

Cálculos con números decimales largos en un ordenador.

Aladar Peter Santha

A veces, los matemáticos necesitan trabajar con números decimales largos en sus ordenadores. Solamente con el software suministrado por el fabricante (suficiente para las necesidades corrientes), estos cálculos no se pueden llevar a cabo y, como en el caso de las operaciones con números enteros grandes, deben desarrollar sus propios programas. Esto es posible y el cálculo con decimales largos se puede reducir al cálculo con números enteros grandes [1]. Así, trabajando en el lenguaje de programación Visual-Basic, en un módulo deben figurar las funciones Sumar, Restar, Multiplicar y Potencias para las operaciones con enteros grandes.

Se sabe que para sumar dos decimales hay que escribirles uno por debajo del otro y alinear los puntos decimales. La suma se efectúa como si tratase de números enteros y luego se coloca el punto decimal a la altura donde estaba situado en los decimales. Por ejemplo, en las sumas siguientes, hay que observar la alineación de los puntos decimales

	7	9	4	.	6	5	3	8	5
+		5	8	.	7	3	4	1	5
	8	5	3	.	3	8	8	0	0

	3	5	4	.	6	5	3	8	5
+			0	.	0	3	4		
	3	5	4	.	6	8	7	8	5

, luego en el resultado hay que suprimir los ceros finales no significativos. Así,

$$794.65385 + 58.73415 = 853.388$$

$$354.65385 + 0.0034 = 354.68785$$

Así, la suma de dos decimales siempre es reducible a la suma de dos enteros, añadiendo ceros no significativos en uno de ellos (para que tengan el mismo número de cifras después del punto decimal), multiplicándoles con 10^r (para suprimir los puntos decimales y transformarles en enteros) y dividiendo la suma de los enteros con 10^r . Por ejemplo:

$$\begin{aligned} 354.65385 + 0.0034 &= 354.65385 + 0.003400 = (35465385 + 3400) : 10000 \\ &= 35468785 : 10000 = 354.68785. \end{aligned}$$

En general, en un ordenador hay que introducir los decimales en dos variables de tipo String y hay que contar en cada uno de ellos el número de las cifras después del punto decimal. Si en el primer número hay p cifras después del punto decimal y q en el segundo ($p \neq q$) entonces en el número que tiene menos cifras después del punto decimal, hay que añadir $|p - q|$ ceros finales (no significativos). Si $p = q$, no hay que añadir ceros finales en ninguno de los números. Igualadas ya el número de las cifras situadas después del punto decimal en los dos números, sea r el número de estas cifras. Luego, se procede a la supresión de los puntos decimales (que corresponde a la multiplicación con 10^r), llegando así a dos enteros (si al suprimir los puntos decimales, algún entero empieza por cifras 0, hay que suprimirlas) a los cuales podemos sumar con la función *Sumar* de los enteros grandes. Después queda colocar el punto decimal tal que en el resultado el número de las cifras situadas después del punto decimal sea r (que corresponde a la división entre 10^r). Finalmente, si hace falta, hay que suprimir los ceros finales no significativos del resultado. Si después de la supresión de ceros el resultado se termina en un punto decimal, hay que suprimirlo.

La suma de dos decimales se puede efectuar mediante el siguiente código Visual-Basic:

```

Public Function SumarDec(ByRef x0() As String, ByVal n As Integer) As String
    Dim sw As Integer, i As Long, j As Long, nd As Long
    Dim ndec(2) As Long, lrg(2) As Long, num(2) As String
    Dim at As String, res As String, te(2) As String
    Dim sg As String, caracter As String, xa(2) As String
    If x0(1) = "0" Or x0(2) = "0" Then
        If x0(2) = "0" Then
            SumarDec = x0(1): Exit Function
        End If
        If x0(1) = "0" Then
            SumarDec = x0(2): Exit Function
        End If
    End If
    lrg(1) = Len(x0(1)): lrg(2) = Len(x0(2))
    For j = 1 To 2
        xa(j) = x0(j): sw = 0: i = 1
        Do
            caracter = Left$(Right$(xa(j), i), 1)
            If caracter = "." Then
                ndec(j) = i - 1
                Exit Do
            End If
            i = i + 1
            If i = lrg(j) + 1 Then Exit Do
        Loop While caracter <> "."
        If ndec(j) <> 0 Then
            te(1) = Left$(xa(j), lrg(j) - ndec(j) - 1)
            te(2) = Right$(xa(j), ndec(j))
            num(j) = te(1) + te(2)
        Else
            num(j) = xa(j)
        End If
        If Left$(num(j), 2) = "-0" Then num(j) = Mid$(num(j), 2): sw = 1
        If Left$(num(j), 1) = "0" Then
            Do
                If Left$(num(j), 1) = "0" Then
                    num(j) = Mid$(num(j), 2)
                Else
                    Exit Do
                End If
            Loop
            If num(j) = "" Then num(j) = "0"
        End If
        If sw = 1 Then num(j) = "-" + num(j): sw = 0
    Next j
    If ndec(1) < ndec(2) Then
        For i = 1 To ndec(2) - ndec(1)
            num(1) = num(1) + "0"
        Next i
        nd = ndec(2)
    End If
    If ndec(1) > ndec(2) Then
        For i = 1 To ndec(1) - ndec(2)
            num(2) = num(2) + "0"
        Next i
        nd = ndec(1)
    End If

```

```

End If
If ndec(1) = ndec(2) Then nd = ndec(1)
at = Sumar(num(), n)
If Left$(at, 1) = "-" Then at = Mid$(at, 2): sg = "-"
If nd <> 0 Then
    cr = Len(at)
    If cr > nd Then
        te(1) = Left(at, cr - nd)
        te(2) = Right$(at, nd)
        at = te(1) + "." + te(2)
    Else
        If cr = nd Then
            at = "0." + at
        Else
            For i = 1 To nd - cr
                at = "0" + at
            Next i
            at = "0." + at
        End If
    End If
    sw = 0
    For i = 1 To Len(at)
        If Right$(Left$(at, i), 1) = "." Then sw = 1: Exit For
    Next i
    ' suprimir ceros no significativos al final del número decimal
    If sw = 1 Then
        i = Len(at)
        Do
            If Right$(Left$(at, i), 1) = "0" Then
                at = Left$(at, Len(at) - 1)
                i = i - 1
            Else
                Exit Do
            End If
        Loop
    End If
    End If
    If Right$(at, 1) = "." Then at = Left$(at, Len(at) - 1)
    res = sg + at
    SumarDec = res
End Function

```

La resta de los números decimales se efectúa de la misma manera, solamente que, en lugar de la función Sumar de los números enteros, hay que utilizar la función Restar. El código Visual-Basic en este caso es la siguiente:

```

Public Function RestarDec(ByRef x0() As String, ByVal n As Integer) As String
    Dim sw As Integer, i As Long, j As Long, nd As Long
    Dim ndec(2) As Long, lrg(2) As Long, num(2) As String
    Dim at As String, res As String, te(2) As String
    Dim sg As String, caracter As String, xa(2) As String
    If x0(1) = "0" And x0(2) = "0" Then
        RestarDec = "0": Exit Function
    End If
    If x0(2) = "0" Then
        RestarDec = x0(1): Exit Function
    End If

```

```

End If
If x0(1) = "0" Then
  If Left$(x0(2), 1) = "-" Then
    res = Mid$(x0(2), 2)
  Else
    res = "-" + x0(2)
  End If
  RestarDec = res: Exit Function
End If
lrg(1) = Len(x0(1)): lrg(2) = Len(x0(2))
For j = 1 To 2
  xa(j) = x0(j): sw = 0: i = 1
  Do
    character = Left$(Right$(xa(j), i), 1)
    If character = "." Then
      ndec(j) = i - 1
      Exit Do
    End If
    i = i + 1
    If i = lrg(j) + 1 Then Exit Do
  Loop While character <> "."
  If ndec(j) <> 0 Then
    te(1) = Left$(xa(j), lrg(j) - ndec(j) - 1)
    te(2) = Right$(xa(j), ndec(j))
    num(j) = te(1) + te(2)
  Else
    num(j) = xa(j)
  End If
  If Left$(num(j), 2) = "-0" Then num(j) = Mid$(num(j), 2): sw = 1
  If Left$(num(j), 1) = "0" Then
    Do
      If Left$(num(j), 1) = "0" Then
        num(j) = Mid$(num(j), 2)
      Else
        Exit Do
      End Do
    Loop
    If num(j) = "" Then num(j) = "0"
  End If
  If sw = 1 Then num(j) = "-" + num(j): sw = 0
Next j
If ndec(1) < ndec(2) And num(1) <> "0" Then
  For i = 1 To ndec(2) - ndec(1)
    num(1) = num(1) + "0"
  Next i
  nd = ndec(2)
End If
If ndec(1) > ndec(2) Then
  For i = 1 To ndec(1) - ndec(2)
    num(2) = num(2) + "0"
  Next i
  nd = ndec(1)
End If
If ndec(1) = ndec(2) Then nd = ndec(1)
at = Restar(num(), n)
If Left$(at, 1) = "-" Then at = Mid$(at, 2): sg = "-"
If nd <> 0 Then
  cr = Len(at)
  If cr > nd Then
    te(1) = Left$(at, cr - nd)
    te(2) = Right$(at, nd)
    at = te(1) + "." + te(2)
  End If
End If

```

```

Else
    If cr = nd Then
        at = "0." + at
    Else
        For i = 1 To nd - cr
            at = "0" + at
        Next i
        at = "0." + at
    End If
End If
sw = 0
For i = 1 To Len(at)
    If Right$(Left$(at, i), 1) = "." Then sw = 1: Exit For
Next i
' suprimir ceros no significativos al final del número decimal
If sw=1 then
    i = Len(at)
    Do
        If Right$(Left$(at, i), 1) = "0" Then
            at = Left$(at, Len(at) - 1)
            i = i - 1
        Else
            Exit Do
        End If
    Loop
End If
If Right$(at, 1) = "." Then at = Left$(at, Len(at) - 1)
res = sg + at
RestarDec = res
End Function

```

Para multiplicar dos números decimales introducidos en variables de tipo string, primero se determina la cantidad de las cifras situadas después del punto decimal en cada variable: *ndec(1)* y *ndec(2)*. Después se suprimen los puntos decimales y los ceros que pueden aparecer al principio de los números debido a la supresión de los puntos decimales. Los enteros así obtenidos se multiplicarán utilizando la función Multiplicar. Luego, se coloca el punto decimal tal que en el resultado el número de las cifras situadas después del punto decimal sea $ndec(1) + ndec(2)$. Si hay ceros no significativos al final del resultado, hay que suprimirlos. Por fin, si el último símbolo del resultado es un punto (el resultado es un entero) hay que suprimir el punto final.

```

Public Function MultiplicarDec(ByRef x0() As String, ByVal n As Integer) As String
    Dim dif As Double, i As Long, j As Long, nd As Long, sw As Integer
    Dim ndec(2) As Long, lrg(2) As Long, num(2) As String
    Dim at As String, bt As String, ct As String, te(2) As String
    Dim caracter As String, sg(2) As String, res As String, xa(2) As String
    If x0(1) = "0" Or x0(2) = "0" Then
        MultiplicarDec = "0": Exit Function
    End If
    For i = 1 To 2
        xa(i) = x0(i)
        If Left$(xa(i), 1) = "-" Then sg(i) = "-": xa(i) = Mid$(xa(i), 2)
    Next i
    lrg(1) = Len(xa(1)): lrg(2) = Len(xa(2))
    For j = 1 To 2
        i = 1
        Do
            caracter = Left$(Right$(xa(j), i), 1)

```

```

    If character = "." Then
        ndec(j) = i - 1
        Exit Do
    End If
    i = i + 1
    If i = lrg(j) + 1 Then Exit Do
Loop While character <> "."
If ndec(j) <> 0 Then
    te(1) = Left$(xa(j), lrg(j) - ndec(j) - 1)
    te(2) = Right$(xa(j), ndec(j))
    num(j) = te(1) + te(2)
Else
    num(j) = xa(j)
End If
If Left$(num(j), 2) = "-0" Then num(j) = Mid$(num(j), 2): sw = 1
If Left$(num(j), 1) = "0" Then
    Do
        If Left$(num(j), 1) = "0" Then
            num(j) = Mid$(num(j), 2)
        Else
            Exit Do
        End If
        If num(j) = "" Then num(j) = "0"
    Loop
End If
If sw = 1 Then num(j) = "-" + num(j): sw = 0
Next j
at = Multiplicar(num(), n)
dif = Len(at) - ndec(1) - ndec(2)
If dif < 0 Then
    For i = 1 To Abs(dif)
        at = "0" + at
    Next i
    at = "0." + at
Else
    If dif = 0 Then
        at = "0." + at
    Else
        ct = "." + Right$(at, ndec(1) + ndec(2))
        bt = Left$(at, Len(at) - ndec(1) - ndec(2))
        at = bt + ct
    End If
End If
sw = 0
For i = 1 To Len(at)
    If Right$(Left$(at, i), 1) = "." Then sw = 1: Exit For
Next i
' Suprimir los ceros no significativos al final del número decimal
If sw = 1 Then
    i = Len(at)
    Do
        If Right$(Left$(at, i), 1) = "0" Then
            at = Left$(at, Len(at) - 1)
            i = i - 1
        Else
            Exit Do
        End If
    Loop
End If
If Right$(at, 1) = "." Then at = Left$(at, Len(at) - 1)
If sg(1) <> sg(2) Then res = "-" + at Else res = at
MultiplicarDec = res

```

End Function

Al dividir un número decimal por otro, el resultado no siempre contiene un número finito de cifras (podría ser un número decimal periódico). En el caso particular cuando se divide por 2, el resultado será siempre un decimal. La división por 2 de los decimales es muy importante cuando se aplica el método de la bipartición para hallar los ceros de un polinomio. Por esta razón, es importante escribir una función que realice esta tarea. Al escribir el código de esta función, hay que observar que dividir entre 2 es lo mismo que multiplicar por 0.5.

```
Public Function DividirPor2Dec(ByVal xa As String, ByVal n As Integer) As String
    Dim x(2) As String, res As String
    If xa = "0" Then
        DividirPor2Dec = "0": Exit Function
    End If
    x(1) = xa: x(2) = "0.5"
    res = MultiplicarDec(x(), n)
    DividirPor2Dec = res
End Function
```

El cociente de dos números decimales es o bien un número decimal o una expresión decimal infinita periódica. Cuando se calcula el cociente hay que suministrar a la función *DividirNumerosDecimales* cuatro argumentos: el dividendo, el divisor, la precisión del cálculo (el número de las cifras que se quieren calcular después del punto decimal, y el número de las cifras en un bloque al hacer las operaciones con enteros. El cálculo del cociente Q de dos números decimales se hace de la manera siguiente: Si $ndec(1)$ es el número de las cifras después del punto decimal en el dividendo y $ndec(2)$ el número de las cifras situadas después del punto decimal en el divisor, en el número que tiene menos cifras después del punto decimal se añaden al final $|ndc(1) - ndec(2)|$ ceros. Después se suprimen los puntos decimales tanto en el dividendo como en el divisor y los ceros que pueden aparecer al principio de cada número, debido a la supresión de los puntos decimales, obteniendo dos enteros a y b . Utilizando la función *DivisionEntera* para enteros, se calcula la el cociente q y el resto r en el formato alfanumérico de a y b . Se pone $Q = q + "."$. Luego, para obtener la siguiente cifra en el cociente de los números decimales se pone $r = r + "0"$ y mediante la función *DivisionEntera*, aplicado para r y b se calcula de nuevo el cociente q_1 y el resto r_1 . Ahora se pone $Q = Q + q_1$. A continuación, para hallar la segunda cifra del cociente situada después del punto decimal se pone $r_1 = r_1 + "0"$ y mediante la función *DivisionEntera*, aplicado para r_1 y b se calcula de nuevo el cociente q_2 y el resto r_2 . El cociente de los números decimales con dos cifras después del punto decimal, será $Q = Q + q_2$. Se continua así hasta que se alcanza la precisión deseada. La función Visual-Basic que se encarga hacer todo lo expuesto anteriormente es la siguiente:

```
Public Function DividirDec(ByRef x0() As String, ByVal p As Long, ByVal n As Integer) As String
    ' p indica la precisión deseada en el resultado: 10^(-p)
    Dim i As Long, lrg(2) As Long, cont As Long, res As String, sw As Integer
    Dim x(2) As String, num(2) As String, ndec(2) As Long, su As String
    Dim dif1 As String, dif2 As String, simbolo As String, rr() As String
    Dim sg(2) As String, j As Long, te(2) As String, xa(2) As String
    If x0(1) = "0" Then
        DividirDec = "0": Exit Function
    End If
    For i = 1 To 2
        xa(i) = x0(i)
```

```

    If Left$(xa(i), 1) = "-" Then sg(i) = "-": xa(i) = Mid$(xa(i), 2)
Next i
lrg(1) = Len(xa(1)): lrg(2) = Len(xa(2))
For j = 1 To 2
    i = 1
    Do
        character = Left$(Right$(xa(j), i), 1)
        If character = "." Then
            ndec(j) = i - 1
            Exit Do
        End If
        i = i + 1
        If i = lrg(j) + 1 Then Exit Do
    Loop While character <> "."
    If ndec(j) <> 0 Then
        te(1) = Left$(xa(j), lrg(j) - ndec(j) - 1)
        te(2) = Right$(xa(j), ndec(j))
        num(j) = te(1) + te(2)
    Else
        num(j) = xa(j)
    End If
    If Left$(num(j), 2) = "-0" Then num(j) = Mid$(num(j), 2)
    If Left$(num(j), 1) = "0" Then
        Do
            If Left$(num(j), 1) = "0" Then
                num(j) = Mid$(num(j), 2)
            Else
                Exit Do
            End If
        Loop
    End If
Next j
If ndec(1) < ndec(2) Then
    For i = 1 To ndec(2) - ndec(1)
        num(1) = num(1) + "0"
    Next i
    nd = ndec(2)
End If
If ndec(1) > ndec(2) Then
    For i = 1 To ndec(1) - ndec(2)
        num(2) = num(2) + "0"
    Next i
    nd = ndec(1)
End If
If ndec(1) = ndec(2) Then nd = ndec(1)
x(1) = num(1): x(2) = num(2)
rr() = DivisionEuclidea(x(), n)
If rr(2) = "0" Then
    res = rr(1)
Else
    res = rr(1) + "."
    cont = 0
    Do
        rr(2) = rr(2) + "0"
        x(1) = rr(2): x(2) = num(2)
        rr() = DivisionEuclidea(x(), n)
        res = res + rr(1): cont = cont + 1
        If rr(2) = "0" Or cont > p Then Exit Do
    Loop
End If
If sg(1) <> sg(2) Then
    res = "-" + res

```



```

End If
For i = 1 To Len(res)
    If Right$(Left$(res, i), 1) = "." Then sw = 1: Exit For
Next i
'Supresión de ceros finales en un decimal
If sw = 1 Then
    i = Len(res)
    Do
        If Right$(Left$(res, i), 1) = "0" Then
            res = Left$(res, Len(res) - 1)
            i = i - 1
        Else
            Exit Do
        End If
    Loop
End If
If Right$(res, 1) = "." Then res = Left$(res, Len(res) - 1)
DividirDec = res
End Function

```

El cociente de dos decimales se puede hallar también por el método de la bipartición. Antes de exponer el método, hay observar que si u , v , $x(2)$ y dif son variables de tipo string, u y v contienen números, $x(1) = u$, $x(2) = v$, $n = 7$ y $dif = Re\ starDec(x(), n)$, entonces

$$u < v \Leftrightarrow Left$(dif, 1) = "-"$$

Como en las funciones anteriores, primero se cuentan, en los dos operandos, el número de las cifras después del punto decimal, luego se añaden ceros no significativos al final del operando que tiene menos y así los dos operandos tendrán el mismo número de cifras después del punto decimal. Luego se suprimen los puntos decimales y los ceros que podrían existir al principio de cada operando, después de la supresión de los puntos decimales. Así se obtienen dos enteros a y b , cuyo cociente es lo mismo que el de los operandos introducidos al principio. El cociente entero p de los enteros a y b se calcula con la función *DivisionEntera*. Entonces el cociente exacto de los números estará en el intervalo $[a(1), a(2)]$, donde $a(1) = p$ y $a(2) = p + 1$.

El punto medio c de este intervalo se obtiene dividiendo entre 2 la suma de las extremidades, utilizando las funciones *SumarDec* y *DivisionPor2Dec* y se calcula el producto pr de c con b , utilizando la función *Multiplificar*. Luego, se calcula la diferencia entre pr y a (utilizando la función *Re starDec*) y si el primer carácter del resultado es "-" entonces el nuevo intervalo que contiene el cociente exacto será $[c, a(2)]$. En el caso contrario se elegirá el intervalo $[a(1), c]$. Se repite el proceso anterior para el nuevo intervalo $[a(1), a(2)]$, hasta que la longitud de este intervalo sea inferior a la precisión deseada. La función Visual-Basic que realiza todo lo expuesto anteriormente es la siguiente:

```

Public Function DividirDecPorBiparticion(ByRef x0() As String, ByVal p As Long, ByVal n As Integer) As String
    ' p indica la precisión deseada en el resultado: 10^(-p)
    Dim i As Long, lrg(2) As Long, cont As Long, p1 As Long, res As String, sw As Integer
    Dim x(2) As String, num(2) As String, ndec(2) As Long, su As String, a(2) As String
    Dim dif1 As String, dif2 As String, simbolo As String, rr() As String
    Dim sg(2) As String, j As Long, te(2) As String, xa() As String
    If x0(1) = "0" Then
        DividirDecPorBiparticion = "0": Exit Function
    End If
    precision = "0."
    For i = 1 To p
        precision = precision + "0"
    Next i

```

```

Next i
precision = precision + "1"
xa() = x0()
For i = 1 To 2
    If Left$(xa(i), 1) = "-" Then sg(i) = "-": xa(i) = Mid$(xa(i), 2)
Next i
lrg(1) = Len(xa(1)): lrg(2) = Len(xa(2))
For j = 1 To 2
    i = 1
    Do
        character = Left$(Right$(xa(j), i), 1)
        If character = "." Then
            ndec(j) = i - 1
            Exit Do
        End If
        i = i + 1
        If i = lrg(j) + 1 Then Exit Do
    Loop While character <> "."
    If ndec(j) <> 0 Then
        te(1) = Left$(xa(j), lrg(j) - ndec(j) - 1)
        te(2) = Right$(xa(j), ndec(j))
        num(j) = te(1) + te(2)
    Else
        num(j) = xa(j)
    End If
    If Left$(num(j), 2) = "-0" Then num(j) = Mid$(num(j), 2)
    If Left$(num(j), 1) = "0" Then
        Do
            If Left$(num(j), 1) = "0" Then
                num(j) = Mid$(num(j), 2)
            Else
                Exit Do
            End If
        Loop
    End If
Next j
If ndec(1) < ndec(2) Then
    For i = 1 To ndec(2) - ndec(1)
        num(1) = num(1) + "0"
    Next i
    nd = ndec(2)
End If
If ndec(1) > ndec(2) Then
    For i = 1 To ndec(1) - ndec(2)
        num(2) = num(2) + "0"
    Next i
    nd = ndec(1)
End If
If ndec(1) = ndec(2) Then nd = ndec(1)
x(1) = num(1): x(2) = num(2)
rr() = DivisionEuclidea(x(), n)
res = rr(1) + "."
x(1) = rr(2): x(2) = num(2)
a(1) = rr(1)
x(1) = a(1): x(2) = "1"
a(2) = Sumar(x(), n)
'Método de la bipartición
Do
    su = SumarDec(a(), n)
    c = DividirPor2Dec(su, n)
    i = Len(c)
    If Right$(c, 1) = "." Then c = Left$(c, Len(c) - 1)

```

```

x(1) = c: x(2) = num(2)
nu = MultiplicarDec(x(), n)
x(1) = nu: x(2) = num(1)
dif1 = RestarDec(x(), n)
If Left$(dif1, 1) = "-" Then
    a(1) = c
Else
    a(2) = c
End If
dif = RestarDec(a(), n)
If Left$(dif, 1) = "-" Then dif = Mid$(dif, 2)
x(1) = dif: x(2) = precision
dif2 = RestarDec(x(), n)
If Left$(dif2, 1) = "-" Then: res = c: Exit Do
Loop
cont = 0
If rr(1) = "0" Then res = "0." Else res = rr(1) + "."
Do
    rr(2) = rr(2) + "0"
    x(1) = rr(2): x(2) = num(2)
    rr() = DivisionEuclidea(x(), n)
    If rr(1) = "0" Then
        res = res + "0": cont = cont + 1
    Else
        res = res + rr(1): cont = cont + 1
    Exit Do
End If
Loop
If cont < p Then p1 = p - cont
For i = 1 To p1
    rr(2) = rr(2) + "0"
Next i
x(1) = rr(2): x(2) = num(2)
rr() = DivisionEuclidea(x(), n)
If rr(1) <> "0" Then
    res = res + rr(1)
End If
For i = 1 To Len(res)
    If Right$(Left$(res, i), 1) = "." Then sw = 1: Exit For
Next i
'Supresión de ceros finales en un decimal
If sw = 1 Then
    i = Len(res)
    Do
        If Right$(Left$(res, i), 1) = "0" Then
            res = Left$(res, Len(res) - 1)
            i = i - 1
        Else
            Exit Do
        End If
    Loop
End If
If sg(1) <> sg(2) Then
    res = "-" + res
End If
DividirDecPorBiparticion = res
End Function

```

El cálculo de las potencias de base decimal y exponente natural se reduce a la multiplicación de los decimales, utilizando la función *MultiplicarDec* y el código es la siguiente:

```

Public Function PotenciasDec(ByRef xa() As String, ByVal n As Integer) As String
Dim z(2) As String, y(2) As String, sgb As Integer, pr As String, ed As Double, i As String
If xa(1) = "0" And xa(2) <> "0" Then
    PotenciasDec = "0"
    Exit Function
End If
If xa(2) = "0" And xa(1) <> "0" Then
    PotenciasDec = "1"
    Exit Function
End If
If xa(2) = "1" Then
    PotenciasDec = xa(1)
    Exit Function
End If
If Left$(xa(1), 1) = "-" Then
    xa(1) = Mid$(xa(1), 2): sgb = 1
End If
pexp = Val(xa(2)) Mod 2
y(1) = xa(1): y(2) = xa(1): i = "1"
Do
    pr = MultiplicarDec(y(), n)
    y(1) = pr: z(1) = i: z(2) = "1"
    i = Sumar(z(), n)
    If i = xa(2) Then Exit Do
Loop
If sgb = 0 Then
    PotenciasDec = pr
Else
    If pexp = 0 Then PotenciasDec = pr Else PotenciasDec = "-" + pr
End If
End Function

```

Para el cálculo de la raíz de índice p de un decimal positivo α se puede utilizar también el método de la bipartición. Obviamente, si $\alpha = 1$, entonces $\sqrt[p]{\alpha} = 1$ y si $\alpha = 0$, entonces $\sqrt[p]{\alpha} = 0$. Luego, si $\alpha > 1$ entonces $\sqrt[p]{\alpha} \in (1, \alpha)$ y si $0 < \alpha < 1$ entonces $\sqrt[p]{\alpha} \in (0, \alpha)$. Por tanto, si el intervalo donde se aplicará el método de la bipartición es $(a(1), a(2))$ entonces

$$\alpha > 1 \Rightarrow a(1) = 1 \text{ y } a(2) = \alpha$$

$$0 < \alpha < 1 \Rightarrow a(1) = 0 \text{ y } a(2) = 1.$$

El punto medio c de este intervalo se obtiene dividiendo entre 2 la suma de las extremidades, utilizando las funciones *SumarDec* y *DivisionPor2Dec* y se calcula la potencia c^p , utilizando la función *PotenciasDec*. Luego se calcula la diferencia entre c^p y α (utilizando la función *RestarDec*) y si el primer carácter del resultado es "-" entonces el nuevo intervalo que contiene el cociente exacto será $[c, a(2)]$. En el caso contrario se elegirá el intervalo $[a(1), c]$. Se repite el proceso anterior para el nuevo intervalo $[a(1), a(2)]$, hasta que la longitud de este intervalo sea inferior a la precisión deseada. La función Visual-Basic que efectúa todo lo expuesto anteriormente es la siguiente:

```

Public Function RaizIndiceNDecPos(ByRef x0() As String, ByVal p As Long, ByVal n As Integer) As String
' p indica la precisión deseada en el resultado: 10^(-p)
' m=xa(2) es el índice de la raíz a calcular.
Dim i As Long, a(2) As String, b As String, c As String, precision As String
Dim x(2) As String, su As String, j As String, nu As String, dif As String
Dim dif1 As String, dif2 As String, simbolo As String, res1 As String, xa() As String
If x0(1) = "1" Then

```

```

    RaizIndiceNDecPos = "1"
    Exit Function
End If
If x0(1) = "0" Then
    RaizIndiceNDecPos = "0"
    Exit Function
End If
xa() = x0()
If Left$(xa(1), 1) = "-" Then xa(1) = Mid$(xa(1), 2)
m = xa(2): num = xa(1)
precision = "0."
For i = 1 To p
    precision = precision + "0"
Next i
precision = precision + "1"
nu = xa(1)
If nu = "1" Then res = "1"
x(1) = nu: x(2) = "1"
dif = RestarDec(x(), n)
If Left$(dif, 1) = "-" Then
    a(1) = nu: a(2) = "1"
Else
    a(1) = "1": a(2) = nu
End If
'Método de la bipartición
Do
    su = SumarDec(a(), n)
    c = DividirPor2Dec(su, n)
    i = Len(c)
    If Right$(c, 1) = "." Then c = Left$(c, Len(c) - 1)
    x(1) = c: x(2) = m
    nu = PotenciasDec(x(), n)
    x(1) = nu: x(2) = num
    dif1 = RestarDec(x(), n)
    If Left$(dif1, 1) = "-" Then
        a(1) = c
    Else
        a(2) = c
    End If
    dif = RestarDec(a(), n)
    If Left$(dif, 1) = "-" Then dif = Mid$(dif, 2)
    x(1) = dif: x(2) = precision
    dif2 = RestarDec(x(), n)
    If Left$(dif2, 1) = "-" Then: res = c: Exit Do
Loop
For i = 1 To Len(c)
    simbolo = Right$(Left$(res, i), 1)
    If simbolo = "." Then
        If i + p < Len(c) Then
            res = Left$(res, i + p)
        Else
            Exit For
        End If
    End If
Next i
cont = 0: res1 = res
'Suprimir cifras 9 finales
If Right$(res, 1) = "9" Then
    For i = 1 To Len(res)
        simbolo = Right$(Left$(res, i), 1)
        cont = cont + 1
        If simbolo = "." Then Exit For
    Next i

```

```

Next i
re = "0."
For i = 1 To Len(res) - cont - 1
    re = re + "0"
Next i
re = re + "1"
x(1) = res: x(2) = re
res = SumarDec(x(), n)
x(1) = res: x(2) = res
nu = MultiplicarDec(x(), n)
If nu <> xa(1) Then res = res1
End If
'Supreción de ceros finales en un decimal
If Right$(res, 1) = "0" Then
    res1 = res
    i = Len(res)
    Do
        simbolo = Right$(Left$(res, i), 1)
        If simbolo = "0" Then
            res = Left$(res, Len(res) - 1)
            i = i - 1
        Else
            Exit Do
        End If
    Loop
    If simbolo = "." Then res = Left$(res, 1)
    x(1) = res: x(2) = res
    nu = MultiplicarDec(x(), n)
    If nu <> xa(1) Then res = res1
End If
If Right$(res, 1) = "." Then res = Mid$(res, 2)
RaizIndiceNDecPos = res
End Function

```

Aplicación: Si $a = 53879567.9876543567898765$, hallar $\sqrt[5]{a}$ con 200 cifras después del punto decimal. Un ordenador con la velocidad de 3GHz, utilizando el programa anterior, dio el resultado siguiente, aproximadamente en 111.87s:

$\sqrt[5]{a} \approx$

```

35.17910781049426924087066806565839130896234615100836900239493192631547129717301
1383095931652123248355613511564328856104931612103436830862503055545116233343906
4581996855426835039116088069183501522832185

```

Para hallar el valor de un polinomio con coeficientes (enteros o decimales) para un valor a (entero o decimal) se puede utilizar la siguiente variante del esquema de Ruffini, donde tanto los coeficientes del polinomio como la variable a son de tipo string:

```

Public Function ValorPolinomio(ByRef p() As String, ByVal a As String, n As Integer) As String
    Dim i As Long, gp As Integer, q() As String, prod As String, x(2) As String
    ' Método de Ruffini
    gp = UBound(p())
    q() = p()
    For i = 0 To gp - 1
        x(1) = q(i): x(2) = a
        prod = MultiplicarDec(x(), n)
        x(1) = prod: x(2) = p(i + 1)
        q(i + 1) = SumarDec(x(), n)
    Next i
    ValorPolinomio = q(gp)

```

End Function

Los ceros de un polinomio con coeficientes enteros o decimales son números algebraicos y se pueden aproximar por el método de la bipartición con la exactitud deseada. El código Visual-Basic es la siguiente:

```
Public Function CNAR(ByRef p() As String, ByRef ab() As String, ByVal pr As Long, ByVal n As Integer) As String
    ' Cálculo de los números algebraicos reales con precisión grande
    Dim i As Double, a(2) As String, fa1 As String, fa2 As String, fc As String
    Dim x(2) As String, su As String, j As String, dif As String, simbolo As String
    Dim sgfa1 As String, sgfa2 As String, sgfc As String, dif1 As String
    Dim c As String, precision As String
    a(1) = ab(0): a(2) = ab(1)
    precision = "0."
    For i = 1 To pr
        precision = precision + "0"
    Next i
    precision = precision + "1"
    fa1 = ValorPolinomio(p(), a(1), n)
    sgfa1 = Left$(fa1, 1)
    If sgfa1 <> "-" Then sgfa1 = ""
    fa2 = ValorPolinomio(p(), a(2), n)
    sgfa2 = Left$(fa2, 1)
    If sgfa2 <> "-" Then sgfa2 = ""
    If sgfa1 = sgfa2 Then
        MsgBox "¡En el intervalo no hay ningún cero!"
        Exit Function
    End If
    Do
        su = SumarDec(a(), n)
        c = DividirPor2Dec(su, n)
        'i = Len(c)
        If Right$(c, 1) = "." Then c = Left$(c, Len(c) - 1)
        fc = ValorPolinomio(p(), c, n)
        sgfc = Left$(fc, 1)
        If sgfc <> "-" Then sgfc = ""
        If sgfa1 <> sgfc Then
            a(2) = c: fa2 = fc: sgfa2 = sgfc
        Else
            a(1) = c: fa1 = fc: sgfa1 = sgfc
        End If
        dif = RestarDec(a(), n)
        If Left$(dif, 1) = "-" Then dif = Mid$(dif, 2)
        x(1) = dif: x(2) = precision
        dif1 = RestarDec(x(), n)
        If Left$(dif1, 1) = "-" Then: res = c: Exit Do
    Loop
    For i = 1 To Len(res)
        simbolo = Right$(Left$(res, i), 1)
        If simbolo = "." Then
            If i + pr < Len(res) Then
                res = Left$(res, i + pr)
            Else
                Exit For
            End If
        End If
    Next i
    CNAR = res
End Function
```

Aplicación: El polinomio $12x^7 - 6x^6 - 21x^5 - 3x^4 + 11x^3 - 8x^2 + 5x - 30$ tiene un único cero α en el intervalo $(1,2)$. El ordenador, utilizando el programa anterior (aproximadamente en 88.78s y a 3GHz de velocidad), dio el resultado siguiente con 150 cifras después del punto decimal:

$$\alpha \approx 1.6388527\ 4730641416030000533660078420037\ 4301441546880693821\ 75787361016977308590\ 946246114847124217985028584550526307731131861090719109811733280547054084$$

Observación: Teóricamente es posible trabajar con números decimales muy largos, sin embargo, los límites existen debido al tiempo y la cantidad de memoria disponibles.

Para introducir un polinomio en el ordenador es muy cómodo teclear sus coeficientes en una caja de textos, separados por comas. Por ejemplo, si los coeficientes se introducen en Form1.Text1 y la variable `t` es de tipo String, definida en Option Explicit, entonces los procedimientos siguientes separan los coeficientes del polinomio como elementos de una matriz unidimensional de tipo String:

```
Private Sub Command1_Click(Index As Integer)
    t = Text1
End Sub
=====
Public Function SeparacionNumeros(ByVal t As String) As Variant
    Dim i As Long, j As Long, k As Long, i0 As Long
    Dim nco As Long, gp As Integer, lt As Long, bb As String, px() As String
    '---- Número de las comas en la cadena t
    If Right$(t, 1) <> "," Then t = t + ","
    k = 1: lt = Len(t): nco = 0
    Do
        bb = Right$(Left$(t, k), 1)
        If bb = "," Then
            nco = nco + 1
        End If
        k = k + 1
        If k > lt Then Exit Do
    Loop
    gp = nco - 1
    '--- Separación de los coeficientes
    ReDim px(gp)
    k = 1: i = 1: i0 = 0: j = 0
    Do
        bb = Right$(Left$(t, k), 1)
        If bb = "," Then
            j = j + 1
            px(i0) = Left$(t, k - 1)
            i = i + 1: i0 = i0 + 1
            t = Mid$(t, k + 1)
            k = 1
        Else
            k = k + 1
        End If
        If j = nco Then Exit Do
    Loop
    SeparacionNumeros = px()
End Function
```


Bibliografía:

- [1] Aladar Peter Santha, Cálculos con números enteros grandes en ordenadores, Monografías.com, 31/01/2012.