Giovanni de Aguirre Tamanini

Desafio 1 - Montagem de uma Criptografia

Florianópolis

2022

Giovanni de Aguirre Tamanini

Desafio 1 - Montagem de uma Criptografia

Trabalho apresentado à disciplina Matemática e Estatística do Curso de Análise e Desenvolvimento de Sistemas do Senai/SC.
Professor: Rafael Bomaro Ferreira

Senai / SC

Graduação em Análise e Desenvolvimento de Sistemas

Florianópolis

2022

Sumário

1 Entendendo o desafio

Como objetivo geral do desafio tem-se a necessidade de desenvolver uma estratégia de criptografia para que seja possível realizar o envio de uma mensagem sem que alguém indesejado descubra seu real conteúdo. Para tal, algumas diretrizes foram dadas pelo professor, como por exemplo a relação de cada caractere (letra do alfabeto mais caracteres esepeciais) com números inteiros. Abaixo, a Figura 1 demonstra a relação.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
P	Q	R	S	T	U	V	W	X	Y	Z	.	,	!	?	[ESPAÇO]
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Figura 1 – Relação caracteres/inteiros

Tendo a relação é possível utilizá-la numa mensagem que poderá ser enviada. Como critério de avaliação, a mensagem a ser enviada é a seguinte: ESTUDAR PARA TRANSFORMAR O MUNDO!

Portanto deve ser gerada uma matriz que represente a mensagem original utilizando a relação dada. Porém, a matriz previamente gerada ainda não apresenta segurança suficiente para o envio da mensagem, pois a relação é elementar. Para melhorar isso faz-se necessário o uso de uma estratégia de criptografia, e a estratégia que será utilizada envolve operações entre matrizes.

Como dado importante do desafio, tem-se a matriz 2x2 COD (Figura 2), que é o codificador que será usado na matriz mensagem que será implementada em tópico posterior.

COD	4	1
	3	1

Figura 2 – Matriz COD (codificadora)

Por fim, não basta apenas conseguir a matriz codificada. É necessário que o receptor tenha uma forma de decodificar essa matriz e ler a mensagem original. No tópico Lógica de resolução do desafio, todas essas questões serão sanadas.

2 Desenvolvimento do desafio

2.1 Lógica de resolução do desafio

Primeiramente, utilizando a relação dada (Figura 1), faz-se a transformação de caracteres para números inteiros da mensagem que deve ser enviada. Abaixo, a transformação é feita (Figura 3 para Figura 4) :

```
{ {E, S, T, U, D, A, R, espaco, P, A, R, A, espaco, T, R, A, N},
  {S, F, O, R, M, A, R, espaco, O, espaco, M, U, N, D, O, exclamacao, espaco} };
```

Figura 3 – Mensagem original

```
5 19 20 21 4 1 18 31 16 1 18 1 31 20 18 1 14
19 6 15 18 13 1 18 31 15 31 13 21 14 4 15 29 31
```

Figura 4 – Mensagem transformada

A mensagem, originalmente possui 33 caracteres. Percebe-se que nas duas imagens a mensagem foi organizada em duas linhas com 17 caracteres cada. Isso não é por acaso. Para conseguirmos codificar a mensagem utilizando a matriz COD já apresentada, iremos multiplicar a matriz COD pela matriz numérica mensagem. Para isso a condição de multiplicação de matrizes deve ser respeitada, ou seja, o número de colunas da primeira matriz (COD), deve ser igual ao número de linhas da segunda matriz (será chamada de MENSAGEM). Para fechar duas linhas de 17 caracteres, foi adicionada no trigésimo quarto caractere um espaço (representado pelo inteiro 31).

A operação é feita multiplicando-se os termos da linha da primeira matriz COD(2x2) pelos termos da coluna da segunda matriz MENSAGEM(2x17), somando os resultados e guardando-os na posição correta, como indicado a seguir para as posições $CODIFICADA_{11}, CODIFICADA_{21}$, e assim por diante para as 34 posições de CODIFICADA(2x17).

$$\text{COD} = \begin{bmatrix} COD_{11} & COD_{12} \\ COD_{21} & COD_{22} \end{bmatrix}$$

$$CODIFICADA_{11} = COD_{11} \times MENSAGEM_{11} + COD_{12} \times MENSAGEM_{21}$$

$$CODIFICADA_{21} = COD_{21} \times MENSAGEM_{11} + COD_{22} \times MENSAGEM_{21}$$

A matriz resultante é a matriz chamada de CODIFICADA(2x17). Essa é a matriz que pode ser enviada ao receptor.

Quando o receptor receber a matriz CODIFICADA ele deverá aplicar um método de decodificação para mostrar a mensagem original. Para isso utiliza-se a matriz inversa de COD, chamada DECOD. Como COD é uma matriz 2x2, a inversa de uma matriz 2x2 pode ser calculada pela fórmula:

$$DECOD = \frac{1}{det(COD)} \times adj(COD)$$

onde,

$$det(COD) = COD_{11} \times COD_{22} - COD_{12} \times COD_{21}$$

$$\text{adj(COD)} = \begin{bmatrix} COD_{22} & -COD_{12} \\ -COD_{21} & COD_{11} \end{bmatrix}$$

Tendo a matriz DECOD(2x2), multiplica-se esta pela matriz CODIFICADA(2x17) para decodificar a matriz e por fim voltando a matriz MENSAGEM numérica. Por último usa-se a tabela de relação número inteiro/caractere, para voltarmos a mensagem original.

2.2 Resolução utilizando programação em C++

Optou-se pela resolução utilizando programação no paradigma procedural da linguagem C++, utilizando o Visual Studio Code. Primeiramente foi criado um arquivo chamado cripto.h (Header File no C++), onde todas as funções foram prototipadas. Além disso neste arquivo declarou-se a relação dos caracteres e números inteiros, as matrizes COD e DECOD utilizadas na resolução do problema além das funções "codifica"e "decodifica"com suas implementações. A função "decodifica"por sua vez, além de decodificar a matriz CODIFICADA, ela retorna no console a mensagem original que foi recebida. A seguir a Figura 5 mostra o início do *header file* cripto.h. Este arquivo reúne todas declarações de variáveis e funções importantes para a resolução das duas problemáticas de criptografia e descriptografia da mensagem. Mesmo tendo alguma experiência prévia com programação no

paradigma procedural em C++, se fez necessário o uso de algumas referências para tirar algumas dúvidas quanto a implementação da multiplicação de matrizes no site Stack Overflow (OVERFLOW, b) e (OVERFLOW, a).

```
1  //cripto.h
2  //11 de outubro de 2022
3  //Autor: Giovanni de Aguirre Tamanini
4  /*Projeto realizado para o Desafio 1 proposto na disciplina Matemática e Estatística do curso
5  Análise e Desenvolvimento de Sistemas do SENAI-SC*/
6  //Professor: Rafael Bomaro Ferreira
7
8  #include <iostream>
9  using namespace std;
10
11 //Declarando protótipo das funções
12 void mostra2x2(int arr_1[][2]);
13 void mostra2x17(int arr_2[][17]);
14 void codifica(int cod[][2], int mensagem[][17]);
15 void decodifica(int decod[][2], int codificada[][17]);
16
17 //Declarando valores numéricos inteiros para cada caractere
18 int A = 1, B = 2, C = 3, D = 4, E = 5, F = 6, G = 7, H = 8, I = 9, J = 10, K = 11, L = 12, M = 13, N = 14,
19 O = 15, P = 16, Q = 17, R = 18, S = 19, T = 20, U = 21, V = 22, W = 23, X = 24, Y = 25, Z = 26,
20 ponto = 27, virgula = 28, exclamacao = 29, interrogacao = 30, espaco = 31;
21
22 //Declarando matriz COD e DECOD (DECOD é a inversa de COD, foi calculada externamente ao programa
23 int cod[2][2] = { {4, 1}, {3, 1} };
24 int decod[2][2] = { {1, -1}, {-3, 4} };
```

Figura 5 – Declarando protótipos de funções, matrizes COD e DECOD e valores inteiros para os caracteres

As próximas figuras, Figuras 6, 7, 8, 9, 10 e 11, mostram a continuação do arquivo cripto.h, com todas as funções importantes implementadas, como a "codifica"e "decodifica"(principais funções da aplicação) e também a "mostra2x2"e "mostra2x17", que mostram no console quando chamadas, matrizes do tipo 2x2 e 2x17 respectivamente.

```cpp
//Implementando as funções (métodos)
//Função que mostra no console matriz 2x2
void mostra2x2(int arr_1[][2])
{
    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 2; j++) {
            cout << arr_1[i][j] << " ";
        }
        cout << endl;
    }
}

//Função que mostra no console matriz 2x17
void mostra2x17(int arr_2[][17])
{
    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 17; j++) {
            cout << arr_2[i][j] << " ";
        }
        cout << endl;
    }
}
```

Figura 6 – Implementando funções mostra2x2 e mostra2x17

```cpp
//Função que codifica matriz mensagem original, usando COD
void codifica(int cod[][2], int mensagem[][17])
{
    cout << "Criptografando matriz mensagem usando COD... \n" << endl;
    cout << "Matriz criptografada:" << endl;
    int codificada[2][17] = { {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                              {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0} };

    for (int linha = 0; linha < 2; linha++) {
        for (int coluna = 0; coluna < 17; coluna++) {
            for (int interno = 0; interno < 2; interno++) {
                /*interno = variável que permite multiplicação de linha por coluna somando resultado
                ao valor inicial da posição indicada pelos dois loops externos*/
                codificada[linha][coluna] += cod[linha][interno] * mensagem[interno][coluna];
            }
            cout << codificada[linha][coluna] << " ";
        }
        cout << "\n";
    }
    cout << endl;
}
```

Figura 7 – Implementando função codifica

```
71  /*Função que decodifica matriz anteriormente codificada por COD (utilizando DECOD)
72  e mostra mensagem descriptografada*/
73  void decodifica(int decod[][2], int codificada[][17])
74  {
75      cout << "Descriptografando matriz usando DECOD..." << endl;
76      int mensagem[2][17] = { {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
77                              {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0} };
78      for (int linha = 0; linha < 2; linha++) {
79          for (int coluna = 0; coluna < 17; coluna++) {
80              for (int interno = 0; interno < 2; interno++) {
81                  mensagem[linha][coluna] += decod[linha][interno] * codificada[interno][coluna];
82              }
83              //cout << mensagem[linha][coluna] << " ";
84          }
85          //cout << "\n";
86      }
87      cout << endl;
88      cout << "Mensagem descriptografada: \n";
89      for (int linha = 0; linha < 2; linha++) {
90          for (int coluna = 0; coluna < 17; coluna++) {
91              switch (mensagem[linha][coluna]) {
92              case 1:
93                  cout << "A";
94                  break;
95              case 2:
96                  cout << "B";
97                  break;
98              case 3:
99                  cout << "C";
100                 break;
```

Figura 8 – Implementando função decodifica - início

```
101             case 4:
102                 cout << "D";
103                 break;
104             case 5:
105                 cout << "E";
106                 break;
107             case 6:
108                 cout << "F";
109                 break;
110             case 7:
111                 cout << "G";
112                 break;
113             case 8:
114                 cout << "H";
115                 break;
116             case 9:
117                 cout << "I";
118                 break;
119             case 10:
120                 cout << "J";
121                 break;
122             case 11:
123                 cout << "K";
124                 break;
125             case 12:
126                 cout << "L";
127                 break;
128             case 13:
129                 cout << "M";
130                 break;
```

Figura 9 – Implementando função decodifica - continuação

```
131             case 14:
132                 cout << "N";
133                 break;
134             case 15:
135                 cout << "O";
136                 break;
137             case 16:
138                 cout << "P";
139                 break;
140             case 17:
141                 cout << "Q";
142                 break;
143             case 18:
144                 cout << "R";
145                 break;
146             case 19:
147                 cout << "S";
148                 break;
149             case 20:
150                 cout << "T";
151                 break;
152             case 21:
153                 cout << "U";
154                 break;
155             case 22:
156                 cout << "V";
157                 break;
158             case 23:
159                 cout << "W";
160                 break;
```

Figura 10 – Implementando função decodifica - continuação

```
161             case 24:
162                 cout << "X";
163                 break;
164             case 25:
165                 cout << "Y";
166                 break;
167             case 26:
168                 cout << "Z";
169                 break;
170             case 27:
171                 cout << ".";
172                 break;
173             case 28:
174                 cout << ",";
175                 break;
176             case 29:
177                 cout << "!";
178                 break;
179             case 30:
180                 cout << "?";
181                 break;
182             case 31:
183                 cout << " ";
184                 break;
185             default:
186                 cout << "Erro";
187             }
188         }
189     }
190     cout << endl;
191 }
```

Figura 11 – Implementando função decodifica - final

2.2.1 Saídas no console das aplicações

Para a resolução do problema foi utilizado o editor de códigos Visual Studio Code, adicionando a instalação de extensões para a linguagem C++, como o C/C++ IntelliSense, debugging, and code browsing. Os testes foram feitos utilizando Linux-Ubuntu com o compilador GDB (GNU Project Debugger) na versão 12.0.90. Quando a aplicação criptografando_mensagem.cpp (Figura 12) foi compilada e executada a saída no console como o esperado foi o demonstrado na Figura 13.

```cpp
//criptografando_mensagem.cpp
//11 de outubro de 2022
//Autor: Giovanni de Aguirre Tamanini
/*Projeto feito para realização do Desafio 1 proposto na disciplina Matemática e Estatística do curso
Análise e Desenvolvimento de Sistemas do SENAI-SC*/
//Professor: Rafael Bomaro Ferreira

#include <iostream>
#include "cripto.h" //Incluindo cripto.h
using namespace std;

int main()
{
    cout << "Mensagem a ser enviada: ESTUDAR PARA TRANSFORMAR O MUNDO!\n" << endl;

    /* Abaixo é possível modificar a mensagem, desde que seja respeitado o limite de caracteres (34) pois
    a mensagem deve ser guardada em uma matriz 2x17. A relação caracteres/inteiros está definido no header
    file cripto.h */
    int mensagem[2][17] = { {E, S, T, U, D, A, R, espaco, P, A, R, A, espaco, T, R, A, N},
                            {S, F, O, R, M, A, R, espaco, O, espaco, M, U, N, D, O, exclamacao, espaco} };
    mostra2x17(mensagem);
    cout << endl;

    /*Chamada da função codifica que utiliza como parâmetros de entrada a matriz COD e a matriz mensagem
    declarada acima. A implementação da função está no header file cripto.h*/
    codifica(cod, mensagem);

    cout << "Copiar matriz criptografada no console e enviar para o receptor." << endl;

    return 0;
}
```

Figura 12 – Arquivo .cpp que chama método de criptografia e devolve matriz criptografada

```
Mensagem a ser enviada: ESTUDAR PARA TRANSFORMAR O MUNDO!

5 19 20 21 4 1 18 31 16 1 18 1 31 20 18 1 14
19 6 15 18 13 1 18 31 15 31 13 21 14 4 15 29 31

Criptografando matriz mensagem usando COD...

Matriz criptografada:
39 82 95 102 29 5 90 155 79 35 85 25 138 84 87 33 87
34 63 75 81 25 4 72 124 63 34 67 24 107 64 69 32 73

Copiar matriz criptografada no console e enviar para o receptor.
[1] + Done                    "/usr/bin/gdb" --interpreter=mi --tty=${DbgTerm} 0<"/tmp/Microsoft-MIEngine-In-l53mwugf.wlq" 1>"/tmp/Microsof
t-MIEngine-Out-wvem0tel.0nz"
giovanni@lenovoubuntu:~/Documents/programming/c-plus-plus/matematica-e-estatistica/Desafio1$
```

Figura 13 – Saída no console para a aplicação criptografando_mensagem.cpp

Por fim utilizando a outra aplicação descriptografando_mensagem.cpp (Figura 14) a partir da matriz CODIFICADA, com a chamada da funçao decodifica, o programa retorna a mensagem original "ESTUDAR PARA TRANSFORMAR O MUNDO!"como mostrado na Figura 15.

```cpp
//descriptografando_mensagem.cpp
//11 de outubro de 2022
//Autor: Giovanni de Aguirre Tamanini
/*Projeto feito para realização do Desafio 1 proposto na disciplina Matemática e Estatística do curso
Análise e Desenvolvimento de Sistemas do SENAI-SC*/
//Professor: Rafael Bomaro Ferreira

#include <iostream>
#include "cripto.h" //Incluindo cripto.h
using namespace std;

int main()
{
    cout << "Matriz com mensagem criptografada recebida: " << endl;

    /*Abaixo adiciona-se a matriz codificada 2x17 com 34 caracteres*/
    int codificada[2][17] = { {39, 82, 95, 102, 29, 5, 90, 155, 79, 35, 85, 25, 138, 84, 87, 33, 87},
                              {34, 63, 75, 81, 25, 4, 72, 124, 63, 34, 67, 24, 107, 64, 69, 32, 73} };
    mostra2x17(codificada);
    cout << endl;

    /*Chamada da função decodifica que utiliza como parâmetros de entrada a matriz DECOD e a matriz
    codificada declarada acima. A implementação da função está no header file cripto.h*/
    decodifica(decod, codificada);

    return 0;
}
```

Figura 14 – Arquivo .cpp que chama método de descriptografia e devolve mensagem original

```
Matriz com mensagem criptografada recebida:
39 82 95 102 29 5 90 155 79 35 85 25 138 84 87 33 87
34 63 75 81 25 4 72 124 63 34 67 24 107 64 69 32 73

Descriptografando matriz usando DECOD...

Mensagem descriptografada:
ESTUDAR PARA TRANSFORMAR O MUNDO!
[1] + Done                       "/usr/bin/gdb" --interpreter=mi --tty=${DbgTerm} 0<"/tmp/Microsoft-MIEngine-In-clz2gycp.fv4" 1>"/tmp/Microsof
t-MIEngine-Out-lnsqobbx.1g2"
giovanni@lenovoubuntu:~/Documents/programming/c-plus-plus/matematica-e-estatistica/Desafio1$
```

Figura 15 – Saída no console para a aplicação descriptografando_mensagem.cpp

De maneira geral, o desafio foi de grande valia, pois permitiu o desenvolvimento de habilidades já conhecidas, assim como o aprendizado de novas habilidades. Para a visualização e teste do projeto desenvolvido, segue *link* com o repositório utilizado no GitHub (TAMANINI,).

Referências

OVERFLOW, I. do S. *Optimized matrix multiplication in C*. Disponível em: <https://stackoverflow.com/questions/1907557/optimized-matrix-multiplication-in-c>. Citado na página 7.

OVERFLOW, I. do S. *Write function to multiply matrices in C++*. Disponível em: <https://stackoverflow.com/questions/37001237/write-function-to-multiply-matrices-in-c>. Citado na página 7.

TAMANINI, G. de A. *Matemática e Estatística*. Disponível em: <https://github.com/giovannitamanini/matematica-e-estatistica>. Citado na página 14.