Raízes Primitivas

Mário Leite

...

As Raízes Primitivas são importantes em Criptografia e na Teoria dos Números; e de acordo com a Enciclopédia Wikipédia: "*um número **g** é raiz primitiva módulo **n** se todo número **a**, coprimo a **n**, for congruente com uma potência de **g** módulo **n**. Ou seja, **g** é uma raiz primitiva **n** se para cada inteiro **a**, coprimo com **n**, existe um inteiro k, tal que* $g^k \equiv a \pmod{n}$*. Esse valor **k** é chamado índice ou logaritmo discreto de **a** para a base **g** módulo **n**. Observe que **g** é uma raiz primitiva de módulo **n** se, e somente se, **g** é um gerador de inteiros módulo **n***". Eu achei esta definição muito complexa e nada esclarecedora. Vamos tentar esclarecer...

Considerando um certo número **g**, ele será raiz primitiva para um determinado número primo **p** se o resto da divisão de **g** por **p** (g *mod* p) possuir um módulo de ordem **p-1**. Isto significa que a lista de (g^1 *mod* p), (g^2 *mod* p), (g^3 *mod* p), ... até g^ *mod* (p-1) *mod* p conterá todos os inteiros de **1** até **(p-1)**.

Até o momento não existe um algoritmo conhecido para calcular raízes primitivas com uma eficiência cem por cento, mas o algoritmo apresentado abaixo é uma boa técnica que pode ser utilizada.

Passo1: Escolha o número primo p=5, por exemplo.

Passo 2: Calcule (2^n *mod* p) para n de a p-1 para produzir a lista.

Passo 3: Verifique se a lista contém todos os restos de 5. A lista estará qualificada se tivermos 2,4,3,1; então 2 seria uma raiz primitiva. Mas, se for 2,1,4,1 (lista para 4), 4 não seria uma raiz primitiva, porque estaria faltando o 3, e assim por diante.

Passo 4: Repita o passo anterior para todos os números menores que 5.

O programa "**RaizesPrimitivas**", testado com o Visualg, mostra as **x** (min 2 e max 50) primeiras raízes primitivas. O programa é modular, contendo três funções:

FunPotModular: Calcula a expressão modular entre dois números.

FunMDCRec: Função recursiva para calcular o MDC de dois números, sendo um deles uma expressão modular.

FunEuler: Verifica a coprimaridade de um número com **n**.

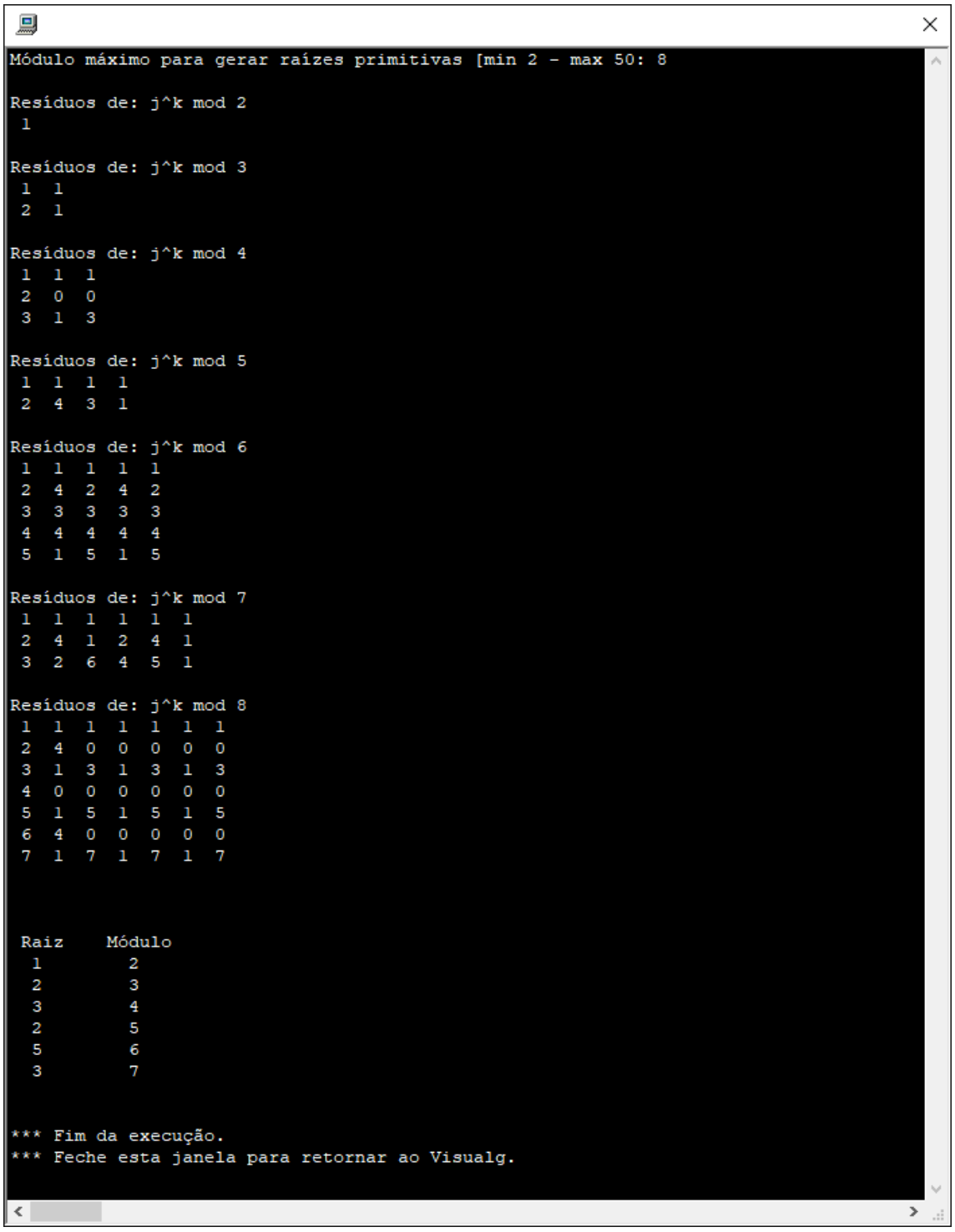

Figura 1 - Saída do programa para as primeiras cinco raízes primitivas

```
Algoritmo "RaizesPrimitivas"
//Gera as x primeiras ocorrências das primeiras raízes primitivas módulo m.
//Autor  : Mário Leite
//E-mail : marleite@gmail.com
//-------------------------------------------------------------------------------
   //Elementos globais
   Const MAXMOD=50  //limita o módulo m
   Var MatRes: vetor[1..100,1..100] de inteiro
       VetRes, VetResX, VetNum, VetMod: vetor[1..100] de inteiro
       j, k, m, n, p, q, r, x, y, z: inteiro
       Aux, MDC, Tam, FiNumMod, Ordem: inteiro
       Pote: real
//-------------------------------------------------------------------------------
  Funcao FunPotModular(B,x,n:inteiro): inteiro
  //Retorna o resultado do cálculo modular: (B^x Mod n).
     var k, Ret: inteiro
  Inicio
     Ret <- 1
     Para k De 1 Ate x Faca
        Ret <-(Ret*B) Mod n
     FimPara
     Retorne Ret
  FimFuncao  //fim da função "FunPotModular"
  //-------------------------------------------------------------------------------
  Funcao FunMDCRec(N1,N2:inteiro): inteiro
  //Retorna o MDC de de um número e seu cálculo modular
  Inicio
    Se(N2=0) Entao
       Retorne N1
    Senao
       Retorne FunMDCRec(N2, N1 Mod N2)
    FimSe
  FimFuncao  //fim da função "FunMDCRec"
//-------------------------------------------------------------------------------
  Funcao FunEuler(d:inteiro): inteiro
  //Verifica se o número é coprimo com d.
     var k, Cont, Aux, MDC: inteiro
  Inicio
     Cont <- 0
     Para k De 1 Ate (d-1) Faca
        MDC <- FunMDCRec(d,k) //chama função para calcular o MDC(d,k)
        Se(MDC=1) Entao
           Cont <- Cont + 1
        FimSe
     FimPara
     Retorne Cont  //retorna a quantidade de coprimos com d, menores que d
  FimFuncao  //fim da função "FunEuler"
//===============================================================================
//Programa principal
Inicio
   Repita
      Escreva("Digite a quantidade de raízes primitivas [min 2-max",MAXMOD,":")
      Leia(x)
      x <- Int(x)
   Ate((x>=2) e (x<=MAXMOD))
   Escreval("")
   n <- 0  //índice dos vetor de módulos e dos números
   m <- 2
   Enquanto (m<=x) Faca
      FiNumMod <- FunEuler(m) //verifica a multiplicidade de m
      Escreval("Resíduos de: j^k mod",m)
      Para j De 1 Ate 7 Faca  //loop das raízes
         Para k De 1 Ate (m-1) Faca  //loop dos expoentes de j
            Pote <- FunPotModular(j,k,m)
            MatRes[j,k] <- Int(Pote)
            Escreva(MatRes[j,k]," ")
            VetRes[k] <- MatRes[j,k]  //cria os elementos do vetor de resíduos
         FimPara  //fim do loop das raízes de geração dos resíduos
         {Aqui já está formado vetor de resíduos do cálculo modular: j^k mod m}
         MDC <- FunMDCRec(j,m) //calcula o MDC(j,m)
```

```
71          Se(MDC=1) Entao  //verifica se j e m são primos entre si
72             {Ordena os elementos do vetor de resíduos}
73             Para y De 1 Ate (m-2) Faca
74                Para z De (y+1) Ate (m-1) Faca
75                   Se(VetRes[y]>VetRes[z]) Entao
76                      Aux <- VetRes[y]
77                      VetRes[y] <- VetRes[z]
78                      VetRes[z] <- Aux
79                   FimSe
80                FimPara
81             FimPara //fim do loop de ordenação
82             {Retira elementos repetidos: calcula a Ordem do módulo}
83             p <- 0  //contador de elementos do vetor sem repetições
84             q <- 1  //elemento anterior
85             r <- q + 1  //elemento posterior
86             Tam <- m-1
87             Enquanto (q<=Tam) Faca
88                Se(VetRes[r]<>VetRes[q]) Entao
89                   p <- p + 1
90                   VetResX[p] <- VetRes[q]  //vetor sem elementos repetidos
91                FimSe
92                q <- q + 1
93                r <- r + 1
94             FimEnquanto
95             Ordem <- p  //tamanho real do vetor de resíduos [sem repetições]
96          FimSe
97          {Testa a multiplicidade}
98          Se(FiNumMod=Ordem) Entao
99             n <- n + 1
100            VetNum[n] <- j
101            VetMod[n] <- m
102            Interrompa  //sai do loop j (já encontrou uma Raiz Primitiva)
103         FimSe
104         Escreval("")
105      FimPara  //fim do loop j
106      Escreval("")
107      Escreval("")
108      m <- m + 1  //pega um novo módulo
109   FimEnquanto  //fim do loop geral dos módulos
110   Escreval("")
111   {Exibe as raízes primitivas com seus respectivos módulos}
112   Escreval(" Raiz    Módulo")
113   Para j De 1 Ate n Faca
114      Escreval(" ",VetNum[j],"       ",VetMod[j])
115   FimPara
116   Escreval("")
117FimAlgoritmo  //fim do programa "ProgGeraRaizesPrimitivas"
```