

Constantes MD5

Mário Leite

...

Para encerrar a série de assuntos sobre “Criptografia” vistos nas partes anteriores, vou falar sobre o Método MD5 (**M**essage-**D**igest algorithm **5**) que, embora não é, propriamente dito, um método criptográfico, de acordo com a Enciclopédia digital Wikipédia é definido como *“uma função de dispersão criptográfica (função hash criptográfica) de 128 bits unidirecional desenvolvido pela RSA Data Security, Inc., descrito na RFC 1321, e muito utilizado por softwares com protocolo ponto-a-ponto (P2P, ou Peer-to-Peer, em inglês) na verificação de integridade de arquivos e logins”*.

Existem muitas literaturas sobre o este assunto, e sempre nos deparamos com alguns valores do tipo **0xD76AA478**, **0xE8C7B756**, **0x242070DB**, **0xC1BDCEEE**, etc. São as tais constantes do método que parecem coisas misteriosas e nem sempre com explicações didáticas e plausíveis. Em todos os *sites* em que pesquisei sempre aparecem esses tais números onde o autor da publicação diz que são as “constantes”, nas quais o método é baseado. Alguns deles até mostram esses valores em blocos, formando uma matriz 16x4, totalizando 64 itens, que são usados na Função Hash. Outros indicam que em Pascal pode ser utilizada a biblioteca MD5 com a instrução “**Uses MD5**” para fazer a criptografia. Isto Seria muito simples se funcionasse em qualquer versão do Pascal; tentei no Dev-Pascal e no Pascal Zim, mas só funcionou no Free Pascal! E mesmo funcionando, a origem das constantes continua bem misteriosa para o programador; principalmente para os iniciantes.

Então, foquei na seguinte questão: “COMO SÃO OBTIDAS AS TAIS CONSTANTES QUE NORTEIAM ESSE ALGORITMO”? Em um desses *sites* - o mais esclarecedor que encontrei - obtive uma dica: *“as constantes são diretamente proporcionais ao **seno(x)**, onde **x** tem o menor valor 1*. E embora não mostrasse como fazer isto, foi uma dica preciosa!

Observei que os valores das constantes são SEMPRE representados em blocos de 8 *bits* (excluindo **0x**), dando um total de **32 bits** em cada linha; o que em alguns *sites* são chamados de PALAVRA. Então, desconfiei que esse 32 era a “chave” que eu precisava.

Primeiramente, segui o óbvio: TUDO em Informática está baseado no sistema binário, sendo os valores potências de **2**. Então, tentei a expressão **2³²** e encontrei 4294967296; e seguindo a dica do fator seno(x): **C = 2³²*seno(x)** e considerando **x=1** para a primeira constante fiz **C1 = 2³²*Seno(1) = 4294967296*0.841470984 = 3614090360**. Mas, como os valores das constantes estão na base hexadecimal, tive que fazer a conversão, o que não foi difícil; e obtive o valor **D76AA478** para a primeira constante, confirmando o que o *site* informava. Então, fazendo **x=2**, **x=3** e **x=4**, (prossequindo com a *segunda*, *terceira* e *quarta* constante) cheguei aos primeiros quatro valores hexadecimais. Deste modo, foi fácil obter todos os 64 valores das constantes do MD5 fazendo um *loop* com x variando de 1 até 64, e dentro *desse loop* as devidas conversões. Um pouco trabalhoso, mas, relativamente fácil; pois, nada que uma boa dose de Lógica e perspicácia não resolva!

Seria bem interessante que esses *sites* mostrassem isto, sem fazer todo aquele mistério a respeito dessas constantes, apavorando os iniciantes em programação!

Mas, embora o MD5 não seja um método criptográfico, e seja bem vulnerável, ele serve para codificar mensagens” não muito secretas”.

Algoritmo "GeraConstantestMD5"

//Gera as constantes do Método MD5.

//Em Visualg

//Autor: Mário Leite

//-----

Var D, R: **vetor**[1..64] **de inteiro**

NX: **vetor**[1..64] **de caractere**

i, j, k, B, Q, T, x, Col, Posi, Dig, DSe: **inteiro**

N: **real**

Aux, BS, NB, NS, Rest: **caractere**

Resp, Acabou: **logico**

Inicio

LimpaTela

Escreval (" Constantes hexadecimais do MD5:")

Escreval ("-----")

B <- 16 //base Hexadecimal

BS <- **NumpCarac**(B)

Col <- 0

Para x **De** 1 **Ate** 64 **Faca**

N <- (2³²)***Abs**(**Sen**(x)) //cria uma constante MD5 decimal

NS <- **NumpCarac**(N)

{Elimina as decimais}

Posi <- **Pos**(".",NS) //pega a posição do ponto (.) em NS

Aux <- ""

Para k **De** 1 **Ate** (Posi-1) **Faca**

Aux <- Aux + **Copia**(NS,k,1) //cria novo NS sem o ponto

FimPara

NS <- Aux

T <- **Compr**(NS)

Resp <- **Verdadeiro**

//Loop para verificar os dígitos do número digitado

Para j **De** 1 **Ate** T **Faca**

NX[j] <- **Copia**(NS,j,1)

Dig <- **Asc**(NX[j])

Se((Dig<48) **ou** (Dig>57)) **Entao**

Resp <- **Falso**

FimSe

FimPara

{É preciso fazer as divisões sucessivas}

j <- 1

Acabou <- **Falso**

//Loop para fazer as divisões sucessivas pela base 16

Enquanto (**Nao**(Acabou)) **Faca**

D[j] <- **Int**((N/B))

Q <- D[j]

R[j] <- (N **Mod** B)

Se(D[j]<B) **Entao**

j <- j + 1

R[j] <- Q

Acabou <- **Verdadeiro** //não precisa mais dividir (Dividendo<Base)

Interrompa //sai do loop incondicionalmente

Senao

N <- D[j]

j <- j + 1

FimSe

FimEnquanto //fim do loop das divisões sucessivas

{Define o número na base 16}

NB <- ""

Para i **De** j **Ate** 1 **Passo** -1 **Faca**

Se(R[i]<0) **Entao** //verifica exceção com o resto

R[i] <- R[i] + 16

FimSe

Escolha R[i]

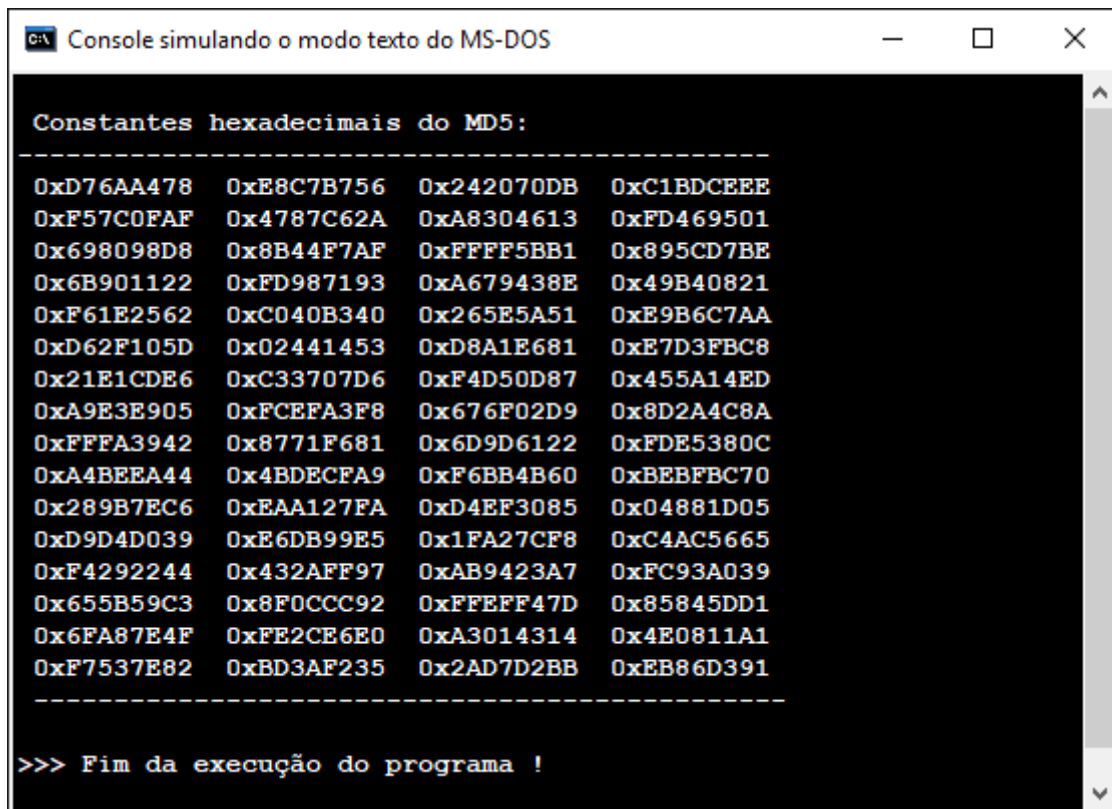
Caso 10

```

        Rest <- "A"
    Caso 11
        Rest <- "B"
    Caso 12
        Rest <- "C"
    Caso 13
        Rest <- "D"
    Caso 14
        Rest <- "E"
    Caso 15
        Rest <- "F"
    OutroCaso
        Rest <- NumpCarac(Abs(R[i]))
    FimEscolha
    NB <- NB + Rest
FimPara
Col <- Col + 1
Se(Col>4) Entao
    Escreval("")
    Col <- 1
FimSe
Se(Compr(NB)<8) Entao
    Escreva(" 0x0",NB, " ") //imprime com um zero à esquerda
Senao
    Escreva(" 0x",NB, " ") //imprime normalmente
FimSe
FimPara
Escreval("-----")
FimAlgoritmo //fim do programa "GeraConstantestestMD5"

```

Figura 1 - Saída do programa “GeraConstantesMD5” em Visualg



```

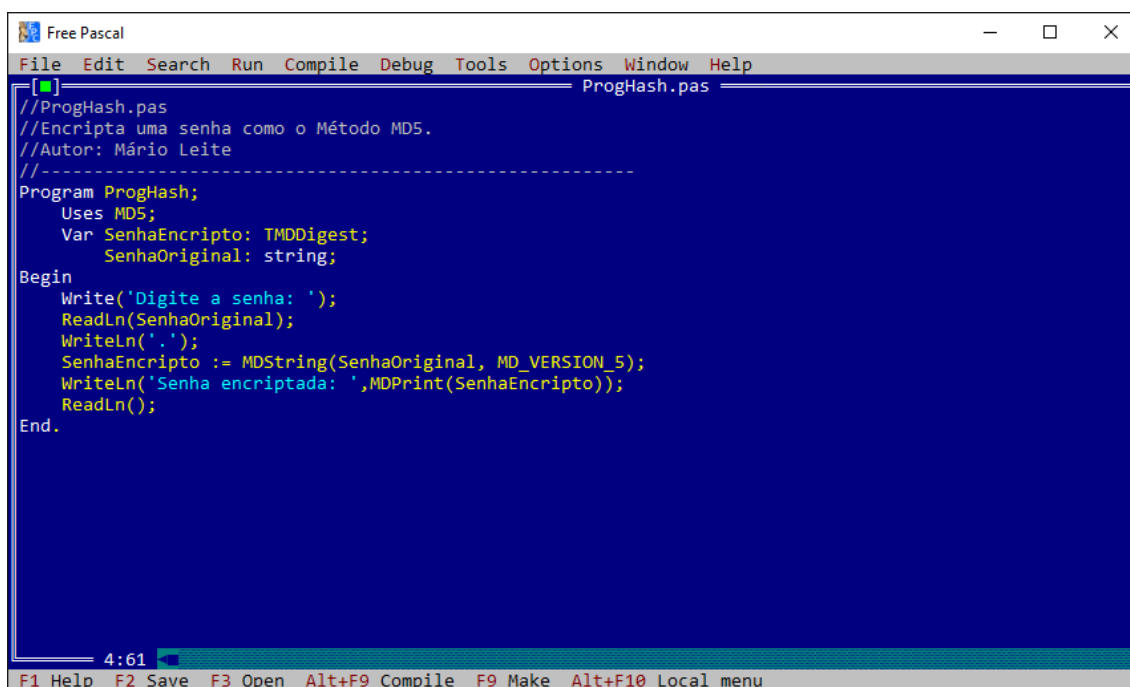
C:\> Console simulando o modo texto do MS-DOS

Constantes hexadecimais do MD5:
-----
0xD76AA478  0xE8C7B756  0x242070DB  0xC1BDCEEE
0xF57C0FAF  0x4787C62A  0xA8304613  0xFD469501
0x698098D8  0x8B44F7AF  0xFFFF5BB1  0x895CD7BE
0x6B901122  0xFD987193  0xA679438E  0x49B40821
0xF61E2562  0xC040B340  0x265E5A51  0xE9B6C7AA
0xD62F105D  0x02441453  0xD8A1E681  0xE7D3FBC8
0x21E1CDE6  0xC33707D6  0xF4D50D87  0x455A14ED
0xA9E3E905  0xFCEFA3F8  0x676F02D9  0x8D2A4C8A
0xFFFA3942  0x8771F681  0x6D9D6122  0xFDE5380C
0xA4BEEA44  0x4BDECFA9  0xF6BB4B60  0xBEBFBC70
0x289B7EC6  0xEAA127FA  0xD4EF3085  0x04881D05
0xD9D4D039  0xE6DB99E5  0x1FA27CF8  0xC4AC5665
0xF4292244  0x432AFF97  0xAB9423A7  0xFC93A039
0x655B59C3  0x8F0CCC92  0xFFEFF47D  0x85845DD1
0x6FA87E4F  0xFE2CE6E0  0xA3014314  0x4E0811A1
0xF7537E82  0xBD3AF235  0x2AD7D2BB  0xEB86D391
-----

>>> Fim da execução do programa !

```

Figura 2 - Saída do programa “GeraConstantesMD5” em Visualg



```

Free Pascal
File Edit Search Run Compile Debug Tools Options Window Help
ProgHash.pas
//ProgHash.pas
//Encripta uma senha como o Método MD5.
//Autor: Mário Leite
//-----
Program ProgHash;
  Uses MD5;
  Var SenhaEncripto: TMDDigest;
      SenhaOriginal: string;
Begin
  Write('Digite a senha: ');
  ReadLn(SenhaOriginal);
  WriteLn('.');
  SenhaEncripto := MDString(SenhaOriginal, MD_VERSION_5);
  WriteLn('Senha encriptada: ',MDPrint(SenhaEncripto));
  ReadLn();
End.
4:61
F1 Help F2 Save F3 Open Alt+F9 Compile F9 Make Alt+F10 Local menu

```

Figura 3 - Código do programa “ProgHash” em Free Pascal

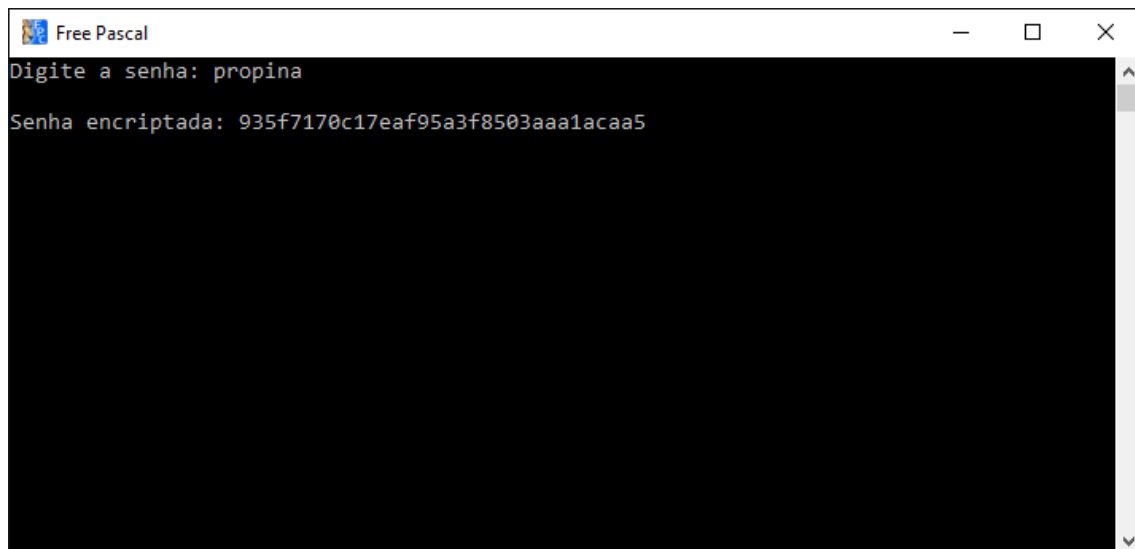


Figura 4 - Saída do programa “ProgHash” em Free Pascal

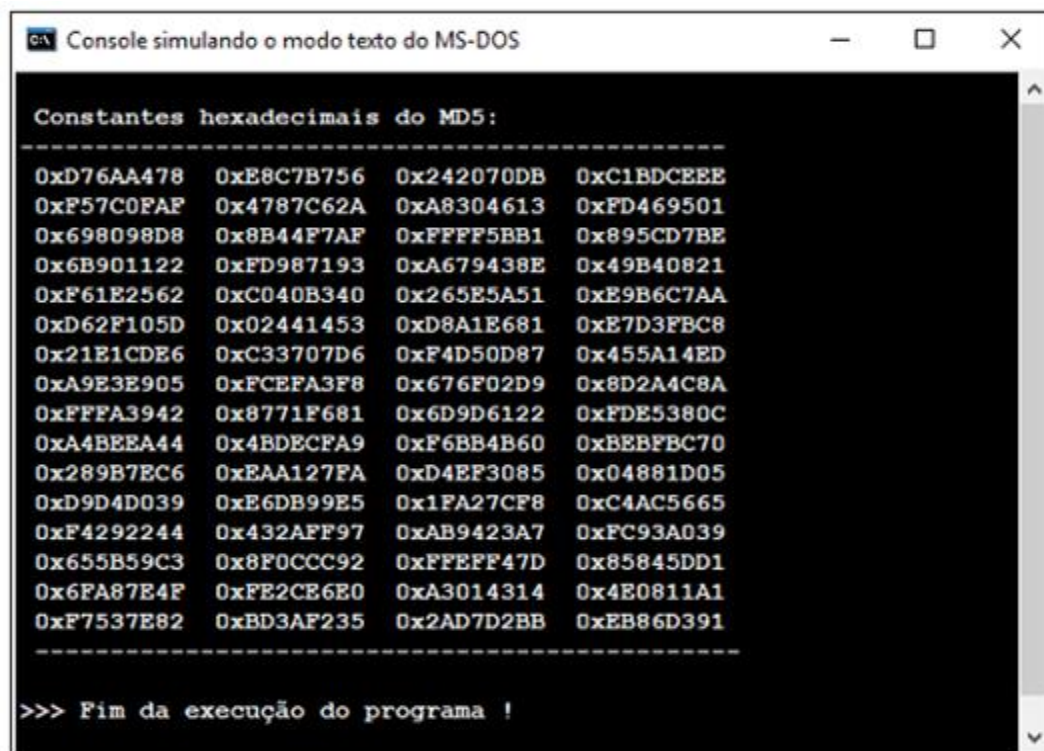


Figura 2 - Saída do programa “GeraConstantesMD5” em Visualg

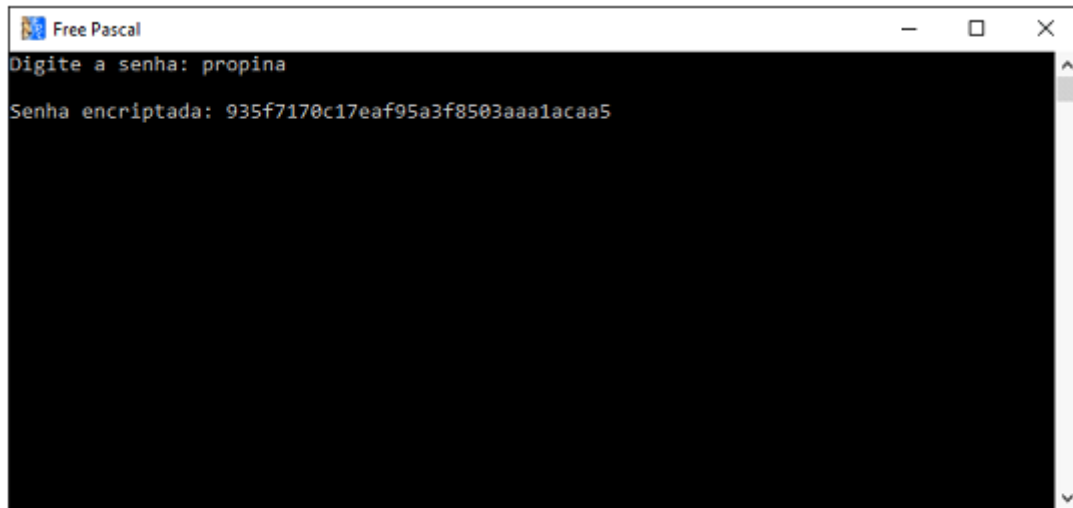


Figura 4 - Saída do programa "ProgHash" em Free Pascal

Nota: Os códigos-fonte completos (em pseudocódigo) de vários programas de Criptografia estão no meu livro ***"1001 Programas Prontos Para Você Codificar Na Sua Linguagem Preferida"***, e pode ser adquirido na Livraria Amazon, em *e-book*, ou diretamente comigo pelo *e-mail*: marleite@gmail.com
