Capicua: Um Número Constrangido

Mário Leite

...

Número Capicua (ou Número Palíndromo) é um número inteiro e positivo que, lendo de frente para trás é exatamente igual lendo de trás para frente; por exemplo, os anos 2002 e 2112 são deste tipo. Neste caso temos uma situação que "constrange" o número; tecnicamente falando é uma situação de escrita constrangida que na verdade define até uma técnica literária chamada de "Escrita Constrangida", que certos autores utilizam em suas poesias. No caso da Matemática esse "constrangimento" se resume a manter o numero em vice-versa.

É fácil identificar um número capicua de poucos dígitos, mas, para números muito grandes a tarefa requer uma computação intensa. Por exemplo, **1234567891357924688642975319876543 21** é um número capicua. Notem que até para montar esse número é uma tarefa penosa; imagine para verificar se ele é capicua; a gente se perde no meio da análise! Para ilustrar este assunto, escrevi o programa modular "GeraNumeroCapicua e o testei em Visualg (código-fonte abaixo) para encontrar números capicuas dentro de um intervalo definido pelo usuário. Além do módulo principal este programa possui duas rotinas:

VerifCapicua(): função que recebe uma string de número e verifica se é capicua.
CriaVetorInvertido: procedimento que cria um vetor de elementos invertidos.

```
Algoritmo "GeraNumerosCapicuas"
//Gera os números capicuas entre dois limites (inclusive) definidos pelo usuário.
//Em Visualg
//Autor : Mário Leite
//--------------------------------------------------------------------------------
   //Elementos globais
   Const MINLim=10  //limite inferior do gerador
         MAXLIM=1000 //limite superior do gerador
   Var VetNum, VetCapicuia, VetNumOrig: vetor[1..MAXLIM] de real
       VetNumS: vetor[1..MAXLIM] de caractere
       p, Cap, Lim1, Lim2: inteiro
       NumOrig, NumLido: real
       Excedeu: logico
//--------------------------------------------------------------------------------
   Funcao VerifCapicua(SomaOrig:caractere): logico
   //Verifica se um número é Capicua
      var m, k: inteiro
          SomaR: real
          CarIF, CarFI, SomaInv: caractere
          EhPalindro: logico
   Inicio
      EhPalindro <- Verdadeiro
      m <- Compr(SomaOrig)
      {Cria a soma inversa}
      SomaInv <- ""
      Para k De 1 Ate m Faca
         SomaInv <- SomaInv + Copia(SomaOrig,(m+1-k),1)
      FimPara
      {Verificaa se SomaOrig e SomaInv são palíndromos}
      Se(SomaOrig<>SomaInv) Entao  //não é um palíndromo
         EhPalindro <- Falso
      FimSe
      Retorne EhPalindro
   FimFuncao  //fim da função "VerifCapicua"
//--------------------------------------------------------------------------------
```

```
Procedimento CriaVetorInvertido
//Cria um vetor com os dígitos do número digitado
   var VetInv: vetor[1..MAXLIM] de real
       i, j, Tam: inteiro
       Soma, NumInv: real
       Aux, NumS, SomaS: caractere
       Resp: logico
Inicio
   Cap <- 0
   Para i De Lim1 Ate Lim2 Faca //loop para varrer a faixa de geradores
      NumOrig <- i
      NumLido <- NumOrig  //preserva o número inicialmente lido
      NumS <- NumpCarac(NumLido)
      Tam  <-  Compr(NumS)
      Para j De 1 Ate Tam Faca
         VetNum[j] <- CaracpNum(Copia(NumS,j,1))
         VetNumS[j] <- NumpCarac(VetNum[j])
      FimPara
      Resp <- Falso
      Enquanto (Resp=Falso) Faca
         {Inverte os elementos do vetor lido}
         Aux <- ""
         Soma <- 0
         Para j De Tam Ate 1 Passo -1 Faca
            VetInv[j] <- VetNum[j]
            Aux <- Aux + NumpCarac(VetInv[j])
         FimPara
         {Soma o número atual com o seu valor invertido}
         NumInv <- CaracpNum(Aux)
         Soma <- NumOrig + NumInv
         SomaS <- NumpCarac(Soma)
         Resp <- VerifNumCapicua(NumS)
         Se(Resp) Entao
            Resp <- Verdadeiro  //o número é capicua
            Interrompa //abandona o loop incondicionalmente
         Senao
            Excedeu <- Falso //número não é capicua
            Aux <- ""
            Para j De 1 Ate Compr(SomaS) Faca
               Aux <- Aux + Copia(SomaS,(3+1-j),1)
            FimPara
            Resp <- Falso
            NumOrig <- Soma + CaracpNum(Aux)
            Tam <- Compr(NumpCarac(NumOrig)) //pega o novo tamanho de número
            Se((Tam>15) ou (NumOrig>2147483646)) Entao //evita overflow
               Excedeu <- Verdadeiro //algo anormal aconteceu
               Interrompa
            FimSe
            NumS <- NumpCarac(NumOrig)
           {Cria um vetor com os dígitos do novo número definido pela soma}
            Para j De 1 Ate Tam Faca
               VetNum[j]  <- CaracpNum(Copia(NumS,j,1))
               VetNumS[j] <- NumpCarac(VetNum[j])
            FimPara
         FimSe
      FimEnquanto
      Se(Nao(Excedeu)) Entao //número capicua gerado com sucesso
        Cap <- Cap + 1
         VetNumOrig[Cap]  <- NumLido
         VetCapicuia[Cap] <- NumOrig
      FimSe
   FimPara  //fim do loop de varredura da faixa de geradores
FimProcedimento  //fim do procedimento "ProCriaVetorInvertido"
```

```
//================================================================================
//Programa principal
Inicio
   Repita
      Escreva("Limite inferior da geração de capicuas [min",MINLim,"]: ")
      Leia(Lim1)
      Lim1 <- Int(Lim1)
      Escreval("")
       Escreva("Limite superior da geração de capicuas [min",(Lim1+1)," - max",MAXLIM,"]: ")
      Leia(Lim2)
      Lim2 <- Int(Lim2)
   Ate((Lim1>=10) e (Lim1<Lim2))
   CriaVetorInvertido //chama rotina para criar o vetor invertido
   LimpaTela
   {Exibe os números capicuas com seus gerados}
   Escreval("Número Gerador     Número Capicua")
   Para p De 1 Ate Cap Faca
      Escreval("     ",VetNumOrig[p],"                ",VetCapicuia[p])
   FimPara
   Escreval("")
FimAlgoritmo  //fim do programa "GeraNumeroCapicua"
```