (MORGADO; NICOLODI, 1996).h

5.1 Entrada e Saída de Dados

Fora criado um algoritmo para interagir com o usuário, perguntando o dia e mês.

Figura 5: Estrutura do VisuAlg



Fonte: Própria do Autor

[2]Pseudolinguagem escrito no idioma português e baseado nos conceitos programacionais de estruturas sequenciais, condicionais e repetição

Observe a figura 5, o quadrante superior esquerdo e direito, este é o espaço destinado a inserção das linhas de códigos. Analisando o código escrito tem-se.

A linha 1 é dedicada a nomear o algoritmo, já as linhas 2 a 5 são especificações, comentários opcionais e não serão executadas, desde que sejam precedidas de //.

Da linha 6 até a 9 há o campo para inserir variáveis, discutiremos mais sobre as variáveis a seguir.

Entre as linha 9 inicio a 15 fimalgoritmo são desenvolvidos os comandos do algoritmo. Observe que fora digitado o comando escreval, assim qualquer variável ou linha de caracter que seja digitado neste comando será impresso na tela, dês de que esteja: (“ ”) para linha de caracter e apenas: () para variáveis, como é o caso das variáveis dia e mes.

Já no quadrante inferior esquerdo, dispõe o campo no qual específica todas as variáveis inseridas.

Por fim, no quadrante inferior direito tem-se a saída dos dados inseridos, seguido da mensagem padrão “fim da execução”, “feche esta janela para retornar ao visualg”.

5.2 Variáveis 1

O conceito de variáveis do visualg é o mesmo da matemática, ou seja, um local que armazena dados arbitrários. Há dois modelos de variáveis: primitivas e compostas.

A priori iremos abordar apenas variáveis primitivas, pois será necessário compreender outros conceitos de algoritmos para então adentrar ao conceito de variável composta.

5.2.1 Variáveis Primitivas

Conceitualmente, refere-se ao tipo de informação básica e rotineira.

Tabela 1: Variáveis Primitivas

Tipo	Descrição
Inteiro	Armazena um número inteiro
Real	Armazena um número real
Caracter	Armazena um caracter ou cadeia de caracter em ””

Fonte: Própria do Autor

No visualg, as variáveis são inseridas abaixo do comando var, da seguinte forma.

Figura 6: Inserção de Variáveis

```
1 algoritmo ""
2 var
3 nome: caracter
4 idade: inteiro
5 salário: real
6 inicio
7 fimalgoritmo
```

Fonte: Própria do Autor

5.3 Operadores Aritméticos

Tabela 2: Operadores Aritméticos

Operação Aritmética	Operador em Português estruturado
Multiplicação	*
Divisão	/
Divisão Inteira	\
Exponenciação	^
Adição	+
subtração	-
Resto da divisão(módulo)	%

Fonte: Própria do Autor

5.4 Operadores Relacionais

Tabela 3: Operadores Relacionais

Operadores relacionais	Português estruturado
Maior	>
Menor	<
Maior ou igual	>=
Menor ou igual	<=
Igual	=
Diferente	><

Fonte: Própria do Autor

5.5 Operadores Lógicos

Tabela 4: Operadores Lógicos

Operadores lógicos	Português estruturado	Resultado
Multiplicação lógica	E	Resulta V se ambas as partes forem V.
Adição lógica	+	Resulta V se uma das partes é V.
Negação	-	Se for V torna-se F, se for F torna-se V.

Fonte: Própria do Autor

5.6 Estrutura Condicional Simples

O comando se baseia na condição verdadeiro ou falso. Se o comando for verdadeiro este é executado, caso contrário a estrutura é finalizada sem executar os comandos, sua sintaxe.

Figura 7: Estrutura Simples

```
Se (condição verdadeira) entao
    (executar comandos)
fimse
```

Fonte: Própria do Autor

5.7 Estrutura Condicional Composta

Caso tenha mais de uma condição e a primeira ou outras sejam falsas e você queira continuar com a execução dos demais comandos, basta adicionar o expressão senão, como podemos observar sua sintaxe.

Figura 8: Estrutura Composta

```
Se (condição verdadeira) entao
    (Ações a serem realizadas se a condição for verdadeira)
Senao
    (Ações a serem realizadas se a condição for falsa)
Fimse
```

Fonte: Própria do Autor

5.8 Estrutura Escolha Caso

É uma solução elegante, utilizada especificamente para que seja possível escolher uma opção dentre várias existentes.

Figura 9: Estrutura De Escolha

```
Escolha (expressão de seleção)
Caso 1, 2, ..., n
        lista de comandos 1 a ser executada
Caso 3, 4, ..., n
        lista de comandos 2 a ser executada
Outrocaso
Fimescolha
```

Fonte: Própria do Autor

5.9 Estrutura de Repetição

Um algoritmo segue uma sequência linear de instrução e execução, onde até então não é possível retornar para a linha anterior, no qual torna-se um problema para certos casos.

Do modo convencional, para inserir a nota de 25 alunos seria necessário escrever 25 linhas de códigos, assim tornando-se inviável. Por outro lado, há uma solução elegante para este problema, basta repetir um trecho do código, então um loop é iniciado, no entanto atentar para definir corretamente o fim do loop, nos comandos Repita e Enquanto.

Neste trabalho utilizaremos especificamente a estrutura "para".

Figura 10: Estruturas De Repetições

```
Repita
        (lista de comandos)
Ate (expressão lógica ou relacional)
```

```
Enquanto (expressão lógica ou relacional) faca
        (lista de comandos)
Fimenquanto
```

```
Para (variável de controle) de (valor inicial) ate (valor final) [passo (incremento)] faca
        (lista de comandos)
Fimpara
```

Fonte: Própria do Autor

5.10 Variáveis 2

5.10.1 Variáveis Compostas Homogêneas

As variáveis primitivas permitem apenas a inserção de um valor para cada variável, desse modo é inviável atribuir um número expressivo de valores para cada variável, pois perderíamos muito tempo, então para solucionar este problema fora criada este conceito, onde pode-se atribuir n valores para n variáveis, desde que sejam do mesmo tipo.

São variáveis que exigem a utilização de qualquer comando de repetição.

5.10.2 Variáveis Indexadas Unidimensionais: Vetores

São Variáveis com uma única dimensão, ou seja, é um conjunto organizado de variáveis referenciadas por um único índice. Veja abaixo sua sintaxe.

Figura 11: Variável Vetorial

```
Identificador: Vetor [Tamanho] de tipo
Tamanho = [VI..VF] Vi= Valor inicial do índice VF = valor Final do índice
```

Fonte: Própria do Autor

Exemplo 16. Algoritmo para contar de 1 a 10 utilizando o comando “para”.

Figura 12: Algoritmo Contador

```
1 algoritmo "Contar de 1 a 10"
2 var
3 i: vetor [1..10] de inteiro
4 contador: inteiro
5 inicio
6 Para contador <- 1 ate 10 faca
7 escreva(contador)
8 fimpara
9 fimalgoritmo
```

Fonte: Própria do Autor

5.10.3 Variáveis Indexadas Bidimensionais: Matrizes

Variáveis indexadas com duas dimensões, também conhecida como matrizes, são referenciadas por dois índices. A sintaxe para declaração é:

Figura 13: Variável Bidimensional

```
Identificador: vetor [tamanho1],[tamanho2] de tipo
Tamanho = [VI..VF] Vi= Valor inicial do índice VF = valor Final do índice
```

Fonte: Própria do Autor

Exemplo 17. Algoritmo para preencher uma matriz de ordem (3×2).

Figura 14: Algoritmo de Preenchimento Matricial

```
1 algoritmo "Preencher matriz"
2 var
3 matriz: vetor [1..3,1..2] de inteiro
4 i, j: inteiro
5 inicio
6 Para i <- 1 ate 3 faca
7 Para j <- 1 ate 2 faca
8 escreva ( "Digite o valor da posição [", i, ",", j,"]:")
9 leia (matriz[i,j])
10 FimPara
11 FimPara
12 Para i <- 1 ate 3 faca
13 Para j <- 1 ate 2 faca
14 escreva ( "Digite o valor da posição [", i, ",", j,"]:")
15 leia (matriz[i,j]: 5)
16 FimPara
17 escreval ()
18 FimPara
19 fimalgoritmo
```

Fonte: Própria do Autor

Observe que é indispensável a utilização de um comando de repetição, neste caso o "para".

5.11 Subalgoritmos

É um algoritmo secundário, no qual resolve um pequeno problema, e está diretamente subordinado a um outro algoritmo principal, no entanto é executado somente se acionado por este principal. É conveniente utilizá-los quando uma determinada tarefa é efetuada mais de uma vez. Além disso, são declarados no início do algoritmo e podem ser chamados em qualquer ponto.

5.11.1 Procedimentos

A sintaxe do subalgoritmo procedimento.

Figura 15: Procedimento

```
Procedimento identificador()
Inicio
Lista de comandos
FimProcedimento
```

Fonte: Própria do Autor

5.12 Implementação de Alguns Resultados Matemáticos no VisuAlg

Nesta secção utilizaremos o visualg para implementação de operações básicas como produto entre matrizes, determinante de uma matriz, congruência modular, inverso multiplicativo modular ou recíproco.

5.12.1 Produto de Matrizes

Vamos criar um algoritmo que calcule o produto entre matrizes,

Exemplo 18. Seja $A = \begin{bmatrix} 8 & -9 \\ 4 & -2 \end{bmatrix}$ e $B = \begin{bmatrix} 6 & -2 \\ 7 & -5 \end{bmatrix}$. Calcular $A \cdot B$

Figura 16: Algoritmo do Produto Entre Matrizes

```
1 algoritmo "Produto de Matrizes"
2 var
3 A: vetor [1..2,1..2] de inteiro
5 PRODUTO: vetor [1..2,1..2] de inteiro
6 i, j: inteiro
7 inicio
8 escreval "-----------------------------------"
9 escreval " PRODUTO DE MATRIZES "
10 escreval "-----------------------------------"
11 Para i <- 1 ate 2 faca
12 Para j <- 1 ate 2 faca
13 escreva ""Digite um valor referente a matriz A [", i, ",", j,"]:")
14 leia(A[i,j])
15 fimpara
16 fimpara
17 escreval "-----------------------------------"
18 Para i <- 1 ate 2 faca
19 Para j <- 1 ate 2 faca
20 escreva ""Digite um valor referente a matriz B [", i, ",", j,"]:")
21 leia(B[i,j])
22 fimpara
23 fimpara
24 Para i <- 1 ate 2 faca
25 Para j <- 1 ate 2 faca
26 PRODUTO[i,j] <- (A[i,1] * B[1,j]) + (A[i,2] * B[2,j])
27 fimpara
28 fimpara
29 escreval "-----------------------------------"
30 escreval " PRODUTO DE (A*B) "
31 escreval "-----------------------------------"
32 Para i <- 1 ate 2 faca
33 Para j <- 1 ate 2 faca
6 escreva (PRODUTO[i,j]:5)
34 fimpara
35 escreval()
36 fimpara
37 fimalgoritmo
```

Fonte: Própria do Autor

Figura 17: Saída Algoritmo do Produto Entre Matrizes

```
Digite um valor referente a matriz A [ 1, 1]:8
Digite um valor referente a matriz A [ 1, 2]:-9
Digite um valor referente a matriz A [ 2, 1]:4
Digite um valor referente a matriz A [ 2, 2]:-2
--------------------------------------
Digite um valor referente a matriz B [ 1, 1]:6
Digite um valor referente a matriz B [ 1, 2]:-2
Digite um valor referente a matriz B [ 2, 1]:7
Digite um valor referente a matriz B [ 2, 2]:-5
--------------------------------------

PRODUTO DE (A*B)

--------------------------------------

-15     29
 10      2
*** Fim da execução.
*** Feche esta janela para retornar ao Visualg.
```

Fonte: Própria do Autor

5.12.2 Determinante de uma Matriz

Apresentaremos o algoritmo para calcular o determinante de uma matriz.

Exemplo 19. Seja $A = \begin{bmatrix} 8 & 4 \\ 3 & 5 \end{bmatrix}$. Construir um algoritmo para obter Det(A).

Figura 18: Algoritmo do Determinante

```
1 algoritmo "Determinate"
2 var
3 A: vetor [1..2,1..2] de inteiro
4 i, j, DET: inteiro
5 inicio
6 escreval " DETERMIANTE "
7 escreval "_____"
8 Para i <- 1 ate 2 faca
9 Para j <- 1 ate 2 faca
10 escreva ""Digite um valor referente a matriz A [", i, ",", j,"])
11 leia(A[i,j])
12 fimpara
13 fimpara
14 Para i <- 1 ate 2 faca
16 Para j <- 1 ate 2 faca
17 DET <- (A[1,1] * A[2,2]) - (A[1,2] * A[2,1])
18 fimpara
19 fimpara
20 escreval "_____"
21 escreval "O determinate da Matriz A=",det )
22 fimalgoritmo
```

Fonte: Própria do Autor

Figura 19: Saída Algoritmo do Determinante

```
DETERMIANTE
_____

Digite um valor referente a matriz A [ 1, 1]:8
Digite um valor referente a matriz A [ 1, 2]:4
Digite um valor referente a matriz A [ 2, 1]:3
Digite um valor referente a matriz A [ 2, 2]:5
_____

O determinate da Matriz A= 28
*** Fim da execução.
*** Feche esta janela para retornar ao Visualg.
```

Fonte: Própria do Autor

5.12.3 Congruência Modular

Para esta subseção abordaremos o comando mod, no qual o VisuAlg dispõem para obter o módulo de um inteiro qualquer.

Exemplo 20. Executar o comando. (82 mod 30).

Figura 20: Comando Modular e Saída

```
1 escreva ( 82 mod 30)
```

```
22
*** Fim da execução.
*** Feche esta janela para retornar ao Visualg.
```

Fonte: Própria do Autor

5.12.4 Recíproco ou Inverso Multiplicativo Modular

O visualg não dispõe de um comando que retorne o inverso multiplicativo modular de um inteiro positivo, no entanto criamos um algoritmo para resolver este problema, ver exemplo para recíproco modulo 15 na figura 21.

Para tal fim utilizamos o algoritmo estendido de Euclides, desse modo conseguimos encontrar os recíprocos modulo(m) de qualquer inteiro positivo. Caso m seja pequeno, basta utilizar o mecanismo rápido de Euclides para obter os recíprocos:

$$a \cdot x \equiv 1(mod\ m).$$

Por fim, utilizamos o comando relacional "escolha caso" para relacionar os inteiros com o seus recíprocos.

Exemplo 21. Criar um algoritmo que retorne todos os recíproco de x (mod 15).

Neste caso, Como m é convencionalmente pequeno, então pelo mecanismo rápido de Euclides para obter todos os recíprocos de x (mod 15).

Para $a = 1$ e $x = 1$ obtemos,

$$1 \cdot 1 \equiv 1(mod\ m)$$

Portanto o inteiro 1 possui recíproco, no qual é ele o própio. Assim consecutivamente os únicos inteiros que possuem recíprocos são $a = 1,\ a = 2,\ a = 4,\ a = 7,\ a = 8,\ a = 13,\ a = 14$. Seus recíprocos são respectivamente $x = 1,\ x = 2,\ x = 4,\ x = 7,\ x = 8,\ x = 13$ e $x = 14$.

Inserindo os dados no algoritmo.

Figura 21: Algoritmo do Recíproco Modular e Saída

```
1 algoritmo "Reciproco"
2 var
3 x, reciproco: inteiro
4 modulo: real
5 Inicio
6 escreval ("Digite um inteiro positivo:")
7 leia (x)
8 modulo <- (x mod 15)
9 escreval (x, " mod 15 =", modulo)
10 Escolha (modulo)
11 caso 1
12 reciproco <- 1
13 caso 2
14 reciproco <- 8
15 caso 4
16 reciproco <- 4
17 caso 7
18 reciproco <- 13
19 caso 8
20 reciproco <- 11
21 caso 13
22 reciproco <- 7
23 caso 14
24 reciproco <- 14
25 OutroCaso
26 escreval ( " Não possui recíproco ")
27 escreval ("O reciproco de ", x, " é ", reciproco)
28 Fimescolha
29 fimalgoritmo
```

```
Digite um inteiro positivo: 53
53 mod 15 = 8
O reciproco de 53 é 2
*** Fim da execução.
*** Feche esta janela para retornar ao Visualg.
```

Fonte: Própria do Autor

6 CRIPTOGRFIA

A criptografia consiste em tornar uma mensagem indecifrável, mesmo que um terceiro a intercepte, este não a compreenderá. Seu surgimento remonta aos tempos de guerra, no qual era imprescindível ocultar qual seria a próxima tática a ser realizada.

Em 475 a.C. Surgiu o primeiro sistema criptográfico por transposição, o bastão de Licurgo, no qual formada por duas hartes de mesmo diâmetro e uma tira de couro estreita e longa o suficiente para caber uma letra e escrever uma mensagem. Ou seja o que tornava a mensagem indecifrável era o fato das hastes possuírem o mesmo diâmetro.

Figura 22: Bastão de Licurgo espartano

Fonte: Enciclopédia livre (2023)

Datada de 50 a.C. A cifra de césar é uma cifra de substituição na qual cada letra do texto é substituída por outra, a partir de um determinado deslocamento. A exemplo definindo o deslocamento igual a 5 para as letras A e B. assim a letra A passará a ser F, já a letra B será substituída por G.

Figura 23: Cifra de César

Fonte: Enciclopédia livre (2023)

Em 1923, Arthur Scherbius tinha patenteado uma máquina de cifrar, sob o nome de ENIGMA, com vários modelos sendo vendidos comercialmente. Primeiramente a marinha alemã a adquiriu, depois exercito e força área, com diferentes modificações para cada força militar.

Para a criptografia moderna tem-se o método de criptografia RSA[3]. Desenvolvido em 1978 tendo como base para criptografar os números primos. Amplamente utilizado para transmissão segura de dados na internet; em compras online, emails (REINHOLD, 2020).

[3]iniciais dos criadores: Ron Rivest; Adi Shamir: Leonard Adleman,

7 IMPLEMENTAÇÃO DA CIFRA DE HILL

Neste capítulo iremos apresentar a teoria referente a Cifra[4] de Hill (BARRETO, 2014), em seguida implementando-o no programa VisuAlg.

7.1 Cifra de Hill

Desenvolvido por Lester S. Hill, em 1929. Esta Cifra, baseada na álgebra linear e teoria dos números é um método de criptografia por substituição de modo poligráfico, ou seja um conjunto de letras serão substituídas por um conjunto números, já o contrário para descriptografar. Em 7.1.1 será apresentado as etapas para criptografar, já em 7.2.2 para descriptografar.

7.1.1 Criptografar

O primeiro passo será definir uma chave codificadora, no qual satisfaça as seguintes condições: Ser $A_{(n\times n)}$ com elementos positivos, pois será necessário obter o $det(A)$ para a seguinte condição; verificar se $A_{(n\times n)}$ é inversível, além disso seja m o total de sinais gráficos, é necessário que $det(A)$ seja coprimo de m, ou seja, $mdc\ (det(A), m) = 1$, caso contrário não haverá recíprocos para $det(A)$ $(modulo\ m)$.

Tabela 5: Converter Letras Para Números

Tabela de conversão																								
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	0

Fonte: Própria do Autor

O Segundo passo será converter cada letra da mensagem pelo caracter numérico que fora atribuído, conforme a tabela 5, ou seja, é necessário que seja uma relação biunívoca [5] entre letras e números, em seguida agrupar os números sucessivamente em matrizes coluna, de acordo com a ordem da matriz A. Para $A_{(2\times 2)}$ agrupar em pares. $A_{(3\times 3)}$ em ternos, ou seja, $p_1,\ p_2,\ p_3,\ ...,\ p_n$ coluna tal que,

$$P = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{bmatrix}$$

Observação 4. A matriz coluna deverá ser totalmente preenchida, caso reste espaços preencha-os repetindo o último símbolo da mensagem que deseja codificar.

O terceiro passo será a codificação. Para isso realize o produto entre a chave codificadora A com as matrizes colunas P, tal que

$$A \cdot P = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \cdot \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

Observação 5. Ressaltamos que o produto entre matrizes não é comutativo, logo atentar para ordem $A \cdot P$.

[4] Algoritmo utilizado para criptografar ou descriptografar dados

[5] Dados dois conjuntos A e B, dizemos que eles estão em correspondência biunívoca quando a cada elemento de A corresponde um único elemento de B e reciprocamente.

O quarto passo será aplicar o (*modulo m*) em $\begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$, pois em decorrência do produto entre matrizes é comum b_1 e b_2 não terem equivalentes alfabéticos.

No quinto passo, retornar os caracteres numéricos para símbolos gráficos, de acordo com a tabela 5, seguido de enviar a mensagem.

7.1.2 Descriptografar

Supondo que o destinatário tenha recebido a mensagem junto a chave codificadora, este terá que seguir os seguintes passos.

Sexto passo, transformar a chave codificadora em decodificadora, para então obter a matriz inversa (*modulo m*) de (A).

De acordo com a definição 8, então supondo que a matriz A é invertível (*modulo m*),

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \cdots & \dots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix}$$

Se P é um vetor comum

$$P = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{bmatrix}$$

Então C é o vetor codificado

$$C = A \cdot P$$

E será o vetor decodificado

$$P = A^{-1} \cdot C$$

Pelo Teorema 2 temos

$$A^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} \tag{15}$$

O recíproco de $det(A)$ é

$$(ad - bc)^{-1} \tag{16}$$

Substituindo (16) em (15) e aplicando (*modulo m*) ao membro da direita.

$$A^{-1} = (ad - bc)^{-1} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} (mod\ m)$$

Seja m fixo e $a \in \mathbb{Z}_+$ com $0 < a < m$, é conveniente recorrer a uma tabela com os valores de todos os reciproco a^{-1} (*modulo m*).

Tabela 6: Tabela de Recorrência Módulo m

Recorrência (módulo m)							
A	a_1	a_2	a_3	a_4	a_5	$\cdots$	a_n
A^{-1}	a_1^{-1}	a_2^{-1}	a_3^{-1}	a_4^{-1}	a_5^{-1}	$\cdots$	a_n^{-1}

Fonte: Própria do Autor

Sexto passo, converter os símbolos gráficos novamente em números, seguindo a tabela 5, ademais reagrupar os termos conforme o passo 2.

Sétimo passo, decodificar

$$A^{-1} \cdot P = (ad - bc)^{-1} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} (mod\ m) \cdot \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

Oitavo passo, aplicar $(modulo\ m)$ em $\begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$

Nono passo, converter novamente números para símbolos conforme a tabela 5.

Exemplo 22. Codificar e decodificar a palavra MELANCIA.

A princípio temos que gerar uma chave criptográfica, assim norteado pela tabela 5. $A = \begin{bmatrix} 5 & 2 \\ 4 & 5 \end{bmatrix}$

A é quadrada de ordem (2×2), bem como inversível, pois $det(A) = 17$, que por sua vez o $mdc\ (17, 26) = 1$. Portanto 17 é coprimo de 26.

Condições satisfeitas, logo converter a palavra em vetores colunas $2 - cifra\ de\ Hill$.

Utilizando os valores correspondestes da tabela 5.

$$P_1 = \begin{bmatrix} M \\ E \end{bmatrix},\ P_2 = \begin{bmatrix} L \\ A \end{bmatrix},\ P_3 = \begin{bmatrix} N \\ C \end{bmatrix},\ P_4 = \begin{bmatrix} I \\ A \end{bmatrix}$$

$$P_1 = \begin{bmatrix} 13 \\ 5 \end{bmatrix},\ P_2 = \begin{bmatrix} 12 \\ 1 \end{bmatrix},\ P_3 = \begin{bmatrix} 14 \\ 3 \end{bmatrix},\ P_4 = \begin{bmatrix} 9 \\ 1 \end{bmatrix}$$

Realizando $A \cdot P$, em seguida aplicando $(modulo\ 26)$.

Aplicar o *modulo* 26 se faz necessário, pois as substituições de letras por números são de 0 a 25, caso número > 25 então não haveria substituições.

$$A \cdot P_1 = \begin{bmatrix} 5 & 2 \\ 4 & 5 \end{bmatrix} \begin{bmatrix} 13 \\ 5 \end{bmatrix} = \begin{bmatrix} 75 \\ 77 \end{bmatrix} (mod\ 26).$$

Pelo algoritmo da divisão,

$$75 = 26 \cdot 2 + 23$$

e

$$77 = 26 \cdot 2 + 25$$

logo,

$$75 \equiv 23\ (mod\ 26)$$

e

$$77 \equiv 25\ (mod\ 26).$$

Por isso, $\begin{bmatrix} 75 \\ 77 \end{bmatrix} (mod\ 26) = \begin{bmatrix} 23 \\ 25 \end{bmatrix} (mod\ 26) \Longrightarrow \begin{bmatrix} W \\ Y \end{bmatrix}$

$$A \cdot P_2 = \begin{bmatrix} 5 & 2 \\ 4 & 5 \end{bmatrix} \begin{bmatrix} 12 \\ 1 \end{bmatrix} = \begin{bmatrix} 62 \\ 53 \end{bmatrix} (mod\ 26).$$

Pelo algoritmo da divisão,

$$62 = 26 \cdot 2 + 10$$

e

$$53 = 26 \cdot 2 + 1$$

deste modo,

$$62 \equiv 10\ (mod\ 26)$$

e

$$53 \equiv 1\ (mod\ 26).$$

Assim sendo, $\begin{bmatrix} 62 \\ 53 \end{bmatrix} (mod\ 26) = \begin{bmatrix} 10 \\ 1 \end{bmatrix} (mod\ 26) \Longrightarrow \begin{bmatrix} J \\ A \end{bmatrix}$

$$A \cdot P_3 = \begin{bmatrix} 5 & 2 \\ 4 & 5 \end{bmatrix} \begin{bmatrix} 14 \\ 3 \end{bmatrix} = \begin{bmatrix} 76 \\ 71 \end{bmatrix} (mod\ 26).$$

Pelo algoritmo da divisão,

$$76 = 26 \cdot 2 + 24$$

e

$$71 = 26 \cdot 2 + 19$$

desta forma,

$$76 \equiv 24\ (mod\ 26)$$

e

$$71 \equiv 19\ (mod\ 26).$$

Por conseguinte, $\begin{bmatrix} 76 \\ 71 \end{bmatrix} (mod\ 26) = \begin{bmatrix} 24 \\ 19 \end{bmatrix} (mod\ 26) \Longrightarrow \begin{bmatrix} X \\ S \end{bmatrix}$

$$A \cdot P_4 = \begin{bmatrix} 5 & 2 \\ 4 & 5 \end{bmatrix} \begin{bmatrix} 9 \\ 1 \end{bmatrix} = \begin{bmatrix} 47 \\ 41 \end{bmatrix} (mod\ 26).$$

Pelo algoritmo da divisão,

$$47 = 26 \cdot 1 + 21$$

e

$$41 = 26 \cdot 1 + 15$$

desta maneira,

$$47 \equiv 21\ (mod\ 26)$$

e

$$41 \equiv 15\ (mod\ 26).$$

Como resultado, $\begin{bmatrix} 47 \\ 41 \end{bmatrix} (mod\ 26) = \begin{bmatrix} 21 \\ 15 \end{bmatrix} (mod\ 26) \Longrightarrow \begin{bmatrix} U \\ O \end{bmatrix}$

Portanto temos a cifra: WYJAXSUO.

Para decodificar a mensagem é necessário ter a inversa $(mod\ 26)$ da matriz chave, para isso pelo Teorema 2.

$$A^{-1} = (ad - bc)^{-1} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} (mod\ 26)$$

Sendo que, $(ad - bc)^{-1}$ é o recíproco de $det(A)$.

Dado Det(A)= 17.

Pelo mecanismo,

$$a \cdot x \equiv 1\ (mod\ 26)$$

Temos que,

$$17 \cdot 23 \equiv 1\ (mod\ 26)$$

Portanto $A^{-1} = 23$ é o recíproco de $Det(A) = 17$.

$$A^{-1} = 23 \begin{bmatrix} 5 & -2 \\ -4 & 5 \end{bmatrix} = \begin{bmatrix} 115 & -46 \\ -92 & 115 \end{bmatrix} = \begin{bmatrix} 11 & 6 \\ 12 & 11 \end{bmatrix} (mod\ 26)$$

Desse modo, realizando $A^{-1} \cdot (P_1, P_2, P_3, P_4)$.

$$A^{-1} \cdot P_1 = \begin{bmatrix} 11 & 6 \\ 12 & 11 \end{bmatrix} \begin{bmatrix} 23 \\ 25 \end{bmatrix} = \begin{bmatrix} 403 \\ 551 \end{bmatrix} (mod\ 26).$$

Pelo algoritmo da divisão,

$$403 = 26 \cdot 15 + 13$$

e

$$551 = 26 \cdot 21 + 5$$

desta forma,

$$403 \equiv 13\ (mod\ 26)$$

e

$$551 \equiv 5\ (mod\ 26).$$

Assim sendo, $\begin{bmatrix} 403 \\ 551 \end{bmatrix} (mod\ 26) = \begin{bmatrix} 13 \\ 5 \end{bmatrix} (mod\ 26) \Longrightarrow \begin{bmatrix} M \\ E \end{bmatrix}$

$$A^{-1} \cdot P_2 = \begin{bmatrix} 11 & 6 \\ 12 & 11 \end{bmatrix} \begin{bmatrix} 10 \\ 1 \end{bmatrix} = \begin{bmatrix} 116 \\ 131 \end{bmatrix} (mod\ 26).$$

Pelo algoritmo da divisão,

$$116 = 26 \cdot 4 + 12$$

e

$$131 = 26 \cdot 5 + 1$$

logo,

$$116 \equiv 12\ (mod\ 26)$$

e

$$131 \equiv 1\ (mod\ 26).$$

Por consequência, $\begin{bmatrix} 116 \\ 131 \end{bmatrix} (mod\ 26) = \begin{bmatrix} 12 \\ 1 \end{bmatrix} (mod\ 26) \Longrightarrow \begin{bmatrix} L \\ A \end{bmatrix}$

$$A^{-1} \cdot P_3 = \begin{bmatrix} 11 & 6 \\ 12 & 11 \end{bmatrix} \begin{bmatrix} 24 \\ 19 \end{bmatrix} = \begin{bmatrix} 378 \\ 497 \end{bmatrix} (mod\ 26).$$

Pelo algoritmo da divisão,

$$378 = 26 \cdot 14 + 14$$

e

$$497 = 26 \cdot 19 + 3$$

deste modo,

$$378 \equiv 14\ (mod\ 26)$$

e

$$497 \equiv 3\ (mod\ 26).$$

Por isso, $\begin{bmatrix}378\\497\end{bmatrix} (mod\ 26) = \begin{bmatrix}14\\3\end{bmatrix} (mod\ 26) \Longrightarrow \begin{bmatrix}N\\C\end{bmatrix}$

$A^{-1} \cdot P_4 = \begin{bmatrix}11 & 6\\12 & 11\end{bmatrix} \begin{bmatrix}21\\15\end{bmatrix} = \begin{bmatrix}321\\417\end{bmatrix} (mod\ 26)$.

Pelo algoritmo da divisão,

$$321 = 26 \cdot 12 + 9$$

e

$$417 = 26 \cdot 16 + 1$$

desta maneira,

$$321 \equiv 9\ (mod\ 26)$$

e

$$417 \equiv 1\ (mod\ 26).$$

Como resultado, $\begin{bmatrix}321\\417\end{bmatrix} (mod\ 26) = \begin{bmatrix}9\\1\end{bmatrix} (mod\ 26) \Longrightarrow \begin{bmatrix}I\\A\end{bmatrix}$

Portanto, a mensagem descriptografada é MELANCIA.

7.2 Implementação da Cifra de Hill no VisuAlg

No VisuAlg elaboramos um pseudocódigo limitado a criptografar e descriptografar mensagens com até 16 caracteres, além do mais os caracteres terão que ser escritos sem acentuação e sem espaço. Para esta implementação utilizaremos a mensagem MATEMATICABASICA.

A cada figura, iremos explanar a finalidade de cada bloco de linhas e por fim mostrar a saída de dados no qual foi gerado.

Na figura 24 linha 1 fora escrito o nome do algoritmo. De 2 a 5 são comentários, do qual faz refência ao nome do autor e data de criação. As variáveis apresentam-se nas linhas 7 a 25, observe que hà variáveis vetoriais, matriciais, caracteres e inteiras. De 30 a 48 são linhas que fazem parte do procedimento topo1, no qual quando chamado escreve letras do alfabeto a partir de determinados números, observe a relação da linha 30 e 31, se a variável $cifmod26$ for 1, logo a letra A será escrita na tela.

Figura 24: Nome do Algoritmo, Variáveis e Procedimento Topo1

```
1  Algoritmo "2-CIFRA DE HILL"
2  // Função :
3  // Autor : Henrique Reis
4  // Data : 10/10/2022
5  // Seção de Declarações
6  Var
7  letras: vetor [1..26] de caracter
8  numeros: vetor [1..26] de inteiro
9  chave: vetor[1..2 , 1..2] de inteiro
10 M: vetor[1..2, 1..8] de inteiro
11 cifra: vetor [1..2, 1..8] de inteiro
12 cifmod26: vetor [1..2, 1..8] de inteiro
13 msgdescrip:vetor [1..2, 1..8] de inteiro
14 cont: inteiro
15 i, j: inteiro
16 c: inteiro
17 det: inteiro
18 reciproco: inteiro
19 nome: caracter
20 numcaracter: inteiro
21 resposta: caracter
22 Invert: vetor [1..2, 1..2] de inteiro
23 chaveinvetmod26: vetor [1..2, 1..2] de inteiro
24 cod: vetor [1..2, 1..2] de inteiro
25 fm: vetor [1..2, 1..8] de inteiro
26
27 procedimento topo1()
28 inicio
29 se (cifmod26[i,j] = 1) entao
30    escreva("A")
31 senao
32    se (cifmod26[i,j] = 2) entao
33       escreva("B")
34    senao
35       se (cifmod26[i,j] = 3) entao
36          escreva("C")
37       senao
38          se (cifmod26[i,j] = 4) entao
39             escreva("D")
40          senao
41             se (cifmod26[i,j] = 5) entao
42                escreva("E")
43             senao
44                se (cifmod26[i,j] = 6) entao
45                   escreva("F")
46                senao
47                   se (cifmod26[i,j] = 7) entao
48                      escreva("G")
```

Fonte: Própria do Autor

As linhas 49 a 96 da figura 25 fazem parte da continuação do procedimento topo1, observe que a relação números letras continuam.

Figura 25: Continuação do Procedimento Topo1

```
49                  senao
50                     se (cifmod26[i,j] = 8) entao
51                        escreva("H")
52                     senao
53                        se (cifmod26[i,j] = 9) entao
54                           escreva("I")
55                        senao
56                           se (cifmod26[i,j] = 10) entao
57                              escreva("J")
58                           senao
59                              se (cifmod26[i,j] = 11) entao
60                                 escreva("K")
61                              senao
62                                 se (cifmod26[i,j] = 12) entao
63                                    escreva("L")
64                                 senao
65                                    se (cifmod26[i,j] = 13) entao
66                                       escreva("M")
67                                    senao
68                                       se (cifmod26[i,j] = 14) entao
69                                          escreva("N")
70                                       senao
71                                          se (cifmod26[i,j] = 15) entao
72                                             escreva("O")
73                                          senao
74                                             se (cifmod26[i,j] = 16) entao
75                                                escreva("P")
76                                             senao
77                                                se (cifmod26[i,j] = 17) entao
78                                                   escreva("Q")
79                                                senao
80                                                   se (cifmod26[i,j] = 18) entao
81                                                      escreva("R")
82                                                   senao
83                                                      se (cifmod26[i,j] = 19) entao
84                                                         escreva("S")
85                                                      senao
86                                                         se (cifmod26[i,j] = 20) entao
87                                                            escreva("T")
88                                                         senao
89                                                            se (cifmod26[i,j] = 21) entao
90                                                               escreva("U")
91                                                            senao
92                                                               se (cifmod26[i,j] = 22) entao
93                                                                  escreva("V")
94                                                               senao
95                                                                  se (cifmod26[i,j] = 23) entao
96                                                                     escreva("W")
```

Fonte: Própria do Autor

As linhas de 97 a 144 da figura 26 também fazem parte do procedimento topo1, observe que a partir da linha 107 as condicionais compostas são fechadas, finalizando o procedimento na linha 123. A partir da linha 134 inicia o procedimento topo2, no qual apresenta a mesma função do topo1, porém a variável a ser averiguada é *msgdescrip*

Figura 26: Fim do Procedimento Topo1 e Início do Topo2

```
97                                                                  senao
98                                                                     se (cifmod26[i,j] = 24) entao
99                                                                        escreva("X")
100                                                                    senao
101                                                                       se (cifmod26[i,j] = 25)  entao
102                                                                          escreva("Y")
103                                                                       senao
104                                                                          se (cifmod26[i,j] = 0) e (letras[cont] = "z") entao
105                                                                             escreva("Z")
106                                                                          fimse
107                                                                       fimse
108                                                                    fimse
109                                                                 fimse
110                                                              fimse
111                                                           fimse
112                                                        fimse
113                                                     fimse
114                                                  fimse
115                                               fimse
116                                            fimse
117                                         fimse
118                                      fimse
119                                   fimse
120                                fimse
121                             fimse
122                          fimse
123                       fimse
124                    fimse
125                 fimse
126              fimse
127           fimse
128        fimse
129     fimse
130   fimse
131 fimse
132 fimprocedimento
133
134 procedimento topo2()
135 Inicio
136 se (msgdescrip[i,j] = 1) entao
137    escreva("A")
138 senao
139    se (msgdescrip[i,j] = 2) entao
140       escreva("B")
141    senao
142       se  (msgdescrip[i,j] = 3) entao
143          escreva("C")
144       senao
```

Fonte: Própria do Autor

Na figura 27 o procedimento topo2 continua.

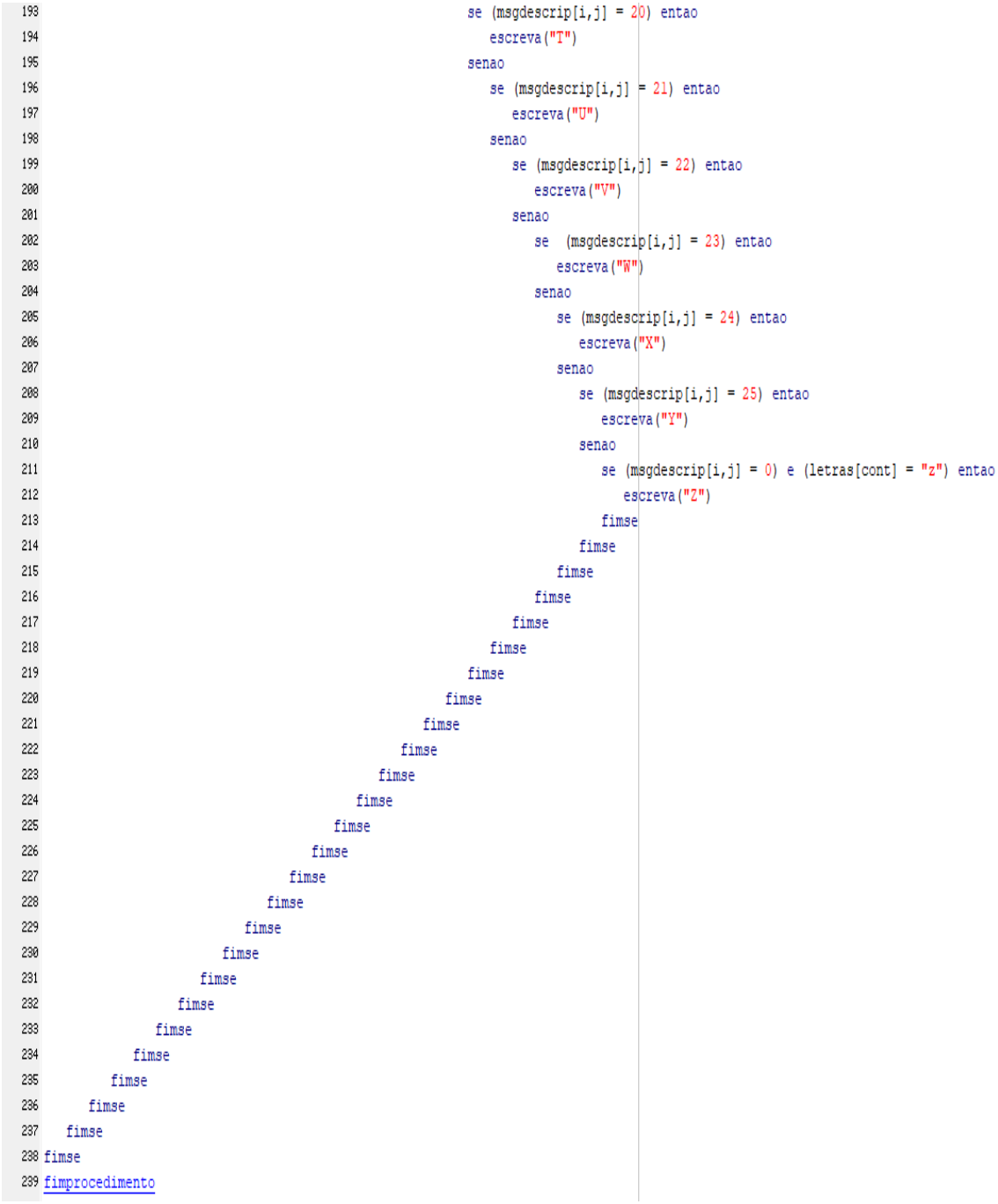
Figura 27: Continuação do Procedimento Topo2

```
145          se (msgdescrip[i,j] = 4) entao
146             escreva("D")
147          senao
148             se (msgdescrip[i,j] = 5) entao
149                escreva("E")
150             senao
151                se (msgdescrip[i,j] = 6) entao
152                   escreva("F")
153                senao
154                   se (msgdescrip[i,j] = 7) entao
155                      escreva("G")
156                   senao
157                      se (msgdescrip[i,j] = 8) entao
158                         escreva("H")
159                      senao
160                         se (msgdescrip[i,j] = 9) entao
161                            escreva("I")
162                         senao
163                            se (msgdescrip[i,j] = 10) entao
164                               escreva("J")
165                            senao
166                               se (msgdescrip[i,j] = 11) entao
167                                  escreva("K")
168                               senao
169                                  se (msgdescrip[i,j] = 12) entao
170                                     escreva("L")
171                                  senao
172                                     se (msgdescrip[i,j] = 13) entao
173                                        escreva("M")
174                                     senao
175                                        se (msgdescrip[i,j] = 14) entao
176                                           escreva("N")
177                                        senao
178                                           se (msgdescrip[i,j] = 15) entao
179                                              escreva("O")
180                                           senao
181                                              se (msgdescrip[i,j] = 16) entao
182                                                 escreva("P")
183                                              senao
184                                                 se  (msgdescrip[i,j] = 17) entao
185                                                    escreva("Q")
186                                                 senao
187                                                    se (msgdescrip[i,j] = 18) entao
188                                                       escreva("R")
189                                                    senao
190                                                       se (msgdescrip[i,j] = 19) entao
191                                                          escreva("S")
192                                                       senao
```

Fonte: Própria do Autor

Observe a figura 28. Analogamente ao procedimento anterior, há o fim do procedimento topo2 na linha 239.

Figura 28: Fim do Procedimento Topo2



Fonte: Própria do Autor

Na figura 29 a partir da linha 241 temos o inicio do algoritmo. Para a linha 245 temos um comando para perguntar o nome do usuário, já na linha 246 outro comando para armazenar o que fora escrito na linha anterior, nesse caso será armazenado na variável *nome*. O bloco de 250 a 254 é referente a tabela de conversão.

Para iniciar o processo de criptografar temos que; na linha 260 é solicitado que o usuário digite uma chave criptográfica de ordem 2×2, já na linha 261 os elementos desta matriz terão que ser inteiros positivos e seu determinante um coprimo de 26, ou seja $MDC(det, 26) = 1$.

As linhas 263 a 268 fazem parte do comando de repetição, no qual receberá os elementos da matriz e armazenará na variável *chave*.

As linhas 271 a 276 fazem com que os elementos da variável *chave* sejam escritos na tela de forma matricial.

As linhas 278 a 283 tem a função de calcular o determinate da matriz *chave* e armazená-lo na variável *det*.

A linha 285 pergunta a quantidade de letras da mensagem, já a linha 286 recebe este valor e o armazena na variável *numcaracter*.

Figura 29: Início do Algoritmo Cifra de Hill

```
241 inicio
242 escreval("================================================================================")
243 escreval("                          C I F R A   D E   H I L L                             ")
244 escreval("================================================================================")
245 escreva("Olá, qual o seu nome? ")
246 leia(nome)
247 escreval(nome,", seja bem vindo ao cifra de Hill!")
248 escreval("Este é um algoritmo para criptografar e descriptografar mensagens.          ")
249 escreval("================================================================================")
250 escreval("                    |T A B E L A   DE   C O N V E R S Ã O|                    ")
251 escreval("--------------------------------------------------------------------------------")
252 escreval("    A B C D E F G H I J  K  L  M  N  O  P  Q  R  S  T  U  V  W  X  Y  Z    ")
253 escreval("    1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 0    ")
254 escreval("--------------------------------------------------------------------------------")
255 escreval("================================================================================")
256 escreval("                                 CRIPTOGRAFAR                                   ")
257 escreval("================================================================================")
258 escreval("PARA CRIPTOGRAFAR UMA MENSAGEM SIGA AS SEGUINTES ORIENTAÇÕES.")
259 escreval("================================================================================")
260 escreval("Crie uma matriz de ordem (2X2), no qual será a chave criptográfica.")
261 escreval("Digite valores inteiros positivos, de modo que det satisfaça: mdc(det,26)=1")
262 escreval()
263 para i <- 1 ate 2 faca
264    para j <- 1 ate 2 faca
265       escreva("Digite um valor referente ao elemento [", i, ",",j, "]:")
266       leia(chave[i,j])
267    fimpara
268 fimpara
269 escreval("================================================================================")
270 escreval("A matriz abaixo será a chave criptográfica.")
271 para i <- 1 ate 2 faca
272    para j <- 1 ate 2 faca
273       escreva(chave[i,j]:5)
274    fimpara
275    escreval()
276 fimpara
277 escreval("================================================================================")
278 para i <- 1 ate 2 faca
279    para j <- 1 ate 2 faca
280       DET<- (chave[1,1] * chave[2,2]) - (chave[1,2] * chave[2,1])
281    fimpara
282 fimpara
283 escrevaL("det(chave)=", det)
284 escreval("================================================================================")
285 escreva("A mensagem a ser criptografada possui quantas letras?: ")
286 leia(numcaracter)
```

Fonte: Própria do Autor

As linhas 287 a 393 da figura 30 são um grande bloco de repetição aninhadas a condicionais compostas, no qual repetirá o processo (linha 288) inserção de letras, em seguida armazenando-as na variável *letras*. Além disso convertendo-as nos seus respectivo número e armazenando-os na variável *numeros*. Esse processo repetirá de acordo com o valor armazenado na variável *numcaracter*.

Figura 30: Inserção de Letras e Conversão

```
287 para cont <- 1 ate numcaracter faca
288    escreva("Digite a ", cont, "° letra:")
289    leia(letras[cont])
290    se (letras[cont] = "a") entao
291       numeros[cont] <- 1
292    senao
293       se (letras[cont] = "b") entao
294          numeros[cont] <- 2
295       senao
296          se (letras[cont] = "c") entao
297             numeros[cont] <- 3
298          senao
299             se (letras[cont] = "d") entao
300                numeros[cont] <- 4
301             senao
302                se (letras[cont] = "e") entao
303                   numeros[cont] <- 5
304                senao
305                   se (letras[cont] = "f") entao
306                      numeros[cont] <- 6
307                   senao
308                      se (letras[cont] = "g") entao
309                         numeros[cont] <- 7
310                      senao
311                         se (letras[cont] = "h") entao
312                            numeros[cont] <- 8
313                         senao
314                            se (letras[cont] = "i") entao
315                               numeros[cont] <- 9
316                            senao
317                               se (letras[cont] = "j") entao
318                                  numeros[cont] <- 10
319                               senao
320                                  se (letras[cont] = "k") entao
321                                     numeros[cont] <- 11
322                                  senao
323                                     se (letras[cont] = "l") entao
324                                        numeros[cont] <- 12
325                                     senao
326                                        se (letras[cont] = "m") entao
327                                           numeros[cont] <- 13
328                                        senao
329                                           se (letras[cont] = "n") entao
330                                              numeros[cont] <- 14
331                                           senao
332                                              se (letras[cont] = "o") entao
333                                                 numeros[cont] <- 15
334                                              senao
```

Fonte: Própria do Autor

Na figura 31 temos a continuação e finalização do processo de inserção da quantidade de letras e conversão para números.

Figura 31: Continuação do Processo de Inserção de Letras e Conversão em Números.

```
335                                se (letras[cont] = "p") entao
336                                   numeros[cont] <- 16
337                                senao
338                                   se (letras[cont] = "q") entao
339                                      numeros[cont] <- 17
340                                   senao
341                                      se (letras[cont] = "r") entao
342                                         numeros[cont] <- 18
343                                      senao
344                                         se (letras[cont] = "s") entao
345                                            numeros[cont] <- 19
346                                         senao
347                                            se (letras[cont] = "t") entao
348                                               numeros[cont] <- 20
349                                            senao
350                                               se (letras[cont] = "u") entao
351                                                  numeros[cont] <- 21
352                                               senao
353                                                  se (letras[cont] = "v") entao
354                                                     numeros[cont] <- 22
355                                                  senao
356                                                     se (letras[cont] = "w") entao
357                                                        numeros[cont] <- 23
358                                                     senao
359                                                        se (letras[cont] = "x") entao
360                                                           numeros[cont] <- 24
361                                                        senao
362                                                           se (letras[cont] = "y") entao
363                                                              numeros[cont] <- 25
364                                                           senao
365                                                              se (letras[cont] = "z") entao
366                                                                 numeros[cont] <- 0
367                                                              fimse
368                                                           fimse
369                                                        fimse
370                                                     fimse
371                                                  fimse
372                                               fimse
373                                            fimse
374                                         fimse
375                                      fimse
376                                   fimse
377                                fimse
378                             fimse
379                          fimse
380                       fimse
381                    fimse
382                 fimse
```

Fonte: Própria do Autor

Na figura 32 o bloco de linhas 396 a 404 escrevem na tela o resultado da conversão de letras para números.

De 407 a 426 os valores da variável *numeros* foram arranjados de modo a preencher a matriz M de ordem 4×4, ou seja 16 elementos. Este é o motivo pelo qual o algoritmo cifra até 16 caracteres. Já no bloco 428 a 432 a matriz M é escrita na tela.

No bloco 435 a 439 a mensagem é criptografada e armazenada na variável $cifra$. Posteriormente na figura 33 bloco 441 a 446, a mensagem é escrita na tela.

Figura 32: Organizando e Criptografando

```
393 fimpara
394 escreval("==============================================================================")
395 escreval("Conforme a tabela, as letras digitadas foram convertidas em números.")
396 para cont <- 1 ate numcaracter faca
397    se (numeros[cont] > 0)  entao
398       escreva ("[", numeros[cont], "]")
399    senao
400       se (numeros[cont] = "0") e (letras[cont] = "z") entao
401          escreva ("[0]")
402       fimse
403    fimse
404 fimpara
405 escreval
406 escreval("Agrupando-os em matrizes de ordem(2X1).")
407 para i <- 1 ate 2 faca
408    para j <- 1 ate numcaracter faca
409       M[1,1] <- numeros[1]
410       M[1,2] <- numeros[3]
411       M[2,1] <- numeros[2]
412       M[2,2] <- numeros[4]
413       M[1,3] <- numeros[5]
414       M[2,3] <- numeros[6]
415       M[1,4] <- numeros[7]
416       M[2,4] <- numeros[8]
417       M[1,5] <- numeros[9]
418       M[2,5] <- numeros[10]
419       M[1,6] <- numeros[11]
420       M[2,6] <- numeros[12]
421       M[1,7] <- numeros[13]
422       M[2,7] <- numeros[14]
423       M[1,8] <- numeros[15]
424       M[2,8] <- numeros[16]
425    fimpara
426 fimpara
427
428 para i <- 1 ate 2 faca
429    para j <- 1 ate 8 faca
430       escreva(M[i,j]:5)
431    fimpara
432    escreval()
433 fimpara
434 escreval("Mensagem criptografada.")
435 para i <- 1 ate 2 faca
436    para j <- 1 ate 8 faca
437       cifra[i,j] <- (chave[i,1] * M[1,j]) + (chave[i,2] * M[2,j])
438    fimpara
439 fimpara
```

Fonte: Própria do Autor

Na figura 33 bloco 449 a 468 é aplicado o *modulo* 26 nos elementos da matriz $cifra$, em seguida armazenando os novos resultados na variável matriz $cifmod26$. De 470 a 479 será escrito na tela o conteúdo da variável $cifmod26$.

Figura 33: Módulo Aplicado a Cifra

```
441 para i <- 1 ate 2 faca
442    para j <- 1 ate 8 faca
443       escreva(cifra[i,j]:5)
444    fimpara
445    escreval()
446 fimpara
447
448 escreval("Mensagem criptografada mod 26.")
449 para i <- 1 ate 2 faca
450    para j <- 1 ate 8 faca
451       cifmod26[1,1] <- cifra[1,1] mod 26
452       cifmod26[2,1] <- cifra[2,1] mod 26
453       cifmod26[1,2] <- cifra[1,2] mod 26
454       cifmod26[2,2] <- cifra[2,2] mod 26
455       cifmod26[1,3] <- cifra[1,3] mod 26
456       cifmod26[2,3] <- cifra[2,3] mod 26
457       cifmod26[1,4] <- cifra[1,4] mod 26
458       cifmod26[2,4] <- cifra[2,4] mod 26
459       cifmod26[1,5] <- cifra[1,5] mod 26
460       cifmod26[2,5] <- cifra[2,5] mod 26
461       cifmod26[1,6] <- cifra[1,6] mod 26
462       cifmod26[2,6] <- cifra[2,6] mod 26
463       cifmod26[1,7] <- cifra[1,7] mod 26
464       cifmod26[2,7] <- cifra[2,7] mod 26
465       cifmod26[1,8] <- cifra[1,8] mod 26
466       cifmod26[2,8] <- cifra[2,8] mod 26
467    fimpara
468 fimpara
469
470 para i <- 1 ate 2 faca
471    para j <- 1 ate 8 faca
472       se cifmod26[i,j] < 0 entao
473          escreva(cifmod26[i,j]+26:5)
474       senao
475          escreva(cifmod26[i,j]:5)
476       fimse
477    fimpara
478    escreval()
479 fimpara
```

Fonte: Própria do Autor

Na Figura 34 do bloco 481 a 521 escreve na tela a mensagem criptografada na forma linear.

Figura 34: Mensagem Cifrada e Aplicada o Módulo 26

```
481 escreval("Mensagem criptografada e na forma linear")
482 para i <- 1 ate 2 faca
483    para j <- 1 ate 1 faca
484       escreva(cifmod26[i,j]:4)
485    fimpara
486 fimpara
487 para i <- 1 ate 2 faca
488    para j <- 2 ate 2 faca
489       escreva(cifmod26[i,j]:4)
490    fimpara
491 fimpara
492 para i <- 1 ate 2 faca
493    para j <- 3 ate 3 faca
494       escreva(cifmod26[i,j]:4)
495    fimpara
496 fimpara
497 para i <- 1 ate 2 faca
498    para j <- 4 ate 4 faca
499       escreva(cifmod26[i,j]:4)
500    fimpara
501 fimpara
502 para i <- 1 ate 2 faca
503    para j <- 5 ate 5 faca
504       escreva(cifmod26[i,j]:4)
505    fimpara
506 fimpara
507 para i <- 1 ate 2 faca
508    para j <- 6 ate 6 faca
509       escreva(cifmod26[i,j]:4)
510    fimpara
511 fimpara
512 para i <- 1 ate 2 faca
513    para j <- 7 ate 7 faca
514       escreva(cifmod26[i,j]:4)
515    fimpara
516 fimpara
517 para i <- 1 ate 2 faca
518    para j <- 8 ate 8 faca
519       escreva(cifmod26[i,j]:4)
520    fimpara
521 fimpara
522 escreval()
523 escreval("================================================================================")
```

Fonte: Própria do Autor

Na figura 35 bloco 525 a 564, chama o procedimento topo1, no qual reconverterá os números para letras.

Figura 35: Chamada para o Procedimento Topo1

```
524 escreva("Mensagem criptografada: ")
525 para i <- 1 ate 2 faca
526    para j <- 1 ate 1 faca
527       topo1()
528    fimpara
529 fimpara
530 para i <- 1 ate 2 faca
531    para j <- 2 ate 2 faca
532       topo1()
533    fimpara
534 fimpara
535 para i <- 1 ate 2 faca
536    para j <- 3 ate 3 faca
537       topo1()
538    fimpara
539 fimpara
540 para i <- 1 ate 2 faca
541    para j <- 4 ate 4 faca
542       topo1()
543    fimpara
544 fimpara
545 para i <- 1 ate 2 faca
546    para j <- 5 ate 5 faca
547       topo1()
548    fimpara
549 fimpara
550 para i <- 1 ate 2 faca
551    para j <- 6 ate 6 faca
552       topo1()
553    fimpara
554 fimpara
555 para i <- 1 ate 2 faca
556    para j <- 7 ate 7 faca
557       topo1()
558    fimpara
559 fimpara
560 para i <- 1 ate 2 faca
561    para j <- 8 ate 8 faca
562       topo1()
563    fimpara
564 fimpara
565 escreval()
```

Fonte: Própria do Autor

Na figura 36, as linhas 566 a 574 versa sobre receber comandos para enviar ou não a mensagem criptografada.

Figura 36: Mecanismo Enviar Mensagem [S] ou [N]

```
566 escreva("Enviar mensagem? [S]ou[N]?")
567 leia(resposta)
568 se (resposta = "n") entao
569    escreva("Pressione a tecla 'ESC' para sair.")
570    leia(resposta)
571 senao
572    escreval("Enviando mensagem...")
573    escreval("Mensagem enviada!")
574 fimse
575 limpatela
```

Fonte: Própria do Autor

Na figura 37, o bloco 580 a 587 é destinado a receber comandos para decodificar ou não a mensagem.

Figura 37: Mecanismo Decodificar Mensagem [S] ou [N]

```
577 escreval("================================================================================")
578 escreval("                              DESCRIPTOGRAFAR                                   ")
579 escreval("================================================================================")
580 escreval("Você tem uma nova mensagem!")
581 escreva("Deseja descriptografa-la? [S]ou[N]?")
582 leia(resposta)
583 se (resposta = "n") entao
584    escreva("Pressione a tecla 'ESC' para sair.")
585    leia(resposta)
586 senao
587 fimse
588 escreval("================================================================================")
```

Fonte: Própria do Autor

Na figura 38, bloco 589 a 621 faz parte do processo para se obter o recíproco do determinante, este será armazenado na variável *reciproco*

Figura 38: Obter Recíproco

```
589 se det > 25 entao
590    det <- (det mod 26)
591 fimse
592
593 escolha det
594 caso 1
595    reciproco <- 1
596 caso 3
597    reciproco <- 9
598 caso 5
599    reciproco <- 21
600 caso 7
601    reciproco <- 15
602 caso 9
603    reciproco <- 3
604 caso 11
605    reciproco <- 19
606 caso 15
607    reciproco <- 7
608 caso 17
609    reciproco <- 23
610 caso 19
611    reciproco <- 11
612 caso 21
613    reciproco <- 5
614 caso 23
615    reciproco <- 17
616 caso 25
617    reciproco <- 25
618    se det > 25 entao
619       escreva(det mod 26)
620    fimse
621 fimescolha
```

Fonte: Própria do Autor

Na figura 39, a partir da chave codificadora o bloco 624 a 631 gera a matriz inversa, em seguida é armazenada na variável *invert*. Posteriormente o bloco 633 a 637 escreve na tela a *invert*.

A chave decodificadora é gerada a partir do bloco 643 a 647 e então armazenada na variável *chaveinvertmod*26. O bloco 649 a 654 escreve na tela a chave decodificadora.

Figura 39: Aplicação do Mecanismo Matriz Inversa · Recíproco

```
623 escreval("Matriz inversa da chave criptografica.")
624 para i <- 1 ate 2 faca
625    para j <- 1 ate 2 faca
626       Invert[1,1] <-   chave[2,2]
627       Invert[1,2] <- -(chave[1,2])
628       Invert[2,1] <- -(chave[2,1])
629       Invert[2,2] <-   chave[1,1]
630    fimpara
631 fimpara
632
633 para i <- 1 ate 2 faca
634    para j <- 1 ate 2 faca
635       escreva(Invert[i,j]:5)
636    fimpara
637    escreval()
638 fimpara
639 escreval("=============================================================================")
640 escreval("O recíproco do det","(",det,") é ","(",reciproco,").")
641 escreval("=============================================================================")
642 escreval("Produto da matriz inversa pelo recíproco")
643 para i <- 1 ate 2 faca
644    para j <- 1 ate 2 faca
645       chaveinvetmod26[i,j] <- reciproco * Invert[i,j]
646    fimpara
647 fimpara
648
649 para i <- 1 ate 2 faca
650    para j <- 1 ate 2 faca
651       escreva (chaveinvetmod26[i,j]:5)
652    fimpara
653    escreval()
654 fimpara
```

Fonte: Própria do Autor

Na figura 40, o bloco 458 a 665 aplica o módulo 26 a chave decodificadora, logo armazenando-a na variável *cod*, em seguida o bloco 667 a 672 escreve na tela a chave decodificadora.

Já o bloco 676 a 680 descriptografa a mensagem, assim armazenado-a na variável fm, posteriormente escrevendo-a na tela em 682 a 687.

Figura 40: Chave Módulo26 e Decodificação

```
656 escreval("Chave inversível mod 26.")
657
658 para i <- 1 ate 2 faca
659    para j <- 1 ate 2 faca
660       cod[1,1] <- chaveinvetmod26[1,1] mod 26
661       cod[1,2] <- chaveinvetmod26[1,2] mod 26 + 26
662       cod[2,1] <- chaveinvetmod26[2,1] mod 26 + 26
663       cod[2,2] <- chaveinvetmod26[2,2] mod 26
664    fimpara
665 fimpara
666
667 para i <- 1 ate 2 faca
668    para j <- 1 ate 2 faca
669       escreva(cod[i,j]:5)
670    fimpara
671    escreval()
672 fimpara
673
674 escreval("================================================================================")
675 escreval("Mensagem descriptografada.")
676 para i <- 1 ate 2 faca
677    para j <- 1 ate 8 faca
678       fm[i,j]<-(cod[i,1] * cifmod26[1,j]) + (cod[i,2] * cifmod26[2,j])
679    fimpara
680 fimpara
681
682 para i <- 1 ate 2 faca
683    para j <- 1 ate 8 faca
684       escreva(fm[i,j]:5)
685    fimpara
686    escreval ()
687 fimpara
```

Fonte: Própria do Autor

Na figura 41, o bloco 690 a 709 aplica o *modulo* 26 a mensagem descriptografada, logo armazenando-a em *msgdescrip*, em seguida escrevendo-a na tela em 711 a 716.

Figura 41: Aplicação do Módulo26

```
689 escreval("Mensagem descriptografada mod 26.")
690 para i <- 1 ate 2 faca
691    para j <- 1 ate 8 faca
692       msgdescrip[1,1] <- fm[1,1] mod 26
693       msgdescrip[1,2] <- fm[1,2] mod 26
694       msgdescrip[2,1] <- fm[2,1] mod 26
695       msgdescrip[2,2] <- fm[2,2] mod 26
696       msgdescrip[1,3] <- fm[1,3] mod 26
697       msgdescrip[1,4] <- fm[1,4] mod 26
698       msgdescrip[2,3] <- fm[2,3] mod 26
699       msgdescrip[2,4] <- fm[2,4] mod 26
700       msgdescrip[1,5] <- fm[1,5] mod 26
701       msgdescrip[1,6] <- fm[1,6] mod 26
702       msgdescrip[2,5] <- fm[2,5] mod 26
703       msgdescrip[2,6] <- fm[2,6] mod 26
704       msgdescrip[1,7] <- fm[1,7] mod 26
705       msgdescrip[1,8] <- fm[1,8] mod 26
706       msgdescrip[2,7] <- fm[2,7] mod 26
707       msgdescrip[2,8] <- fm[2,8] mod 26
708    fimpara
709 fimpara
710
711 para i <- 1 ate 2 faca
712    para j <- 1 ate 8 faca
713       escreva(msgdescrip[i,j]:5)
714    fimpara
715    escreval
716 fimpara
```

Fonte: Própria do Autor

Na figura 42 o bloco 718 a 758 escreve na tela a mensagem descriptografada e na forma linear.

Figura 42: Forma Linear

```
718 escreval("Mensagem descriptografada e na forma linear")
719 para i <- 1 ate 2 faca
720    para j <- 1 ate 1 faca
721       escreva(msgdescrip[i,j]:4)
722    fimpara
723 fimpara
724 para i <- 1 ate 2 faca
725    para j <- 2 ate 2 faca
726       escreva(msgdescrip[i,j]:4)
727    fimpara
728 fimpara
729 para i <- 1 ate 2 faca
730    para j <- 3 ate 3 faca
731       escreva(msgdescrip[i,j]:4)
732    fimpara
733 fimpara
734 para i <- 1 ate 2 faca
735    para j <- 4 ate 4 faca
736       escreva(msgdescrip[i,j]:4)
737    fimpara
738 fimpara
739 para i <- 1 ate 2 faca
740    para j <- 5 ate 5 faca
741       escreva(msgdescrip[i,j]:4)
742    fimpara
743 fimpara
744 para i <- 1 ate 2 faca
745    para j <- 6 ate 6 faca
746       escreva(msgdescrip[i,j]:4)
747    fimpara
748 fimpara
749 para i <- 1 ate 2 faca
750    para j <- 7 ate 7 faca
751       escreva(msgdescrip[i,j]:4)
752    fimpara
753 fimpara
754 para i <- 1 ate 2 faca
755    para j <- 8 ate 8 faca
756       escreva(msgdescrip[i,j]:4)
757    fimpara
758 fimpara
759 escreval()
760 escreval("================================================================================")
```

Fonte: Própria do Autor

Na figura 43, o bloco 762 a 801 chama o procedimento *topo*2, no qual de acordo com a tabela de conversão possibilita que os números da mensagem descriptografada sejam reconvertidos em letras. Assim finalizando o algoritmo.

Figura 43: Chamada Para Procedimento Topo2

```
761 escreva("Mensagem descriptografada: ")
762 para i <- 1 ate 2 faca
763    para j <- 1 ate 1 faca
764       topo2()
765    fimpara
766 fimpara
767 para i <- 1 ate 2 faca
768    para j <- 2 ate 2 faca
769       topo2()
770    fimpara
771 fimpara
772 para i <- 1 ate 2 faca
773    para j <- 3 ate 3 faca
774       topo2()
775    fimpara
776 fimpara
777 para i <- 1 ate 2 faca
778    para j <- 4 ate 4 faca
779       topo2()
780    fimpara
781 fimpara
782 para i <- 1 ate 2 faca
783    para j <- 5 ate 5 faca
784       topo2()
785    fimpara
786 fimpara
787 para i <- 1 ate 2 faca
788    para j <- 6 ate 6 faca
789       topo2()
790    fimpara
791 fimpara
792 para i <- 1 ate 2 faca
793    para j <- 7 ate 7 faca
794       topo2()
795    fimpara
796 fimpara
797 para i <- 1 ate 2 faca
798    para j <- 8 ate 8 faca
799       topo2()
800    fimpara
801 fimpara
802 escreval()
803 escreval("================================================================================")
804 fimalgoritmo
```

Fonte: Própria do Autor

Observe as figuras 44 e 45, do qual expressam a saída de dados do algoritmo.

Figura 44: Saída de Dados Parte 1

```
================================================================================
                          C I F R A   D E   H I L L
================================================================================
Olá, qual o seu nome? Henrique
Henrique, seja bem vindo ao cifra de Hill!
Este é um algoritmo para criptografar e descriptografar mensagens.
================================================================================
                    |T A B E L A   DE   C O N V E R S Ã O|
--------------------------------------------------------------------------------
    A B C D E F G H I J  K  L  M  N  O  P  Q  R  S  T  U  V  W  X  Y  Z
    1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 0
--------------------------------------------------------------------------------
================================================================================
                                  CRIPTOGRAFAR
================================================================================
PARA CRIPTOGRAFAR UMA MENSAGEM SIGA AS SEGUINTES ORIENTAÇÕES.
================================================================================
Crie uma matriz de ordem (2X2), no qual será a chave criptográfica.
Digite valores inteiros positivos, de modo que det satisfaça: mdc(det,26)=1

Digite um valor referente ao elemento [ 1, 1]:18
Digite um valor referente ao elemento [ 1, 2]:17
Digite um valor referente ao elemento [ 2, 1]:11
Digite um valor referente ao elemento [ 2, 2]:26
================================================================================
A matriz abaixo será a chave criptográfica.
   18   17
   11   26
================================================================================
det(chave)= 281
================================================================================
A mensagem a ser criptografada possui quantas letras?: 16
Digite a  1º letra:m
Digite a  2º letra:a
Digite a  3º letra:t
Digite a  4º letra:e
Digite a  5º letra:m
Digite a  6º letra:a
Digite a  7º letra:t
Digite a  8º letra:i
Digite a  9º letra:c
Digite a  10º letra:a
Digite a  11º letra:b
Digite a  12º letra:a
Digite a  13º letra:s
Digite a  14º letra:i
Digite a  15º letra:c
Digite a  16º letra:a
================================================================================
```

Fonte: Própria do Autor

Figura 45: Saída de Dados Parte 2

```
Conforme a tabela, as letras digitadas foram convertidas em números.
[ 13][ 1][ 20][ 5][ 13][ 1][ 20][ 9][ 3][ 1][ 2][ 1][ 19][ 9][ 3][ 1]
Agrupando-os em matrizes de ordem(2X1).
   13   20   13   20    3    2   19    3
    1    5    1    9    1    1    9    1
Mensagem criptografada.
  251  445  251  513   71   53  495   71
  169  350  169  454   59   48  443   59
Mensagem criptografada mod 26.
   17    3   17   19   19    1    1   19
   13   12   13   12    7   22    1    7
Mensagem criptografada e na forma linear
  17  13   3  12  17  13  19  12  19   7   1  22   1   1  19   7
================================================================================
Mensagem criptografada: QMCLQMSLSGAVAASG
Enviar mensagem? [S]ou[N]?s
Enviando mensagem...
Mensagem enviada!
================================================================================
                              DESCRIPTOGRAFAR
================================================================================
Você tem uma nova mensagem!
Deseja descriptografa-la? [S]ou[N]?s
================================================================================
Matriz inversa da chave criptografica.
   26  -17
  -11   18
================================================================================
O recíproco do det( 21) é ( 5).
================================================================================
Produto da matriz inversa pelo recíproco
  130  -85
  -55   90
Chave inversível mod 26.
    0   19
   23   12
================================================================================
Mensagem descriptografada.
  247  228  247  228  133  418   19  133
  547  213  547  581  521  287   35  521
Mensagem descriptografada mod 26.
   13   20   13   20    3    2   19    3
    1    5    1    9    1    1    9    1
Mensagem descriptografada e na forma linear
  13   1  20   5  13   1  20   9   3   1   2   1  19   9   3   1
================================================================================
Mensagem descriptografada: MATEMATICABASICA
================================================================================

Fim da execução.
```

Fonte: Própria do Autor

8 CONSIDERAÇÕES FINAIS

Neste trabalho foi apresentado um pseudocódigo para o método criptográfico cifra de Hill, desenvolvido sob os conteúdos matrizes, determinantes, congruência modular e algoritmo estendido de Euclides, em seguida explanamos algoritmos e lógica de programação, sendo este último explorado com mais destaque no capítulo referente ao programa visualg.

O processo de escrita do pseudocódigo se deu do seguinte modo: primeiramente foi dividido em etapas menores; elaboração matricial, substituição de letras por números, etc.., em seguida foi observado alguns padrões de repetição, logo refinamos o processo para então se obter um algoritmo mais elegante. Ao iniciante na programação, é estritamente necessário que o código funcione para depois torna-lo elegante, em outros termos, se (código funciona) então (torna-lo elegante).

O pseudocódigo foi desenvolvido pelo método básico de aprendizagem; tentativa e erro. Visto que não houve (ao menos não foi encontrado) outro código fonte ou pseudocódigo para basear-se. Devido ao conhecimento matemático, alguns blocos foram fáceis de ser construídos, como é o caso do produto entre matrizes, já outros demandaram mais tempo para encontrar uma solução, como é o caso do inverso multiplicativo modular. Ademais ao utilizar uma matriz quadrada de ordem 2 com limite para cifrar até 16 caracteres, gerou-se em torno de 800 linhas de códigos, assim caso o limite de caracteres fosse maior, o número de linhas de código seria muito maior.

A relevância deste trabalho refere-se ao ensino de conteúdos matemáticos alinhado a introdução a linguagem de programação; integrando as ciências matemáticas e computacionais, assim desenvolvendo o raciocínio computacional por meio da matemática.

Para trabalhos futuros propomos que a cifra de Hill seja aplicada em uma linguagem de programação, assim tornando-o mais eficiente ao expandir o limite de caracter a ser cifrado. Além disso, ainda no VisuAlg é possível implementar outros métodos criptográficos; cifra de César e cifra de Vigenère.

REFERÊNCIAS

ALENCAR, Edgar. **Teoria elementar dos números**. [S.I]. São Paulo: Nobel, 1981. 381 p.

ANTON, Howard; RORRES, Chris. **Álgebra linear com aplicações**. 10º. ed. Porto Alegre: Bookman, 2012. 768 p.

BARRETO, Regene. **Aritmética modular, códigos elementares e criptografia** 2014. 100 f. Dissertação de mestrado (Em Matemática) - Departamento de matemática, Universidade Federal de Sergipe, São Cristóvão. Disponível em: http://ri.ufs.br/jspui/handle/riufs/6508. Acesso em: 10 set. 2022.

BEZERRA, Nazaré. **Teoria dos Números:** um curso introdutório. 1º. ed. Belém: EditAed. 2018. 193. p.

COUTINHO, S.C. **Números inteiros e criptografia RSA**. 2º. ed. Rio de Janeiro: IMPA. 2007. 226 p.

FUNDAÇÃO BRADESCO. Escola virtual. in **Fundamentos de Lógica de Programação**. São paulo. 2022. https://www.ev.org.br/cursos/fundamentos-de-logica-de-programacao. Acesso em: 19 jul. 2022.

GUANABARA, Gustavo. Curso em video. In: **Algoritmos**. São paulo: Gustavo Guanabara,2014. Disponível em: https://www.cursoemvideo.com/curso/curso-de-algoritmo/. Acesso em: 4 jun. 2022.

HEFEZ, Abramo. **Elementos de aritmética**. Sociedade Brasileira de Matemática, Rio de Janeiro, 2006.

MANZANO, José: OLIVEIRA, Jayr. **Algoritmos Lógica para desenvolvimento de programação de computadores**, 29º. ed. São paulo: Érica, 2019. 368 p. Disponível em: https://books.google.com.br/books?id=c4uwDwAAQBAJ. Acesso em: 24 out. 2022.

MARTINS, Maria.**Ada Lovelace: a primeira programadora da história**. correio dos Açores. jul. 2016. Disponivel em: http://hdl.handle.net/10400.3/4025 Acesso em: 15 nov

MORGADO, Claudio; NICOLODI, Antonio. Visualg. In: MORGADO, Claudio; NICOLODI, Antonio. **Visualg**. 2.5. Blumenal, 1996. Disponível em: https://visualg3.com.br/. Acesso em: 13 maio 2022.

REINHOLD, Cleyson. **Criptografia - parte 1.** 2020. Disponível em:https://cgreinhold.dev/2020/03/13/criptografia/. Acesso em: 2 dez. 2022.

STEINBRUCH, Alfredo; WINTERLE, Paulo. **Álgebra linear**. 2º. ed. São Paulo: McGraw-Hill, 1987. 593 p.

A APÊNDICE - LINK PARA O PSEUDOCÓDIGO-FONTE CIFRA DE HILL

Links para o pseudocódigo-fonte referente a 2-cifra de Hill.
Clique aqui para baixar o Pseudocódigo no GitHub
Ou
Clique aqui para baixar o Pseudocódigo do Google Drive