

## Criptografia: Parte 7 (Cifra de Vigenère-3)

Mário Leite

...

Aqui está o programa completo da “Cifra de Vigenère” (em Pseudocódigo) que pode ser convertido em qualquer linguagem de programação.

---

*Continua com o “Método RSA-1” (Parte 8)*

```
Programa "ProgCifraVigenere"
//Programa de cifragem/decifragem pelo método "Cifra de Vigenère"
//Em Pseudocódigo
//Autor: Mário Leite - Copyright (2017)
//E-mail: marleite@gmail.com
//-----
//Variáveis globais
Declare MatVig: array[1..26,1..26] de caractere
    MensagemOrig, MensagemCif, Chave: caractere
    VetChaveTot, VetChave: array[1..100] de caractere
    VetMsgCif, VetMsgDecif, VetMsgOrig: array[1..100] de caractere
    x, Tracos, TamMsgOrig, TamMsgCif, TamChave: inteiro
    Opcao, Msg, Car: caractere
//-----

Função FunVerifTexto(Texto,Msg:caractere; TamTexto:inteiro): logico
//Valida a mensagem ou a chave para considerar apenas letras.
Início
    Car ← "X"
    Se (TamTexto<=100) Então
        Selecione Msg
            Caso "Msg"
                Para x De 1 Até TamTexto Faça
                    VetMsgOrig[x] ← Copia(Texto,x,1)
                    Se ((Asc(VetMsgOrig[x])<65) ou (Asc(VetMsgOrig[x])>90)) Então
                        Car ← "?"
                        Abandone //encontrou caractere inválido na mensagem (sai do loop)
                    FimSe
                FimPara
            Caso "Chv"
                Para x De 1 Até TamTexto Faça
                    VetChave[x] ← Copia(Texto,x,1)
                    Se ((Asc(VetChave[x])<65) ou (Asc(VetChave[x])>90)) Então
                        Car ← "?"
                        Abandone //encontrou caractere inválido na chave (sai do loop)
                    FimSe
                FimPara
            CasoContrário
                Car ← "?"
        FimSelecione
    FimSe
    Se ((Car="?") ou (TamTexto>100)) Então
        Retorne Falso
    Senão
        Retorne Verdadeiro
    FimSe
FimFunção //fim da função "FunVerifTexto"
```

```

//-----
Procedimento ProGeraGrelha
//Monta a "Grelha de Vigenère" como uma matriz 26x26 de letras.
  Declare i, j, k, m, n: inteiro
Início
  EscrevaLn("A B C E R F G H I J K L M N O P Q R S T U V W X Y Z")
  Para i De 1 Até 26 Faça
    Escreva(Carac(i+64))
    EscrevaLn(" ")
    Para j De 1 Até 26 Faça
      k ← 63 + i + j //primeira linha da tabela considerada (A-Z sem exceção)
      Se(Asc(Carac(k))>90) Então //primeira exceção (segunda linha)
        k ← 90 - (27-i)
      FimSe
      MatVig[i,j] ← Carac(k)
      Escreva(" ", MatVig[i,j], " ") //imprime letra
      Se((MatVig[i,j]="Z") e (j<26)) Então //encontrou "Z" antes do fim da linha
        k ← 0
        m ← j + 1
        Para n De m Até 26 Faça
          k ← k + 1
          MatVig[i,n] ← Carac(64+k) //imprime a partir da letra "A"
          Escreva(" ",MatVig[i,n], " ")
        FimPara
      Abandone //abandona o loop j para pegar nova linha da grelha
    FimSe
  FimPara
  EscrevaLn("") //escreve uma linha em branco (salta linha)
FimPara
FimProcedimento //fim do procedimento "ProGeraGrelha"
//-----

Procedimento ProMontaEsquema
//Monta um esquema para criar o vetor completo da chave.
  Declare TamMsg, j, k, n: inteiro
Início
  LimpaTela
  EscrevaLn("")
  Se(Opcao="2") Então //cifrar
    TamMsg ← TamMsgOrig
    EscrevaLn("Processo de cifragem de mensagem pelo método 'Cifra de Vigenère'")
  Senão //decifrar
    TamMsg ← TamMsgCif
    EscrevaLn("Processo de decifragem de mensagem pelo método 'Cifra de Vigenère'")
  FimSe
  EscrevaLn("-----")
  EscrevaLn("")
  Se(Opcao="2") Então
    EscrevaLn("Original: ")
  Senão
    EscrevaLn("Cifrada: ")
  FimSe
  Se(Opcao="2") Então
    Para j De 1 Até TamMsg Faça
      Escreva(VetMsgOrig[j], " ")
    FimPara

```

```

    Senão
        Para j De 1 Até TamMsg Faça
            Escreva(VetMsgCif[j], " ")
        FimPara
    FimSe
    EscrevaLn("")
    EscrevaLn("")
    k ← 0
    n ← 0
    EscrevaLn("Chave: ")
    Para j De 1 Até TamMsg Faça //loop para varrer toda a mensagem
        //VetChaveTot[] é um vetor da chave completa para cifrar/decifrar
        n ← n + 1
        Se (j<=TamChave) Então //mensagem ainda do tamanho da chave
            Escreva(VetChave[j], " ") //escreve letra da chave
            VetChaveTot[n] ← VetChaveTot[n] + VetChave[j]
        Senão //posição da mensagem é superior ao tamanho da chave
            k ← k + 1 //recomeça com novo elemento da chave
            Se (k<=TamChave) Então //chave completa na mensagem, mas ainda
                TamMsg>TamChave
                Escreva(VetChave[k], " ")
                VetChaveTot[n] ← VetChaveTot[n] + VetChave[k]
        Senão //esgotou as letras da chave e ainda TamMsg>TamChave
            k ← 1
            Escreva(VetChave[k], " ") //recomeça a escrever caracteres da chave
            VetChaveTot[n] ← VetChaveTot[n] + VetChave[k]
        Fimse
    FimSe
    FimPara //fim do loop de varredura da mensagem
    EscrevaLn("")
    EscrevaLn("")
FimProcedimento //fim do procedimento "ProMontaEsquema"
//-----
Função FunPegaNumLetra(Letra:caractere): inteiro
//Obtém o número da letra da mensagem em função da tabela VI.1
    Declare NumLetra: inteiro
Início
    Selecione Letra
        Caso "A"
            NumLetra ← 0
        Caso "B"
            NumLetra ← 1
        Caso "C"
            NumLetra ← 2
        Caso "D"
            NumLetra ← 3
        Caso "E"
            NumLetra ← 4
        Caso "F"
            NumLetra ← 5
        Caso "G"
            NumLetra ← 6
        Caso "H"
            NumLetra ← 7
        Caso "I"
            NumLetra ← 8
        Caso "J"

```

```

    NumLetra ← 9
Caso "K"
    NumLetra ← 10
Caso "L"
    NumLetra ← 11
Caso "M"
    NumLetra ← 12
Caso "N"
    NumLetra ← 13
Caso "O"
    NumLetra ← 14
Caso "P"
    NumLetra ← 15
Caso "Q"
    NumLetra ← 16
Caso "R"
    NumLetra ← 17
Caso "S"
    NumLetra ← 18
Caso "T"
    NumLetra ← 19
Caso "U"
    NumLetra ← 20
Caso "V"
    NumLetra ← 21
Caso "W"
    NumLetra ← 22
Caso "X"
    NumLetra ← 23
Caso "Y"
    NumLetra ← 24
Caso "Z"
    NumLetra ← 25
FimSelecione
Retorne NumLetra
FimFunção //fim da função "FunPegaNumLetra"
//-----
Procedimento ProCifraMsgOrig
//Cifra a mensagem original baseando na fórmula:  $C = (P + K) \bmod 26$ .
// C: caractere do texto cifrado
// P: caractere do texto original
// K: caractere da chave
//-----
Declare j, NumLetraOrig, NumLetraChv, NumLetraCif: inteiro
    LetraCif: caractere
Início
    EscrevaLn("Cifragem: ")
    Para j De 1 Até TamMsgOrig Faça
        NumLetraOrig ← FunPegaNumLetra(VetMsgOrig[j])
        NumLetraChv ← FunPegaNumLetra(VetChaveTot[j])
        NumLetraCif ← NumLetraOrig + NumLetraChv
        NumLetraCif ← NumLetraCif Mod 26 //obtem número da letra da tabela
        NumLetraCif ← NumLetraCif + 65 //obtem código ASCII da letra
        LetraCif ← Carac(NumLetraCif) //obtem a letra cifrada
        Escreva(LetraCif, " ")
    FimPara
    EscrevaLn("")
FimProcedimento //fim do procedimento "ProCifraMsgOrig"
//-----

```

#### Procedimento ProDeCifraMsgCif

```
//Decifra a mensagem original baseando na fórmula:  $P=(C-K) \bmod 26$ .  
// P: caractere do texto original  
// C: caractere do texto cifrado  
// K: caractere da chave
```

```
//-----
```

```
Declare j, NumLetraCif, NumLetraChv, NumLetraDecif: inteiro  
LetraDecif: caractere
```

##### Início

```
EscrevaLn("Decifragem: ")
```

```
Para j De 1 Até TamMsgCif Faça
```

```
NumLetraChv ← FunPegaNumLetra(VetChaveTot[j])
```

```
NumLetraCif ← FunPegaNumLetra(VetMsgCif[j])
```

```
//Verifica exceção na subtração
```

```
Se (NumLetraCif < NumLetraChv) Então
```

```
NumLetraDecif ← (NumLetraCif + 26) - NumLetraChv
```

```
Senão
```

```
NumLetraDecif ← (NumLetraCif - NumLetraChv)
```

```
FimSe //fim da verificação de exceção na subtração
```

```
NumLetraDecif ← NumLetraDecif Mod 26 //obtem número da letra da tabela
```

```
NumLetraDecif ← NumLetraDecif + 65 //obtem código ASCII da letra
```

```
LetraDecif ← Carac(NumLetraDecif) //obtem a letra decifrada
```

```
Escreva(LetraDecif, " ")
```

```
FimPara
```

```
EscrevaLn("")
```

```
FimProcedimento //fim do procedimento "ProDeCifraMsgCif"
```

```
//-----
```

#### Procedimento ProMontaMenu

```
//Mota o menu de opções.
```

##### Início

```
LimpaTela
```

```
EscrevaLn("")
```

```
EscrevaLn("===== Menu Principal =====")
```

```
EscrevaLn("Apenas exibir a Grelha de Vigenère.....[1]")
```

```
EscrevaLn("Cifrar uma mensagem.....[2]")
```

```
EscrevaLn("Decifrar uma mensagem.....[3]")
```

```
EscrevaLn("Fechar o programa.....[4]")
```

```
EscrevaLn("-----")
```

```
EscrevaLn("")
```

```
FimProcedimento //fim do procedimento "ProMontaMenu"
```

```

//=====
//Programa principal
Início
//Entrada dos dados básicos do programa e montagem do menu de opções
Opcao ← ""
Enquanto (Opcao<>"4") Faça //mantém o menu de opções na tela
    ProMontaMenu //chama procedimento para montar o menu de opções
    //Limpa todos os vetores
    Para x De 1 Até 100 Faça
        VetMsgCif[x] ← ""
        VetChave[x] ← ""
        VetChaveTot[x] ← ""
    FimPara
    Opcao ← ""
    Escreva("Digite sua opção [1 a 4]: ")
    Leia(Opcao)
    Se(Opcao="4") Então
        Abandone //sai do loop incondicionalmente
    FimSe
    EscrevaLn("")
    Selecione Opcao
    Caso "1"
        ProGeraGrelha //chama procedimento para gerar a "Grelha de Vigenère"
    Caso "2"
        TamMsgOrig ← 101
        //Cria/valida a mensagem original
        Msg ← "Msg"
        Enquanto (Nao(FunVerifTexto(MensagemOrig,Msg,TamMsgOrig))) Faça
            Escreva("Mensagem original [apenas letras e máximo 100]: ")
            Leia(MensagemOrig)
            TamMsgOrig ← Compr(MensagemOrig)
            MensagemOrig ← Maiusc(MensagemOrig)
        FimEnquanto //fim de validação da mensagem original
        Para x De 1 Até TamMsgOrig Faça
            VetMsgOrig[x] ← Copia(MensagemOrig,x,1)
        FimPara
        //Cria/valida a chave de cifragem
        TamChave ← 101
        Msg ← "Chv"
        Enquanto (Nao(FunVerifTexto(Chave,Msg,TamChave))) Faça
            Escreva("Chave de cifragem [tamanho máximo da mensagem]: ")
            Leia(Chave)
            Chave ← Maiusc(Chave)
            TamChave ← Compr(Chave)
        FimEnquanto //fim de validação da chave de cifragem
        EscrevaLn("")
        //Loop para montar o vetor inicial da chave de cifragem
        Para x De 1 Até TamChave Faça
            VetChave[x] ← Copia(Chave,x,1)
        FimPara
        LimpaTela
        ProMontaEsquema //chama procedimento para montar o esquema
        ProCifraMsgOrig //chama procedimento para cifrar mensagem
    Caso "3"
        TamMsgCif ← 101
        Msg ← "Msg"

```

```

//Cria/valida a mensagem cifrada
Enquanto (Nao(FunVerifTexto(MensagemCif,Msg,TamMsgCif))) Faça
    Escreva("Mensagem cifrada [apenas letras e máximo 100]: ")
    Leia(MensagemCif)
    TamMsgCif ← Compr(MensagemCif)
    MensagemCif ← Maiusc(MensagemCif)
FimEnquanto //fim de validação da mensagem cifrada
Para x De 1 Até TamMsgCif Faça
    VetMsgCif[x] ← Copia(MensagemCif,x,1)
FimPara
//Cria/valida a chave de decifragem
TamChave ← 101
Msg ← "Chv"
Enquanto (Nao(FunVerifTexto(Chave,Msg,TamChave))) Faça
    Escreva("Chave de decifragem [tamanho máximo da mensagem]: ")
    Leia(Chave)
    Chave ← Maiusc(Chave)
    TamChave ← Compr(Chave)
FimEnquanto //fim de validação da chave de decifragem
EscrevaLn("")
//Loop para montar o vetor inicial da chave de decifragem
Para x De 1 Até TamChave Faça
    VetChave[x] ← Copia(Chave,x,1)
FimPara
LimpaTela
ProMontaEsquema //chama procedimento para montar o esquema
ProDeCifraMsgCif //chama procedimento para decifrar mensagem
CasoContário
    //Não faz nada
FimSelecione
Se(Opcao="1") Então
    Tracos ← 79
Senão
    Tracos ← 66
Fimse
Para x De 1 Até Tracos Faça
    Escreva(" ")
FimPara
EscrevaLn("")
EscrevaLn("")
EscrevaLn("")
Escreva("Tecle Enter para continuar ")
Leia(Car) //apenas para interromper temporariamente o processamento
FimEnquanto //fim da manutenção do menu de opções
LimpaTela
FimPrograma //fim do programa "ProgCifraVigenere"

```