Escuela Universitaria de Ingeniería Vitoria-Gasteiz
Ingeniaritzako Unibertsitate Eskola Vitoria-Gasteiz

Fundamentos de Informática

Examen – 29 de Junio de 2007

Resolución propuesta

1. *(1 punto)*

Base 2	Base 4	Base 7	Base 10	Base 16
111010	322	112	58	3a
010101	**111**	30	21	15
111001	321	**111**	57	39

2. *(2 puntos)*

a) Hay diversas posibilidades (todas válidas) a las hora de expresar las condiciones. Se proponen tres: con If-s anidados, con If-ElseIf-Else y con Select-Case.

```
Function CategoriaHuracan(ByVal v As Integer) As Integer
  If v < 119 Then
    CategoriaHuracan = 0
  Else
    If v < 154 Then
      CategoriaHuracan = 1
    Else
      If v < 178 Then
        CategoriaHuracan = 2
      Else
        If v < 210 Then
          CategoriaHuracan = 3
        Else
          If v < 250 Then
            CategoriaHuracan = 4
          Else
            CategoriaHuracan = 5
          End If
        End If
      End If
    End If
  End If
End Function
```

```
Function CategoriaHuracan(ByVal v As Integer) As Integer
  If v < 119 Then
    CategoriaHuracan = 0
  ElseIf v < 154 Then
    CategoriaHuracan = 1
  ElseIf v < 178 Then
    CategoriaHuracan = 2
  ElseIf v < 210 Then
    CategoriaHuracan = 3
  ElseIf v < 250 Then
    CategoriaHuracan = 4
  Else
    CategoriaHuracan = 5
  End If
End Function
```

```
Function CategoriaHuracan(ByVal v As Integer) As Integer
  Select Case v
    Case Is < 119
      CategoriaHuracan = 0
    Case 154 To 177 ' Es posible pero no necesario
      CategoriaHuracan = 1
    Case Is < 178
      CategoriaHuracan = 2
    Case Is < 210
      CategoriaHuracan = 3
    Case Is < 250
      CategoriaHuracan = 4
    Case Else
      CategoriaHuracan = 5
    End Select
End Function
```

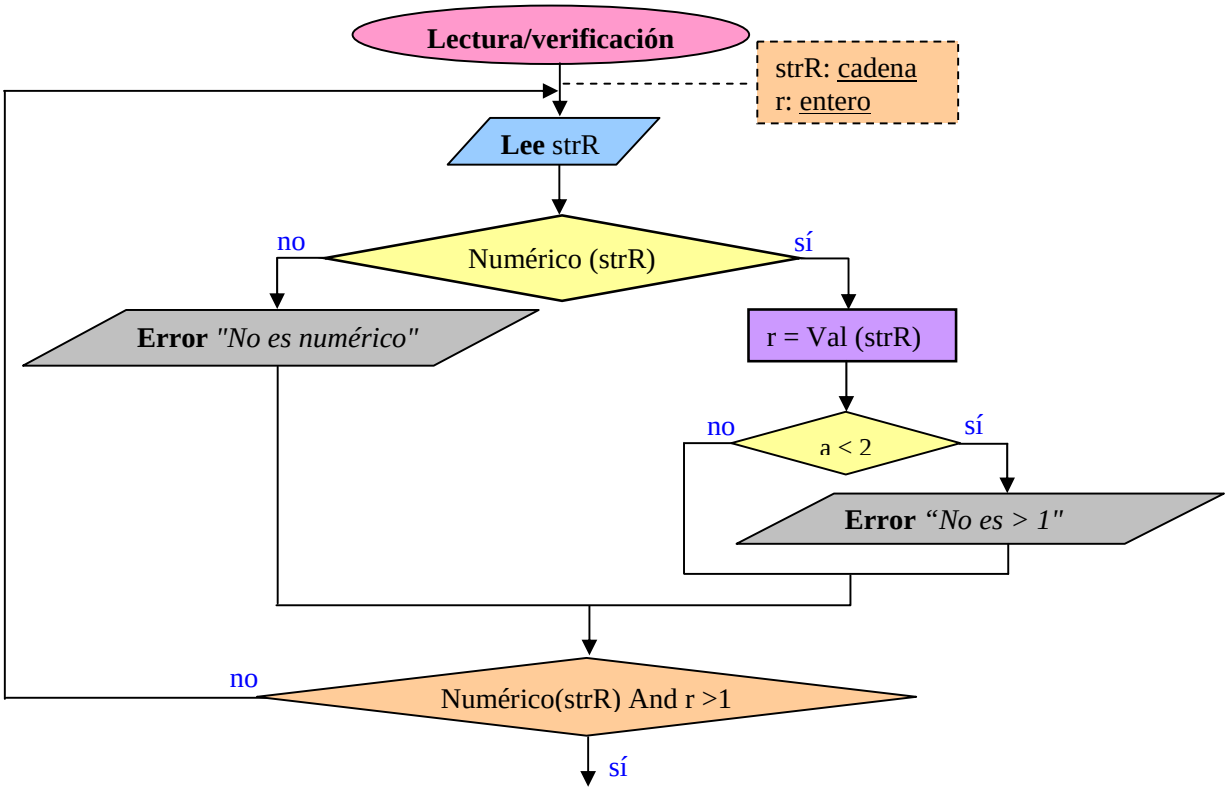
b) Hay diversas posibilidades (todas válidas) a las hora de expresar las condiciones. Como es la más diferente del anterior sólo se propone una, la del modelo Select-Case.

```
Sub DañosEstimados(ByVal c As Integer)
  Select Case c
    Case 0
      MsgBox ("No es huracán")
    Case 1
      MsgBox ("Daños mínimos")
    Case 2
      MsgBox ("Daños moderados")
    Case 3
      MsgBox ("Daños extensos")
    Case 4
      MsgBox ("Daños extremos")
    Case 5
      MsgBox ("Daños catastróficos")
    Case Else
      MsgBox ("Categoría incorrecta")
    End Select
End Sub
```

Nótese que el aspecto principal de este ejercicio es la especificación adecuada de los subprogramas, procedimiento (**Sub**) o función (**Function**), los parámetros de entrada, los parámetros de salida y el valor devuelto (cuando se trata de una función), cuando los haya. Son errores graves volver a declarar los parámetros de entrada, leer un valor que se supone viene dado en un parámetro de entrada o sacar por pantalla (escribir) en vez de devolver un valor.

3. *(4 puntos)*
Para simplificar el diagrama de flujo y su comprensión vamos a centrarnos primero en el diseño del algoritmo de lectura y verificación de un dato, por ejemplo, el la razón **`r`**. Para ello tenemos que leer una cadena con el dato, verificar que esa cadena representa un número y si es así verificar si el número cumple las restricciones de ser natural o mayor que 1.

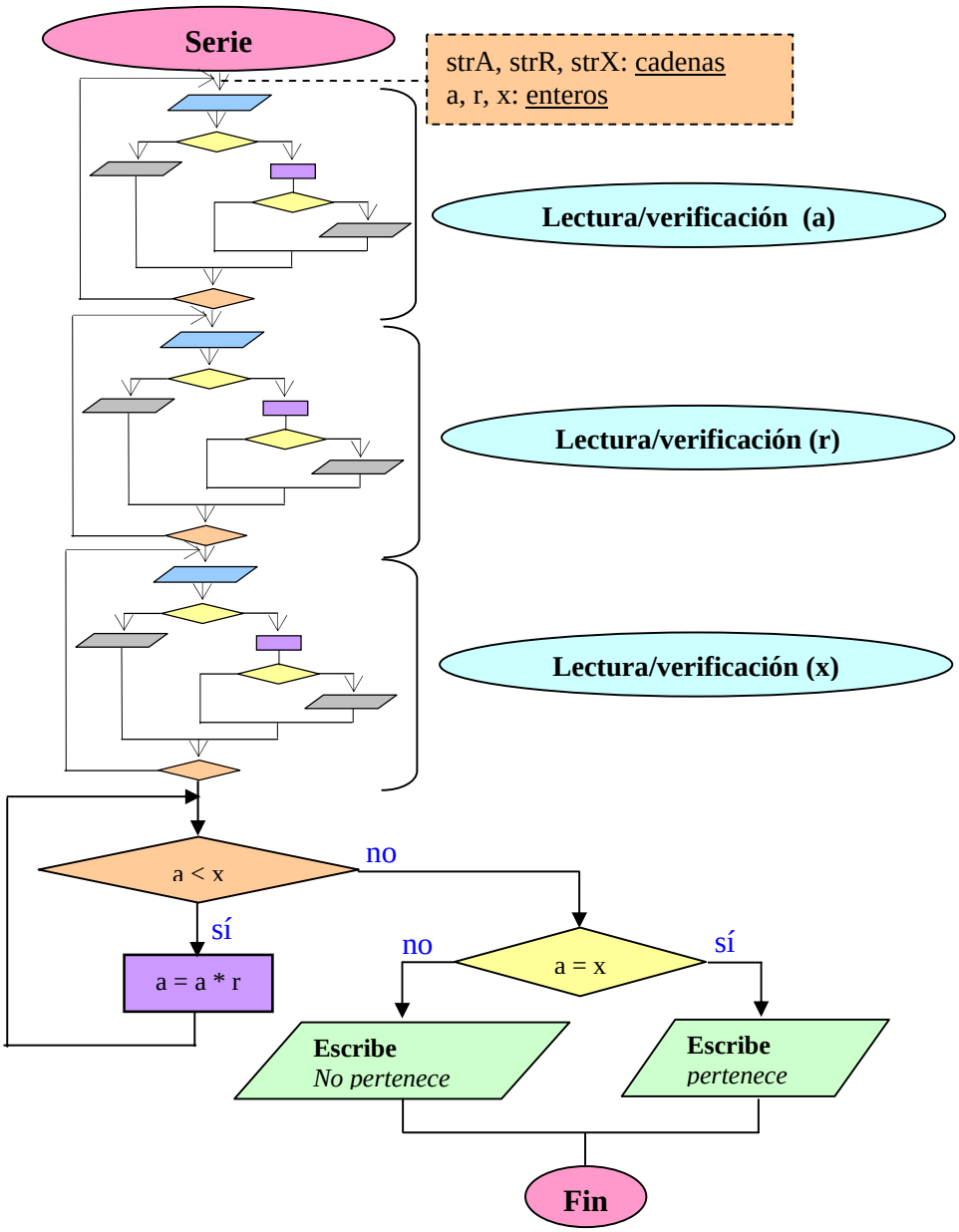
Si en vez de leer la cadena con el número pretendiéramos leer directamente a la variable entera r nos encontraríamos que al introducir un valor no numérico, por ejemplo, "B", tendríamos un error de ejecución y el programa no podría continuar. Nótese que la función **`IsNumeric`** recibe como parámetro una cadena.



Se puede observar que hemos optado por una estructura **`Do-Loop-Until`**. Puede utilizarse alternativamente una estructura **`Do-Loop-While`**, negando toda la condición final, es decir, en vez de "Numérico (strR) And r > 1" pondríamos "Not Numérico(strR) Or r <= 1".

Otra alternativa válida sería utilizar una estructura **`While`**, lo cual nos obligaría a duplicar la lectura, es decir, leer y mirar si cumple ambas condiciones de ser numérico y representar un valor válido.

El diagrama de flujo completo en el que se repite tres veces la misma estructura (con variables distintas y con valores válidos distintos, como se verá el programa final) sería el siguiente:



En la resolución del programa VB correspondiente a este diagrama de flujo hemos modificado la tercera verificación para utilizar una estructura **`Do-Loop-While`**, tal y como se describía anteriormente.

```
Private Sub Command2_Click()
  Dim sA As String, a As Integer
  Dim sR As String, r As Integer
  Dim sX As String, x As Integer

  Do
    sA = InputBox("Introduce el primer elemento a0")
    If IsNumeric(sA) Then
      a = Val(sA)
      If a < 0 Then
        MsgBox (a & " no es natural")
      End If
    Else
      MsgBox (sA & " no es numérico")
    End If
  Loop Until IsNumeric(sA) And a >= 0

  Do
    sR = InputBox("Introduce la razón r")
    If IsNumeric(sR) Then
      r = Val(sR)
      If r <= 1 Then
        MsgBox (r & " no es mayor que 1")
      End If
    Else
      MsgBox (sR & " no es numérico")
    End If
  Loop Until IsNumeric(sR) And r > 1

  Do
    sX = InputBox("Introduce el número x")
    If IsNumeric(sX) Then
      x = Val(sX)
      If x < 0 Then
        MsgBox (x & " no es un número natural")
      End If
    Else
      MsgBox (sX & " no es numérico")
    End If
  Loop While Not IsNumeric(sX) Or x < 0

  While a < x
    a = a * r
  Wend

  If x = a Then
    MsgBox (x & " pertenece a la serie")
  Else
    MsgBox (x & " no pertenece a la serie")
  End If
 End Sub
```

4. *(3 puntos)*

```
Function Suma(ByVal s1 As String, ByVal s2 As String) As String
  Dim i1 As String, i2 As String
  Dim i As Integer, n As Integer
  Dim d1 As String, d2 As String
  Dim llevo As Integer
  Dim res As String
  Dim sum As Integer
  i1 = Invertir(s1)
  i2 = Invertir(s2)
  If Len(i1) > Len(i2) Then
    n = Len(i1)
  Else
    n = Len(i2)
  End If
  llevo = 0
  res = ""
  For i = 1 To n Step 1
    d1 = Mid(i1, i, 1)
    d2 = Mid(i2, i, 1)
    sum = Val(d1) + Val(d2) + llevo ' Val("") devuelve 0
    res = sum Mod 10 & res
    llevo = sum \ 10
  Next i
  If llevo = 1 Then
    res = "1" & res
  End If
  Suma = res
End Function
```

Si se desea probar esta función, una posible codificación de la función `Invertir` (que suponíamos proporcionada) y un botón que lo invoca es:

```
Private Sub Command1_Click()
  Dim s1 As String, s2 As String
  s1 = InputBox("Introduce el primer número")
  s2 = InputBox("Introduce el segundo número")
  MsgBox (s1 & " + " & s2 & " = " & Suma(s1, s2))
End Sub
Function Invertir(ByVal s As String) As String
  Dim res As String
  Dim i As Integer, n As Integer
  res = ""
  n = Len(s)
  For i = 1 To n Step 1
    res = Mid(s, i, 1) & res
  Next i
  Invertir = res
End Function
```