

Cálculos con números enteros largos en ordenadores. Aladár Péter Sántha

Haciendo cálculos con lápiz y papel, el tamaño de los operandos está limitado solamente por el tamaño del papel y por el tiempo disponible. Cuando los cálculos se hacen con un ordenador, además de la cantidad de memoria y el factor tiempo, hay que tener en cuenta que las variables de precisión doble pueden almacenar $2p$ cifras. Normalmente, tenemos $2p = 16$. Un número con más de $2p$ cifras se presentará en punto flotante. Esto significa que conoceremos solamente las primeras $2p - 1$ cifras del resultado puesto que la cifra que ocupa el lugar $2p$ ya está afectada por el redondeo. Por otra parte, las dificultades empiezan ya al introducir los operandos en el ordenador. Al intentar introducir un operando de más de $2p$ cifras en una variable numérica de precisión doble, el resultado será almacenado en punto flotante, perdiendo las últimas cifras. Así pues, haciendo operaciones con números grandes, los operandos deberán ser introducidos en variables alfanuméricas (de tipo string), que pueden recibir números largos. El número de las cifras de un número que se puede introducir en una variable de tipo alfanumérico está limitado únicamente por la memoria disponible en el ordenador. Haciendo cálculos con números grandes, los operandos deberán ser divididos en bloques. El tamaño de los bloques depende de las operaciones a efectuar. El resultado de la operación entre los números situados en dos bloques se almacenará en una variable de doble precisión y así, para sumas, restas y divisiones euclideas, el número de las cifras por bloque no tiene que ser mayor a $n = 2p - 1$. En el caso de las multiplicaciones, el número de las cifras por bloque no debe superar las $n = (2p - 2) / 2$ cifras.

Cuando se efectúan todo tipo de operaciones, es conveniente que el número de cifras por bloque sea siempre el mismo, menor o igual a $n = (2p - 2) / 2$.

Ejemplo 1: Para sumar los números

$$A = 66792450456123787438765 \text{ y } B = 29456299730597435689,$$

, los números se dividirán en bloques de 8 cifras. La formación de los bloques se hará desde la extremidad derecha de cada número, hacia la izquierda. Dado que las calculadoras de bolsillo exponen 10 cifras, los cálculos de este ejemplo se pueden seguir con éstas calculadoras.

$$A = 6679245 \ 04561237 \ 87438765$$

$$B = 2945 \ 62997305 \ 97435689$$

Sumando los bloques que ocupan el mismo lugar, se obtienen los números

$$6682190 \ 67558542 \ 184874454$$

, donde el último número no es un bloque de 8 cifras. Separando la cifra 1 del número 184874454 se obtiene el arrastre 1 que se sumará al bloque siguiente 67558542. Así, se obtienen los bloques:

$$6682190 \ 67558543 \ 84874454$$

, y uniendo los bloques, resulta que:

$$A + B = 66821906755854384874454$$

Ejemplo 2: Para calcular la diferencia de los números

$$A = 58987633349576152374887 \text{ y } B = 5896875869678361876$$

, los números A y B se dividirán en bloques de 8 cifras:

$$A = 5898763 \ 33495761 \ 52374887$$

$$B = 589\ 68758696\ 78361876$$

Restando de los bloques de A los bloques de B que ocupan el mismo lugar, se obtienen los números:

$$5898174 - 35262935 - 25986989$$

Para hallar los bloques de A - B hay que sumar 10^8 a -25986989 y restar 1 del número -35262935, luego, hay que repetir lo mismo para los primeros dos números. Así, A-B se obtiene por el camino siguiente:

$$5898174 - 35262936\ 74013011$$

$$5898174 - 35262935\ 74013011$$

$$5898173\ 64737065\ 74013011$$

$$A - B = 58981736473706574013011$$

Para hallar B-A, los cálculos se harían de la misma manera, cambiando previamente el primer operando con el segundo, luego el resultado tendrá el signo menos.

En el caso de la suma algebraica de dos enteros, es conveniente separar los signos de los números desde el principio y operar con los valores absolutos, de acuerdo con las reglas siguientes:

Si los dos números tienen el mismo signo, se suman los valores absolutos y a esta suma se le asigna el signo común de los operandos.

Si los dos números son de signos distintos, pero no son opuestos, del número de mayor valor absoluto se resta el otro y el resultado tendrá el signo del número de mayor valor absoluto.

Si los números son opuestos, el resultado es cero.

Teniendo en cuenta que en un programa de ordenador los valores absolutos de los números aparecerán en campos alfanuméricos, la comparación de los valores absolutos se hará según las reglas siguientes: Si A\$ y B\$ son los campos alfanuméricos que contienen los números A y B, respectivamente, y la longitud del campo A\$ es mayor que la del campo B\$, entonces $A > B$. Si los campos A\$ y B\$ tienen la misma longitud, $A > B$ si y solamente si $A\$ > B\$$. Recordamos que la comparación de A\$ y B\$ se hace comparando sucesivamente las cifras que ocupan la misma posición en las dos variables, de izquierda a la derecha. Al encontrar las primeras dos cifras distintas, el campo mayor será aquello que contiene el mayor de estas dos cifras. Evidentemente, si todas las cifras, que ocupan posiciones idénticas, coinciden, los dos campos son iguales.

Para multiplicar los números $A = 52317$ y $B = 15341245$, se dividen estos números en bloques de dos cifras: $A = 5\ 23\ 17$ y $B = 15\ 34\ 12\ 45$.

Entonces, $A = 17 + 23 \cdot 10^2 + 5 \cdot 10^4$ y $B = 45 + 12 \cdot 10^2 + 34 \cdot 10^4 + 15 \cdot 10^6$

, y multiplicando todos los sumandos de A con los sumandos de B, sucesivamente, resulta que

$$\begin{aligned} A \cdot B &= (225 \cdot 10^4 + 1035 \cdot 10^2 + 765) + (60 \cdot 10^6 + 276 \cdot 10^4 + 204 \cdot 10^2) + \\ &\quad + (170 \cdot 10^8 + 782 \cdot 10^6 + 578 \cdot 10^4) + (75 \cdot 10^{10} + 345 \cdot 10^8 + 255 \cdot 10^6) = \\ &= [65 + (35 + 7) \cdot 10^2 + (25 + 10) \cdot 10^4 + 2 \cdot 10^6] + [4 \cdot 10^2 + (76 + 2) \cdot 10^4 + (60 + 2) \cdot 10^6] + \\ &\quad + [78 \cdot 10^4 + (82 + 5) \cdot 10^6 + (70 + 7) \cdot 10^8 + 1 \cdot 10^{10}] + [55 \cdot 10^6 + (45 + 2) \cdot 10^4 + (75 + 3) \cdot 10^{10}] \end{aligned}$$

$$A \cdot B = 65 + (35 + 7 + 4) \cdot 10^2 + (25 + 10 + 76 + 2 + 78) \cdot 10^4 + (2 + 60 + 2 + 82 + 5 + 55) \cdot 10^6 +$$

$$\dots + (70 + 7 + 45 + 2) \cdot 10^8 + (1 + 75 + 3) \cdot 10^{10} \quad (1)$$

25	35	65
60	76	04
70	82	78
75	45	55

Piso bajo

02	10	07
00	02	02
01	07	05
00	03	02

Piso superior

Así los elementos del piso bajo y del piso superior de A son $A(i, j, 1)$ y $A(i, j, 2)$, respectivamente, donde $1 \leq i \leq 4$ y $1 \leq j \leq 3$. Por tanto, las sumas de los números de la tabla 1, que se encuentran sobre las líneas aa' , bb' , cc' , dd' , ee' , ff' y gg' son:

$$\begin{aligned} &A(1,3,1) \\ &A(1,2,1)+A(2,3,1)+A(1,3,2) \\ &A(1,1,1)+A(2,2,1)+A(3,3,1)+A(1,2,2)+A(2,3,2) \\ &A(2,1,1)+A(3,2,1)+A(4,3,1)+A(1,1,2)+A(2,2,2)+A(3,3,2) \\ &A(1,3,1)+A(2,4,1)+A(2,1,2)+A(3,2,2)+A(4,3,2) \\ &A(1,4,1)+A(1,3,2)+A(2,4,2) \\ &A(1,4,2). \end{aligned}$$

Obviamente, el mismo producto se podría calcular formando bloques con más cifras. Por ejemplo, si se forman bloques de 3 cifras, los cálculos son los siguientes:

$$A = 52 \ 317 \ ; \ B = 15 \ 341 \ 245$$

12	740	77	665
17	732	108	097
	780	4	755

Tabla 2

801	1607	914	665
802	607	914	665

Tabla 2

, y así $A \cdot B = 802607914665$, como en el caso anterior.

Ejemplo 3: Si $A = 547888431678967$ y $B = 99345687652349$, los números sean divididos en bloques de longitud 5

$$A = 54788 \ 84316 \ 78967 ; B = 9934 \ 56876 \ 52349$$

97012	58284	43483	<i>Tabla 3</i>					
28680	44139	41338						
22888	56816	27092						
31161	47953	4913						
63992	95144	58175						
5442	8373	7844						
			5442	103528	201911	301057	126714	43483
			5443	03530	01914	01058	26714	43483

, y así $A \cdot B = 54430353001914010582671443483$.

La división euclídea de dos números naturales se reduce a una sucesión de restas, de fácil programación. No obstante, la duración de una división euclídea es bastante superior al tiempo necesario para efectuar a una suma o a una multiplicación.

El método expuesto para efectuar las operaciones con números naturales grandes, dividiéndolos en bloques, se conoce ya desde hace mucho tiempo pero el autor de este escrito no pudo averiguar de quien fue la idea. A continuación se exponen las funciones necesarias en el lenguaje VISUAL-BASIC, para efectuar operaciones con números **enteros** grandes. Para esto, al principio se suprimen los signos de los operandos y luego se establece el signo del resultado. La matriz xa() contiene los dos operandos, y n es el número de las cifras por bloque. En el caso de la división euclídea el resultado está contenido en la matriz rt(), rt(1) siendo el cociente entero y rt(2) el resto.

```
Public Function Multiplicar(ByRef xa() As String, ByVal n As Integer) As String
    Dim i As Integer, j As Integer, k As Integer, l(2) As Double
    Dim m() As Double, a() As Double, x(2) As String, r1() As Double, r2() As Double
    Dim prod As Double, p As Double, p0 As String, at As String
    Dim sg1 As String, sg2 As String, sg As String, bt As String
    Dim xx As String, ll As Double, da As Integer, c As Integer, f As Integer
    For i = 1 To 2: x(i) = xa(i): Next i
    '----- RutinaSigno
    sg = "": sg1 = "": sg2 = "": at = "": bt = ""
    For i = 1 To 2
        If Left$(x(i), 1) = " " then x(i) = Mid$(x(i), 2)
        If Left$(x(i), 1) = "-" Then
            sg = "-"
            x(i) = Mid$(x(i), 2)
        End If
        If i = 1 Then sg1 = sg Else sg2 = sg
        l(i) = Len(x(i)): sg = ""
    Next i
    '----- Multiplicacion
    If l(1) + l(2) <= 2*n Then
        prod = Val(x(1)) * Val(x(2))
        If prod <> 0 Then
            If sg1 <> sg2 Then
```

```

        at = "-"
    End If
    at = at + Mid$(Str$(prod), 2)
Else
    at = "0"
End If
Else
    Dim nb As Integer, nc As Integer
    If l(1) > l(2) Then
        xx = x(1): x(1) = x(2): x(2) = xx 'Cambio
        ll = l(1): l(1) = l(2): l(2) = ll
    End If
    r1() = FormacionBloques(x(1), l(1), n)
    c = UBound(r1())
    r2() = FormacionBloques(x(2), l(2), n)
    f = UBound(r2())
    ReDim m(f, c, 2), a(c + f)
    k = 1
    For j = 1 To f
        For i = 1 To c
            p = r1(i) * r2(j)
            p0 = Right$(Str$(p), 2 * n)
            If Left$(p0, 1) = " " Then
                p0 = Mid$(p0, 2)
            End If
            nb = Len(p0)
            If nb = 2 * n Then
                nc = n
            Else
                nc = nb - n
            End If
            If nc < 0 Then nc = 0
            m(k, c - i + 1, 2) = Val(Left$(p0, nc))
            m(k, c - i + 1, 1) = Val(Right$(p0, n))
        Next i
        k = k + 1
    Next j
    For k = 1 To c + f - 1
        If k < c + 1 Then
            i = 1: j = c - k + 1
        Else
            i = k - c + 1: j = 1
        End If
        Do While i <= f
            a(k) = a(k) + m(i, j, 1)
            a(k + 1) = a(k + 1) + m(i, j, 2)
            If j <> c Then
                i = i + 1: j = j + 1
            Else
                Exit Do
            End If
        Loop
    Next k
    da = c + f - 1
    bt = Desbloqueo(a(), da, n)
    If sg1 = sg2 Then
        at = bt
    Else

```

```

        at = "-" + bt
        If at = "-0" Then at = "0"
    End If
End If
Multiplicar = at
End Function

' =====

Public Function Sumar(ByRef xa() As String, ByVal n As Integer) As String
    Dim i As Integer, i0 As Integer, j As Integer, k As Integer, l(2) As Double
    Dim a() As Double, rt(2) As Variant, x(2) As String, at As String, bt As String
    Dim Sum As Double, p As Double, p0 As String, r1() As Double, r2() As Double
    Dim sg1 As String, sg2 As String, sg As String, xq As Long
    Dim xx As String, ll As Double, da As Integer, c As Integer, f As Integer
    For i = 1 To 2: x(i) = xa(i): Next i
    '----- RutinaSigno
    sg = "": sg1 = "": sg2 = ""
    For i = 1 To 2
        If left$(x(i), 1) = " " then x(i) = Mid$(x(i), 2)
        If Left$(x(i), 1) = "-" Then
            sg = "-"
            x(i) = Mid$(x(i), 2)
        End If
        If i = 1 Then sg1 = sg Else sg2 = sg
        l(i) = Len(x(i)): sg = ""
    Next i
    '----- Suma
    If l(1) <= 2*n And l(2) <= 2*n Then
        x(1) = sg1 + x(1)
        x(2) = sg2 + x(2)
        Sum = Val(x(1)) + Val(x(2))
        at = Str$(Sum)
        If Left$(at, 1) = " " Then
            at = Mid$(at, 2)
        End If
    Else
        at = "": bt = ""
        Dim cf As Integer, fr As Integer
        Dim a0 As Double
        If l(1) > l(2) Then
            sg = sg1
        Else
            If l(1) < l(2) Then 'Cambio
                xx = x(1): x(1) = x(2): x(2) = xx
                ll = l(1): l(1) = l(2): l(2) = ll
                sg = sg2
            Else
                If x(1) > x(2) Then
                    sg = sg1
                Else
                    If x(1) = x(2) Then
                        sg = ""
                    Else 'Cambio
                        xx = x(1): x(1) = x(2): x(2) = xx
                        ll = l(1): l(1) = l(2): l(2) = ll
                        sg = sg2
                    End If
                End If
            End If
        End If
    End If

```

```

        End If
    End If
End If
r1() = FormacionBloques(x(1), l(1), n)
c = UBound(r1())
r2() = FormacionBloques(x(2), l(2), n)
f = UBound(r2())
cf = c
If cf < f Then cf = f
ReDim a(cf + 1)
da = cf
If sg1 <> sg2 Then fr = -1 Else fr = 1
For k = c To 1 Step -1
    a(k) = a(k) + r1(k)
Next k
For k = f To 1 Step -1
    a(k) = a(k) + fr * r2(k)
Next k
If fr = -1 Then
    For k = 1 To c
        If a(k) < 0 Then
            a0 = 1
            For i = 1 To n
                a0 = a0 * 10
            Next i
            a(k) = a0 + a(k)
            a(k + 1) = a(k + 1) - 1
        End If
    Next k
End If
bt = Desbloqueo(a(), da, n)
at = sg + bt
If at = "-0" Then at = "0"
End If
Sumar = at
End Function

' =====

Public Function Restar(ByRef xa() As String, Byval n As Integer) As String
    Dim i As Integer, i0 As Integer, j As Integer, k As Integer
    Dim a() As Double, x(2) As String, at As String, bt As String, a0 As Double
    Dim dif As Double, p As Double, p0 As String, r1() As Double, r2() As Double
    Dim sg1 As String, sg2 As String, l(2) As Double, sg As String, xq As Long
    Dim xx As String, ll As Double, da As Integer, c As Integer, f As Integer
    For i = 1 To 2: x(i) = xa(i): Next i
    '----- RutinaSigno
    sg = "": sg1 = "": sg2 = ""
    For i = 1 To 2
        If left$(x(i), 1) = " " then x(i) = Mid$(x(i), 2)
        If Left$(x(i), 1) = "-" Then
            sg = "-"
            x(i) = Mid$(x(i), 2)
        End If
        If i = 1 Then sg1 = sg Else sg2 = sg
        l(i) = Len(x(i)): sg = ""
    Next i
    '----- Resta

```



```

at = "": bt = ""
If l(1) <= 2*n And l(2) <= 2*n Then
    x(1) = sg1 + x(1)
    x(2) = sg2 + x(2)
    dif = Val(x(1)) - Val(x(2))
    at = Str$(dif)
    If Left$(at, 1) = " " Then
        at = Mid$(at, 2)
    End If
Else
    Dim cf As Integer, fr As Integer
    If sg2 = "-" Then sg2 = "" Else sg2 = "-"
    If l(1) > l(2) Then
        sg = sg1
    Else
        If l(1) < l(2) Then 'Cambio
            xx = x(1): x(1) = x(2): x(2) = xx
            ll = l(1): l(1) = l(2): l(2) = ll
            sg = sg2
        Else
            If x(1) > x(2) Then
                sg = sg1
            Else
                If x(1) = x(2) Then
                    sg = ""
                Else 'Cambio
                    xx = x(1): x(1) = x(2): x(2) = xx
                    ll = l(1): l(1) = l(2): l(2) = ll
                    sg = sg2
                End If
            End If
        End If
    End If
End If
r1() = FormacionBloques(x(1), l(1), n)
c = UBound(r1())
r2() = FormacionBloques(x(2), l(2), n)
f = UBound(r2())
cf = c
If cf < f Then cf = f
ReDim a(cf + 1)
da = cf
If sg1 <> sg2 Then fr = -1 Else fr = 1
For k = c To 1 Step -1
    a(k) = a(k) + r1(k)
Next k
For k = f To 1 Step -1
    a(k) = a(k) + fr * r2(k)
Next k
If fr = -1 Then
    For k = 1 To c
        If a(k) < 0 Then
            a0 = 1
            For i = 1 To n
                a0 = a0 * 10
            Next i
            a(k) = a0 + a(k)
            a(k + 1) = a(k + 1) - 1
        End If
    End If

```

```

        Next k
    End If
    bt = Desbloqueo(a(), da, n)
    at = sg + bt
    If at = "-0" Then at = "0"
    End If
    Restar = at
End Function

'=====

Public Function DivisionEuclidea(ByRef xa() As String, ByVal n As Integer) As Variant
    Dim i As Integer, i0 As Integer, j As Integer, k As Integer
    Dim m() As Double, a() As Double, rt(2) As String, x(2) As String
    Dim dif As String, coci As Double, qx As String
    Dim sg1 As String, sg2 As String, l(2) As Double, sg As String, xq As Long
    Dim ll As Double, da As Integer, rx As String, at As String, bt As String
    For i = 1 To 2: x(i) = xa(i): Next i
    '----- RutinaSigno
    sg = "": sg1 = "": sg2 = ""
    For i = 1 To 2
        If left$(x(i), 1) = " " then x(i) = Mid$(x(i), 2)
        If Left$(x(i), 1) = "-" Then
            sg = "-"
            x(i) = Mid$(x(i), 2)
        End If
        If i = 1 Then sg1 = sg Else sg2 = sg
        l(i) = Len(x(i)): sg = ""
    Next i
    '----- Division Euclidea
    If l(1) < l(2) Or (l(1) = l(2) And x(1) < x(2)) Then
        rt(1) = "0": rt(2) = xa(1)
        DivisionEuclidea = rt()
        Exit Function
    End If
    If l(1) < 2*n+2 And (l(1) > l(2) Or (l(1) = l(2) And x(1) >= x(2))) Then
        coci = Int(Val(x(1)) / Val(x(2)))
        qx = Mid$(Str$(coci), 2)
        rx = Mid$(Str$(Val(x(1)) - coci * Val(x(2))), 2)
    Else
        Dim dl As Integer
        Dim zz As String
        qx = "": rx = "": xq = 0
        dl = Abs(l(1) - l(2))
        If dl <> 0 Then
            If x(1) < x(2) Then
                dl = dl - 1
                If dl <> 0 Then
                    For k = 1 To dl
                        x(2) = x(2) + "0"
                    Next k
                    l(2) = l(2) + dl
                End If
            Else
                For k = 1 To dl
                    x(2) = x(2) + "0"
                Next k
                l(2) = l(2) + dl
            End If
        End If
    End If

```

```

End If
End If
Do
    Do
        dif = Restar(x(), n)
        If Left$(dif, 1) = "-" Then
            x(1) = Mid$(dif, 2)
        Else
            x(1) = dif
        End If
        xq = xq + 1
        If Len(x(1)) < Len(x(2)) Then Exit Do
        If Len(x(1)) = Len(x(2)) And x(1) < x(2) Then
            Exit Do
        Else
            at = "": bt = "": l(1) = Len(x(1))
        End If
    Loop
    zz = Str(xq): xq = 0
    Do
        If Left$(zz, 1) = " " Then
            zz = Mid$(zz, 2)
        Else
            Exit Do
        End If
    Loop
    Do
        qx = qx + zz
        If dl = 0 Then
            rx = x(1): Exit Do
        End If
        x(2) = Mid$(x(2), 1, Len(x(2)) - 1)
        If Len(x(1)) > Len(x(2)) Then Exit Do
        If Len(x(1)) = Len(x(2)) And x(1) >= x(2) Then
            Exit Do
        Else
            dl = dl - 1: zz = "0"
        End If
    Loop
    If dl = 0 Then Exit Do
    dl = dl - 1
    l(1) = Len(x(1)): l(2) = Len(x(2))
Loop
End If
If qx <> "0" Then
    If sg1 = "" And sg2 = "" Then
        rt(1) = qx: rt(2) = rx
    End If
    If sg1 = "-" And sg2 = "-" Then
        rt(1) = qx: rt(2) = "-" + rx
    End If
    If sg1 = "" And sg2 = "-" Then
        rt(1) = "-" + qx: rt(2) = rx
    End If
    If sg1 = "-" And sg2 = "" Then
        rt(1) = "-" + qx: rt(2) = "-" + rx
    End If
Else

```

```

        rt(1) = "0" And rt(2) = xa(2)
    End If
    If rt(2) = "-0" Then rt(2) = "0"
    DivisionEuclidea = rt()
End Function

'=====

Public Function Potencias(ByRef xa() As String, ByVal n As Integer) As String
    Dim i As Long, x(2) As String, sgb As Integer, pexp As Integer
    Dim ed As String, j As String, pr As String, y(2) As String, z(2) As String
    For i = 1 To 2
        x(i) = xa(i)
        If Left$(x(i), 1) = " " Then x(i) = Mid$(x(i), 2)
    Next i
    If Left$(x(1), 1) = "-" Then
        x(1) = Mid$(x(1), 2): sgb = 1
    End If
    If Left$(x(2), 1) = "-" Then
        x(2) = Mid$(xa(2), 2) ' el exponente no puede ser negativo
    End If
    If x(1) = "0" And x(2) = "0" Then
        Potencias = "Resultado indeterminado"
        Exit Function
    End If
    If x(1) = "0" And x(2) <> "0" Then
        Potencias = "0"
        Exit Function
    End If
    If x(2) = "0" And x(1) <> "0" Then
        Potencias = "1"
        Exit Function
    End If
    If x(2) = "1" Then
        Potencias = x(1)
        Exit Function
    End If
    ed = Right$(x(2), 1)
    pexp = Val(ed) Mod 2
    y(1) = x(1): y(2) = x(1): j = "1"
    Do
        pr = Multiplicar(y(), n)
        y(1) = pr: z(1) = j: z(2) = "1"
        j = Sumar(z(), n)
        If j = x(2) Then Exit Do
    Loop
    If sgb = 0 Then
        Potencias = pr
    Else
        If pexp = 0 Then Potencias = pr Else Potencias = "-" + pr
    End If
End Function

'=====

Public Function MaxComDiv(ByRef xa() As String, ByVal n As Integer) As String
    Dim i As Integer, j As Integer, k As Integer
    Dim x(2) As String, l(2) As Double, rr() As String

```

```

Dim xx As String, ll As Double, rr0 As String
Dim q0 As Double, q1 As Double, r0 As Double, r1 As Double
For i = 1 To 2: x(i) = xa(i): Next i
'----- RutinaSigno
For i = 1 To 2
    If left$(x(i),1) = " " then x(i) = Mid$(x(i), 2)
    If Left$(x(i), 1) = "-" Then x(i) = Mid$(x(i), 2)
    l(i) = Len(x(i))
Next i
'Máximo Común Divisor
If l(1) <= l(2) Then
    If l(1) < l(2) Then 'Cambio
        xx = x(1): x(1) = x(2): x(2) = xx
        ll = l(1): l(1) = l(2): l(2) = ll
    Else
        If x(1) < x(2) Then 'Cambio
            xx = x(1): x(1) = x(2): x(2) = xx
            ll = l(1): l(1) = l(2): l(2) = ll
        End If
    End If
End If
Do
    If l(1) <= 2*n And l(2) <= 2*n Then
        q0 = Val(x(1)): r0 = Val(x(2))
        Do
            q1 = q0 / r0
            q1 = Int(q1)
            r1 = q0 - q1 * r0
            If r1 = 0 Then Exit Do
            q0 = r0: r0 = r1
        Loop
        rr0 = Mid$(Str(r0), 2)
        MaxComDiv = rr0
        Exit Function
    Else
        y0 = x(2)
        rr() = DivisionEuclidea(x(), n)
        If rr(2) = "0" Then
            rr0 = y0
            MaxComDiv = rr0
            Exit Function
        Else
            x(1) = y0: l(1) = Len(x(1))
            x(2) = rr(2): l(2) = Len(x(2))
        End If
    End If
Loop
End Function

'=====

Public Function MinComMult(ByRef xa() As String, ByVal n As Integer) As String
    Dim i As Integer, x(2) As String, rr0 As String, x2 As String, rr() As String
    For i = 1 To 2: x(i) = xa(i): Next i
    '----- RutinaSigno
    For i = 1 To 2
        If left$(x(i),1) = " " then x(i) = Mid$(x(i), 2)
        If Left$(x(i), 1) = "-" Then x(i) = Mid$(x(i), 2)
    
```

```

Next i
'Mínimo Común Múltiplo
rr0 = MaxComDiv(x(), n) 'MaximoComunDivisor
If Len(x(1)) <= 2*n Then
    x(1) = Mid$(Str$(Val(x(1)) / Val(rr0)), 2)
Else
    x2 = x(2): x(2) = rr0
    rr() = DivisionEuclidea(x(), n)
    x(1) = rr(1): x(2) = x2
End If
MinComMult = Multiplicar(x(), n)
End Function
'=====

Public Function FormacionBloques(xx As String, lx As Double, ByVal n As Integer) As Variant
'----- FORMACIÓN DE LOS BLOQUES
Dim l1 As Single, k As Double, r() As Double
Dim li As Double, lm As Double, na As Integer
l1 = lx / n
If l1 = Int(l1) Then lm = l1 Else lm = Int(l1) + 1
ReDim r(lm)
na = n
For k = 1 To lm
    li = lx - k * n + 1
    If li < 1 Then
        na = na - (1 - li)
        li = 1
    End If
    r(k) = Val(Mid$(xx, li, na))
Next k
FormacionBloques = r()
End Function
'=====

Public Function Desbloqueo(ByRef a() As Double, ByVal da As Integer, ByVal n As Integer) As String
'-----DESBLOQUEO
Dim la As Integer, lp As Integer, lr As Double, w As Integer, at As String
Dim aa As String, a1 As String, a2 As String, k As Integer, i As Integer, bt As String
For k = 1 To da
    aa = Str$(a(k)): la = Len(aa)
    a1 = Right$(aa, n)
    a(k) = Val(a1)
    If la > n Then
        a2 = Left$(aa, la - n)
        a(k + 1) = a(k + 1) + Val(a2)
    End If
Next k
w = da + 1
Do
    If a(w) <> 0 Then Exit Do
    If w = 0 Then
        Desbloqueo = "0"
        Exit Function
    End If
    w = w - 1
Loop
For k = w To 1 Step -1

```

```

aa = Str$(a(k)): lp = Len(aa) - 1
For i = 1 To n - lp
    at = at + "0"
Next i
at = at + Right$(aa, lp)
Next k
Do Until Mid$(at, 1, 1) <> "0"
    at = Mid$(at, 2)
Loop
lr = Len(at)
Desbloqueo = Mid$(at, 1, lr)
End Function

```

Si la variable x es de precisión doble, $p = \text{len}(x)$ es el número de los bytes que ocupa la variable en la memoria y entonces $2p$ es el número máximo de cifras que se puede almacenar en ella sin pasar a la representación en punto flotante. En las funciones anteriores el valor de n se puede establecer de la manera siguiente:

```

Option explicit
Public n As Integer, test As Double
' =====
n = len(test) - 1

```

Este tamaño n de los bloques vale para todas las operaciones a efectuar. Por ejemplo, la multiplicación de los números $x(1)$ y $x(2)$ se podría hacer por el código siguiente:

```

Dim Producto As String
Producto = Multiplicar(x(), n)

```

En el caso de la división entera de los enteros $x(1)$ y $x(2)$ el código es

```

Dim CR() As String, Cociente As String, Resto As String
CR() = DivisionEuclidea(x(), n)
Cociente = CR(1): Resto = CR(2)

```

Obviamente, este código es muy útil al escribir programas para el cálculo del máximo común divisor y del mínimo común múltiplo de polinomios con coeficientes enteros. En estos cálculos, si los grados de los polinomios no son pequeños, pueden aparecer enteros grandes.

En el anillo $\mathbb{Z}[X]$, de los polinomios con coeficientes enteros, no se puede aplicar el algoritmo de Euclides, puesto que este anillo no es euclideo. Sin embargo, el anillo $\mathbb{Q}[X]$, de los polinomios con coeficientes racionales, ya es euclideo y los polinomios de $\mathbb{Z}[X]$ pueden ser considerados como elementos de $\mathbb{Q}[X]$. En $\mathbb{Q}[X]$ el máximo común divisor y el mínimo común múltiplo de dos polinomios está determinado hasta un factor inversible de este anillo, es decir, hasta un factor numérico de \mathbb{Q} . Si el máximo común divisor o el mínimo común múltiplo de dos polinomios con coeficientes enteros, calculado en $\mathbb{Q}[X]$, no tiene coeficientes enteros, se puede multiplicarlo con el mínimo común múltiplo de los denominadores de sus coeficientes y así se obtiene un máximo común divisor o un mínimo común múltiplo con coeficientes enteros.

Observación: Al calcular el máximo común divisor de dos polinomios de $\mathbb{Q}[X]$ con coeficientes enteros por el algoritmo de Euclides, y teniendo en cuenta que el máximo común divisor está determinado hasta un factor inversible, se puede demostrar que, al

efectuar las divisiones en el algoritmo de Euclides, se puede multiplicar el dividiendo o los restos parciales o finales con elementos no nulos de \mathbb{Z} , para evitar las operaciones con fracciones.

Sean, por ejemplo

$$P(X) = 2X^4 + 3X^3 + 4X^2 + X - 2 \text{ y } Q(X) = 3X^3 + 2X^2 + 5X - 2$$

, dos polinomios con coeficientes enteros de $\mathbb{Q}[X]$.

A continuación, se presentará el cálculo del máximo común divisor de ellos de dos maneras, trabajando con fracciones o evitándolas. Efectuando las divisiones de manera exacta en $\mathbb{Q}[X]$, los cálculos son los siguientes:

$$\begin{array}{r|l}
 \begin{array}{rrrrr}
 2X^4 & + & 3X^3 & + & 4X^2 & + & X & - & 2 \\
 -2X^4 & - & 4/3X^3 & - & 10/3X^2 & + & 4/3X & & \\
 \hline
 0 & 5/3X^3 & + & 2/3X^2 & + & 7/3X & & - & 2 \\
 & - & 5/3X^3 & - & 10/9X^2 & - & 25/9X & + & 10/9 \\
 \hline
 & 0 & - & 4/9X^2 & - & 4/9X & & - & 8/9
 \end{array} & \begin{array}{rrrr}
 3X^3 & + & 2X^2 & + & 5X & - & 2 \\
 \hline
 2/3X & + & 5/9 & & & &
 \end{array} \\
 \hline
 \begin{array}{rrrrr}
 3X^4 & + & 2X^2 & + & 5X & - & 2 \\
 -3X^3 & - & 3X^2 & - & 6X & & \\
 \hline
 0 & - & X^2 & - & X & - & 2 \\
 & & X^2 & + & X & + & 2 \\
 \hline
 & 0 & 0 & 0 & & &
 \end{array} & \begin{array}{rrrr}
 -4/9 X^2 & - & 4/9 X & - & 8/9 \\
 \hline
 -27/4 X & + & 9/4 & &
 \end{array}
 \end{array}$$

La segunda manera evitará los cálculos con fracciones:

$$\begin{array}{r|l}
 \begin{array}{rrrrr}
 2X^4 & + & 3X^3 & + & 4X^2 & + & X & - & 2 \\
 6X^4 & + & 9X^3 & - & 12X^2 & + & 3X & - & 6 \\
 -6X^4 & - & 4X^3 & - & 10X^2 & + & 4X & & \\
 \hline
 0 & 5X^3 & + & 2X^2 & + & 7X & - & 6 \\
 & 15X^3 & + & 6X^2 & + & 21X & - & 18 \\
 & - & 15X^3 & - & 10X^2 & - & 25X & + & 10 \\
 \hline
 & 0 & - & 4X^2 & - & 4X & - & 8 \\
 & & X^2 & + & X & + & 2
 \end{array} & \begin{array}{rrrr}
 3X^3 & + & 2X^2 & + & 5X & - & 2 \\
 \hline
 2X & + & 5 & & & &
 \end{array}
 \end{array}$$

$$\begin{array}{r|rrr}
 3X^3 & +2X^2 & +5X & -2 \\
 -3X^3 & -3X^2 & -6X & \\
 \hline
 0 & -X^2 & -X & -2 \\
 & X^2 & +X & +2 \\
 & 0 & 0 & 0
 \end{array}
 \quad
 \begin{array}{r|rrr}
 & X^2 & +X & +2 \\
 \hline
 & 2X & +9 &
 \end{array}$$

Se observa que la diferencia entre los dos resultados es un factor inversible:

$$-\frac{4}{9}X^2 - \frac{4}{9}X - \frac{8}{9} = -\frac{4}{9}(X^2 + X + 2)$$

, y así, los dos resultados son correctos. La ventaja del segundo método consiste en que no hay que trabajar con fracciones y por tanto, se pueden utilizar las funciones expuestas anteriormente para operara con números grandes. El código siguiente (en Visual.-Basic) utiliza el segundo método para el cálculo del máximo común divisor de dos polinomios con coeficientes enteros:

```

Public Function CalculoMCDPOL(ByRef c1() As String, ByRef c2() As String, ByVal n As Integer) As Variant
' ---- CÁLCULO DE M.C.D. DE LOS POLINOMIOS
Dim i As Integer, k As Integer, gz As Integer, rr() As String, rr0 As String
Dim s1 As Integer, sw2 As Integer, jr As Integer, p1() As String, z() As String
Dim mp As String, m11 As String, ya As String, ym As String, m2 As String, x(2) As String
g1 = UBound(c1()): g2 = UBound(c2())
ReDim z(g2)
Do
    gz = 0: s1 = 0
    For i = 0 To g1 - g2
        If c1(i) <> "0" Then
            x(1) = c1(i): x(2) = c2(0)
            x(1) = MinComMult(x(), n)
            x(2) = c1(i)
            rr() = DivisionEuclidea(x(), n)
            m11 = rr(1)
            If Left$(m11, 1) = "-" Then
                m11 = Mid$(m11, 2)
            End If
            If m11 <> "1" Then
                For k = i To g1
                    x(1) = c1(k): x(2) = m11
                    c1(k) = Multiplicar(x(), n)
                Next k
            End If
            x(1) = c1(i): x(2) = c2(0)
            rr() = DivisionEuclidea(x(), n)
            m2 = rr(1)
            For k = i To i + g2
                x(1) = c2(k - i): x(2) = m2
                x(2) = Multiplicar(x(), n)
                x(1) = c1(k)
                c1(k) = Restar(x(), n)
            Next k
        End If
    Next i
    jr = g1 - g2 + 1: j = jr
    For i = jr To g1
        If c1(i) = "0" Then

```

```

        If s1 = 1 Then
            z(i - j) = c1(i): gz = gz + 1
            If s1 = 0 Then s1 = s1 + 1
        Else
            j = j + 1
        End If
    Else
        z(i - j) = c1(i): gz = gz + 1
        If s1 = 0 Then s1 = s1 + 1
    End If
Next i
sw2 = 0
For i = 0 To gz - 1
    If z(i) <> "0" Then
        ReDim p1(gz - 1)
        For j = 0 To gz - 1: p1(j) = z(j): Next j
        c1() = c2(): c2() = p1()
        g1 = UBound(c1()): g2 = UBound(c2())
        c2() = RutinaMCDPOL(c2(), n)
        sw2 = 1: Exit For
    End If
Next i
If sw2 = 0 Then Exit Do
Loop
c2() = RutinaMCDPOL(c2(), n)
CalculoMCDPOL = c2()
End Function

' =====

Public Function RutinaMCDPOL(ByRef c2x() As String, ByVal n As Integer) As Variant
    Dim i As Integer, g2 As Integer, rr0 As String, rr() As String, x(2) As String
    g2 = UBound(c2x()): rr0 = c2x(0)
    If Left$(rr0, 1) = "-" Then rr0 = Mid$(rr0, 2)
    For i = 1 To g2
        If c2x(i) <> "0" Then
            x(1) = rr0: x(2) = c2x(i)
            rr0 = MaxComDiv(x(), n)
            If rr0 = "1" Then Exit For
        End If
    Next i
    If rr0 <> "1" Then
        For i = 0 To g2
            x(1) = c2x(i): x(2) = rr0
            rr() = DivisionEuclidea(x(), n)
            c2x(i) = rr(1)
        Next i
    End If
    If Left$(c2x(0), 1) = "-" Then
        For i = 0 To g2
            If c2x(i) <> "0" Then
                If Left$(c2x(i), 1) = "-" Then
                    c2x(i) = Mid$(c2x(i), 2)
                Else
                    c2x(i) = "-" + c2x(i)
                End If
            End If
        Next i
    End If
    RutinaMCDPOL = c2x()
End Function

```

'=====

Para hallar el mínimo común múltiplo de dos polinomios, tras haber calculado su máximo común divisor, hay que dividir el producto de los polinomios con el máximo común divisor.

```
Public Function ProductoPolinomios(ByRef c1() As String, ByRef c2() As String, ByVal n As Integer) As Variant
    ' --- CÁLCULO DEL PRODUCTO DE LOS POLINOMIOS
    Dim i As Integer, j As Integer, pp() As String, g1 As Integer, g2 As Integer
    Dim x(2) As String
    g1 = UBound(c1()): g2 = UBound(c2())
    ReDim pp(g1 + g2)
    For i = 0 To g1
        For j = 0 To g2
            If c1(i) = "0" Then Exit For
            If c2(j) <> "0" Then
                x(1) = c1(i): x(2) = c2(j)
                x(1) = Multiplicar(x(), n)
                x(2) = pp(i + j)
                pp(i + j) = Sumar(x(), n)
            End If
        Next j
    Next i
    ProductoPolinomios = pp()
End Function
```

'=====

La división del producto de dos polinomios con coeficientes enteros con su máximo común divisor (con coeficientes enteros) se efectuará mediante una división especial, para que el resultado tenga coeficientes enteros. El cociente exacto y el especial difieren en un factor numérico racional. Para el cálculo del cociente especial se puede utilizar la función siguiente:

```
Public Function DivisionEspecialPOL(ByRef pp() As String, c2() As String, ByVal n As Integer) As Variant
    '----- División exacta del producto de los polinomios por el M.C.D de ellos.
    Dim i As Integer, k1 As Integer, k2 As Integer, gpp As Integer, g2 As Integer
    Dim gm As Integer, ya As String, x(2) As String
    Dim mp As String, rq As String, w As String, qx As String, rr() As String
    Dim m1() As String, u() As String, rr0 As String, sw As Integer
    gpp = UBound(pp()): g2 = UBound(c2())
    gm = gpp - g2
    ReDim u(gm), m1(gm + 1)
    m1(gm + 1) = "1"
    For i = 0 To gm
        If pp(i) = "0" Then
            m1(i) = "1": u(i) = "0"
        Else
            x(1) = pp(i): x(2) = c2(0)
            x(1) = MinComMult(x(), n)
            x(2) = pp(i)
            rr() = DivisionEuclidea(x(), n)
            qx = rr(1)
            If Left$(qx, 1) = "-" Then
                qx = Mid$(qx, 2)
            End If
            m1(i) = qx
            If m1(i) <> "1" Then
                For k1 = i To gpp

```

```

                                x(1) = pp(k1): x(2) = m1(i)
                                pp(k1) = Multiplicar(x(), n)
                            Next k1
                        End If
                        x(1) = pp(i): x(2) = c2(0)
                        rr() = DivisionEuclidea(x(), n)
                        u(i) = rr(1)
                        For k1 = i To i + g2
                            x(1) = c2(k1 - i): x(2) = u(i)
                            x(2) = Multiplicar(x(), n)
                            x(1) = pp(k1)
                            pp(k1) = Restar(x(), n)
                        Next k1
                    End If
                Next i
                For k2 = 0 To gm
                    For i = 0 To k2
                        x(1) = u(i): x(2) = m1(k2 + 1)
                        u(i) = Multiplicar(x(), n)
                    Next i
                Next k2
                sw = 0: rr0 = u(0)
                For i = 1 To gm
                    If u(i) <> "0" Then
                        rr0 = MaxComDiv(x(), n)
                        If rr0 = "1" Then sw = 1: Exit For
                    End If
                Next i
                If sw = 0 And rr0 <> "1" Then
                    For i = 0 To gm
                        If u(i) <> "0" Then
                            x(1) = u(i): x(2) = rr0
                            u(i) = DivisionEuclidea(x(), n)
                        End If
                    Next i
                End If
                If Left$(u(0), 1) = "-" Then
                    For i = 0 To gm
                        If u(i) <> "0" Then
                            If Left$(u(i), 1) = "-" Then
                                u(i) = Mid$(u(i), 2)
                            Else
                                u(i) = "-" + u(i)
                            End If
                        End If
                    Next i
                End If
                DivisionEspecialPOL = u()
            End Function

```

La detección de los ceros de una función polinomio por el método del barrido tiene el problema que solo se pueden detectar los ceros de orden impar. Luego, el cálculo de un cero por el método de la bipartición solo es aplicable a los ceros de orden impar y la precisión de los cálculos disminuye cuando se trata de un cero múltiple. Si se divide el polinomio $P(X)$ (con coeficientes enteros) entre el máximo común divisor del polinomio y de su derivada (utilizando el procedimiento Function DivisionEspecialPol) se obtiene un polinomio $Q(X)$ que tiene los mismos ceros que $P(X)$, cuyos ceros son simples y detectables por el método del barrido y pueden ser calculados con gran precisión por el método de la bipartición. Con la ayuda de los ordenadores y lo expuesto anteriormente, hallar el polinomio $Q(X)$ no presenta dificultad alguna. Por tanto, en

vez de resolver la ecuación $P(X)=0$ es preferible resolver la ecuación $Q(X)=0$. Calculadas ya con gran precisión los ceros del polinomio $P(X)$ a partir del polinomio $Q(X)$, existen métodos para hallar las ordenes de multiplicidad de los ceros de $P(X)$. (Ver la monografía: Orden de multiplicidad de los ceros reales aproximados de un polinomio real, del mismo autor en Monografías.com)

Para calcular la derivada del polinomio $P(X)$ con coeficientes alfanuméricos se puede usar el código siguiente:

```
Public Function PolDerivado(ByRef p() As String, n As Integer) As Variant
    Dim i As Integer, gx As Integer, x(2) As String, xd() As String
    gx = UBound(p())
    ReDim xd(gx - 1)
    For i = 0 To gx - 1
        x(1) = Mid$(Str$(gx - i), 2): x(2) = p(i)
        xd(i) = Multiplicar(x(), n)
    Next i
    PolDerivado = xd()
End Function
```

Finalmente, para visualizar el polinomio $p()$ con coeficientes alfanuméricos en la caja de texto text1 de la Form1 (por ejemplo), hay que ejecutar la instrucción

Form1.Text1=FormatoPolinomio(P())

, donde la función FormatoPolinomio es la siguiente:

```
Public Function FormatoPolinomio(ByRef x() As String) As String
    Dim i As Integer, gx As Integer, ax As String, cx As String
    gx = UBound(x())
    For i = 0 To gx
        If x(i) <> "0" Then
            If i = 0 Then
                If x(0) <> "1" And x(0) <> "-1" Then
                    cx = x(0)
                Else
                    If gx <> 0 Then
                        If x(0) = "-1" Then
                            cx = "-"
                        End If
                    Else
                        If x(0) = "-1" Then
                            cx = Str$(-1)
                        Else
                            cx = Mid$(Str$(1), 2)
                        End If
                    End If
                End If
            End If
            If gx <> 0 Then
                If gx = 1 Then
                    cx = cx + " X"
                Else
                    cx = cx + " X^" + Mid$(Str$(gx), 2)
                End If
            End If
        Else
            If Left$(x(i), 1) = "-" Then
```

```

        ax = " - "
    Else
        ax = " + "
    End If
    End If
    If (x(i) <> "1" And x(i) <> "-1") Or i = gx Then
        If Left$(x(i), 1) = "-" Then
            ax = ax + Mid$(x(i), 2)
        Else
            ax = ax + x(i)
        End If
    End If
    If gx > 1 Then
        If i < gx - 1 Then
            ax = ax + " X^"
            ax = ax + Mid$(Str$(gx - i), 2)
        Else
            If i = gx - 1 Then
                ax = ax + " X"
            End If
        End If
    End If
    End If
    cx = cx + ax: ax = ""
End If
End If
Next i
FormatoPolinomio = cx
End Function

' =====

```

Observación: Teniendo en cuenta que el máximo común divisor y el mínimo común múltiplo de dos polinomios $P_1(X)$ y $Q_1(X)$ de $Q[X]$ está determinado hasta un factor inversible, la teoría anterior se puede aplicar también para estos polinomios con coeficientes fraccionarios o decimales, multiplicándoles con un entero elegido de manera apropiada, obteniendo los polinomios con coeficientes enteros $P_2(X)$ y $Q_2(X)$. En el caso cuando los coeficientes son fraccionarios, hay que multiplicar cada polinomio con el mínimo común múltiplo de los denominadores de los coeficientes. Cuando los coeficientes son decimales, hay que multiplicar cada polinomio con 10^s , donde s es el número máximo de los decimales en los coeficientes del polinomio. El máximo común divisor (mínimo común múltiplo) de $P_2(X)$ y $Q_2(X)$ será un máximo común divisor (un mínimo común múltiplo) con coeficientes enteros también para $P_1(X)$ y $Q_1(X)$.

Por ejemplo, si

$$P_1(X) = X^4 - \frac{17}{4}X^3 - \frac{33}{8}X^2 + \frac{7}{4}X + \frac{5}{8}$$

$$Q_1(X) = X^4 - \frac{7}{12}X^3 - \frac{17}{24}X^2 + \frac{5}{24}X + \frac{1}{12}$$

, entonces, en el ordenador se trabajará con los polinomios:

$$P_2(X) = 8X^4 - 34X^3 - 33X^2 + 14X + 5$$

$$Q_2(X) = 24X^4 - 14X^3 - 17X^2 + 5X + 2$$

Se obtiene que un máximo común divisor de $P_1(X)$ y $Q_1(X)$ es $8X^2 - 2X - 1$.

Si

$$P_1(X) = X^4 - 0.1X^3 - 4.06X^2 + 0.4X + 0.24$$

$$Q_1(X) = 0.1X^4 + 2.97X^3 - 1.3X^2 - 11.88X + 3.6$$

, entonces, para los cálculos en el ordenador se utilizarán los polinomios:

$$P_2(X) = 100X^4 - 10X^3 - 406X^2 + 40X + 24$$

$$Q_2(X) = 10X^4 + 297X^3 - 130X^2 - 1188X + 360$$

Luego un máximo común divisor de $P_1(X)$ y $Q_1(X)$ es $10X^3 - 3X^2 - 40X + 12$.

Al calcular, las permutaciones de p elementos, el producto de los números naturales desde p hasta q ($p < q$), las variaciones de p elementos tomados de k en k, las variaciones con repeticiones de p elementos de k en k, las combinaciones de p elementos de k en k, las combinaciones con repeticiones de p elementos tomados de k en k o las permutaciones con repeticiones de $m_0, m_1 \dots m_q$ elementos de tipos distintos, se usan las fórmulas

$$P_k = k!, \text{ ProductoPaQ} = p \cdot (p+1) \cdots (q-1) \cdot q, V_p^k = (p-k+1) \cdot (p-k) \cdots p$$

$$VR_p^k = p^k, C_p^k = \frac{V_p^k}{k!}, CR_p^k = C_{p+k-1}^k, PR = \frac{(m_0 + m_1 + \cdots + m_q)!}{m_1! m_2! \cdots m_q!}$$

, y se pueden obtener resultados grandes si los argumentos no son pequeños. Las funciones que se exponen a continuación, permiten calcular estas expresiones con exactitud. Sin embargo en el cálculo de las variaciones con repeticiones (potencias en definitiva) y de los factoriales hay que moderarse con el tamaño de los argumentos.

Public Function Factorial(ByRef p As String, ByVal n As Integer) As String

Dim i As String, y As String, x(2) As String

If Left\$(p, 1) = " " Then p = Mid\$(p, 2)

If Left\$(p, 1) = "-" Then p = Mid\$(p, 2)

If Left\$(p, 1) = "0" Then p = "0"

i = "1": y = "1"

If p = "0" Or p = "1" Then

Factorial = "1"

Else

Do

x(1) = i: x(2) = "1"

i = Sumar(x(), n)

x(1) = y: x(2) = i

y = Multiplicar(x(), n)

If i = p Then

Factorial = y: Exit Do

End If

Loop

End If

End Function

'=====

Public Function ProductoPaQ(ByRef p() As String, n As Integer)

'Se calcula el producto de los números naturales de p(1) a p(2) (p(1)<p(2))

```

Dim i As String, k As Integer, y As String, x(2) As String, dif As String
For k = 1 To 2
    If Left$(p(k), 1) = " " Then p(k) = Mid$(p(k), 2)
    If Left$(p(k), 1) = "-" Then p(k) = Mid$(p(k), 2)
Next k
x(1) = p(2): x(2) = p(1)
dif = Restar(x(), n)
If Left$(dif, 1) = "-" Then
    MsgBox "p(1) tiene que ser un entero menor que p(2)"
    Exit Function
End If
i = p(1): y = p(1)
Do
    x(1) = i: x(2) = 1
    i = Sumar(x(), n)
    x(1) = y: x(2) = i
    y = Multiplicar(x(), n)
    If i = p(2) Then
        ProductoPaQ = y: Exit Do
    End If
Loop
End Function

```

‘=====

```

Public Function Variaciones(ByVal p As String, ByVal k As String, ByVal n As Integer) As String
    Dim i As String, y As String, x(2) As String, rr As String, dif As String
    If Left$(p, 1) = " " Then p = Mid$(p, 2)
    If Left$(p, 1) = "-" Then p = Mid$(p, 2)
    If Left$(k, 1) = " " Then k = Mid$(k, 2)
    If Left$(k, 1) = "-" Then k = Mid$(k, 2)
    x(1) = p: x(2) = k
    dif = Restar(x(), n)
    If Left$(dif, 1) = "-" Then
        MsgBox "p tiene que ser un entero mayor o igual que k"
        Exit Function
    End If
    x(1) = p: x(2) = k
    x(1) = Restar(x(), n): x(2) = "1"
    x(1) = Sumar(x(), n): x(2) = p
    Variaciones = ProductoPaQ(x(), n)
End Function

```

‘=====

```

Public Function VariacionesConRepeticiones(ByVal p As String, ByVal k As String, ByVal n As Integer) As String
    Dim x(2) As String
    If Left$(p, 1) = " " Then p = Mid$(p, 2)
    If Left$(p, 1) = "-" Then p = Mid$(p, 2)
    If Left$(k, 1) = " " Then k = Mid$(k, 2)
    If Left$(k, 1) = "-" Then k = Mid$(k, 2)
    x(1) = p: x(2) = k
    VariacionesConRepeticiones = Potencias(x(), n)
End Function

```

‘=====

```

Public Function Combinaciones(ByVal p As String, ByVal k As String, ByVal n As Integer) As String
    Dim x(2) As String, Var As String, Fact As String, rr() As String, dif As String

```



```

If Left$(p, 1) = " " Then p = Mid$(p, 2)
If Left$(p, 1) = "-" Then p = Mid$(p, 2)
If Left$(k, 1) = " " Then k = Mid$(k, 2)
If Left$(k, 1) = "-" Then k = Mid$(p, 2)
x(1) = p: x(2) = k
dif = Restar(x(), n)
If Left$(dif, 1) = "-" Then
    MsgBox "p tiene que ser un entero mayor o igual que k"
    Exit Function
End If
If k = 0 Or k = p Then
    Combinaciones = "1"
    Exit Function
End If
If k = 1 Then
    Combinaciones = p
    Exit Function
End If
Var = Variaciones(p, k, n)
Fact = Factorial(k, n)
x(1) = Var: x(2) = Fact
rr() = DivisionEuclidea(x(), n)
Combinaciones = rr(1)
End Function

' =====

Public Function CombinacionesConRepeticiones(ByVal p As String, ByVal k As String, ByVal n As Integer) As String
    If Left$(p, 1) = " " Then p = Mid$(p, 2)
    If Left$(p, 1) = "-" Then p = Mid$(p, 2)
    If Left$(k, 1) = " " Then k = Mid$(k, 2)
    If Left$(k, 1) = "-" Then k = Mid$(p, 2)
    Dim x(2) As String, pp As String
    x(1) = p: x(2) = k
    pp = Sumar(x(), n)
    x(1) = pp: x(2) = "1"
    pp = Restar(x(), n)
    CombinacionesConRepeticiones = Combinaciones(pp, k, n)
End Function

' =====

Public Function PermutacionesConRepeticiones(ByRef m() As String, ByVal n As Integer) As String
    Dim x(2) As String, i As Long, Sum As String, fsum As Variant, q As Long, rr() As String
    q = UBound(m())
    For i = 1 To q
        If Left$(m(i), 1) = "-" Then m(i) = Mid$(m(i), 2)
        If Left$(m(i), 1) = " " Then m(i) = Mid$(m(i), 2)
    Next i
    Sum = m(0)
    For i = 1 To q
        x(1) = Sum: x(2) = m(i): Sum = Sumar(x(), n)
    Next i
    fsum = Factorial(Sum, n)
    For i = 0 To q
        x(1) = fsum: x(2) = Factorial(m(i), n)
        rr() = DivisionEuclidea(x(), n)
        fsum = rr(1)
    Next i
    PermutacionesConRepeticiones = fsum
End Function

```

'=====

También son útiles las dos funciones siguientes, que calculan

$$(2k+1)!! = 1 \cdot 3 \cdot 5 \cdot \dots \cdot (2k-1) \cdot (2k+1) \quad \text{y} \quad (2k)!! = 2 \cdot 4 \cdot 6 \cdot \dots \cdot (2k-2) \cdot (2k)$$

```
Public Function FactorialImpar(ByRef p As String, ByVal n As Integer) As String
    Dim i As String, y As String, x(2) As String, z As String
    z = Right$(p, 1)
    If z = "0" Or z = "2" Or z = "4" Or z = "6" Or z = "8" Then
        MsgBox "¡El argumento p tiene que ser un entero impar!"
        Exit Function
    End If
    If Left$(p, 1) = " " Then p = Mid$(p, 2)
    If Left$(p, 1) = "-" Then p = Mid$(p, 2)
    i = "1": y = "1"
    If p = "1" Then
        FactorialImpar = "1"
    Else
        Do
            x(1) = i: x(2) = "2"
            i = Sumar(x(), n)
            x(1) = y: x(2) = i
            y = Multiplicar(x(), n)
            If i = p Then
                FactorialImpar = y: Exit Do
            End If
        Loop
    End If
End Function
```

'=====

```
Public Function FactorialPar(ByRef p As String, ByVal n As Integer) As String
    Dim i As String, y As String, x(2) As String, z As String
    z = Right$(p, 1)
    If z = "1" Or z = "3" Or z = "5" Or z = "7" Or z = "9" Then
        MsgBox "¡El argumento p tiene que ser un entero par!"
        Exit Function
    End If
    If Left$(p, 1) = " " Then p = Mid$(p, 2)
    If Left$(p, 1) = "-" Then p = Mid$(p, 2)
    i = "0": y = "1"
    If p = "0" Then
        FactorialPar = "1"
    Else
        Do
            x(1) = i: x(2) = "2"
            i = Sumar(x(), n)
            x(1) = y: x(2) = i
            y = Multiplicar(x(), n)
            If i = p Then
                FactorialPar = y: Exit Do
            End If
        Loop
    End If
End Function
```

