make.tol de Rae.Lemario

Programa selector de palabras, de entre las contenidas en un lemario, por determinadas condiciones, por ejemplo, por ser palindromos, por contener todas las vocales o ser todas sus letras diferentes. Ejemplos de lemarios que este programa puede manejar son los de la Real Academia Española, que de sus siglas Rae este programa toma su nombre.

Este programa realiza diversas selecciones que proceden de diferentes aplicaciones o necesidades como juegos, ejercios de lengua, localizaciones de palabras por su terminación, para publicaciones, de lemas que cumplan ciertas restricciones, etc. Rae.Lemario se presenta junto con un lemario reducido de 8.025 palabras curiosas de prueba, pero ha sido ejecutado con lemarios mas grandes de hasta 95.746 palabras. Las versiones de Tol 1.1.1, 1.1.5, 1.1.6 y 2.0.1 pueden procesar el lemario de 8.025 palabras, pero solo las 3 ultimas uno de 95.746 palabras.

Rae.Lemario extrae palabras con los siguientes criterios: a) Palabras con determinadas terminaciones de una o mas letras, por ejemplo, palabras terminadas en j o en k como reloj o anorak. b) Palindromos, programados de 2 formas diferentes como reconocer o rezar. c) Palabras que contienen todas las vocales a, e, i, o y u, una sola vez, sin repeticion de ninguna de las 5 vocales, por ejemplo, abrenuncio. d) Palabras que tienen tienen todas las vocales una o mas veces, esto es, con repeticion, por ejemplo, albaricoque que tiene las 5 vocales, pero 2 aes. e) Pares de palabras que una son una la simetrica de otra, como por ejemplo, orar y raro f) Palabras especialmente largas, por ejemplo, antirreglamentario. g) Palabras que empiezan y terminan por las mismas letras, de manera que puedan formar un circulo, como aderezada, que empieza con ad y termina en da. h) Palabras que tienen un numero alto de grafemas, acentos, virgulillas, diéresis, puntos de las ies, como por ejemplo, sociolingüístico o pedigüeñería. i) Con muchas mas vocales que consonantes, por ejemplo, auxilio. j) Con muchas mas consonantes que las vocales, como, brillantez, k) Conjuntos de palabras que tienen las mismas letras, por ejemplo, serrato, retraso, terrosa, arresto, sortera, ostrera, asertor, sortear, rastreo y trasero. l) Palabras que tienen todas sus caracteres diferentes, como culteranismo. m) Palabras que tienen todos sus letras 2 veces, bien de forma estricta como el caso de adorador o sin ser tan estricto, por ejemplo con los acentos como es el caso de allá. n) Palabras que tienen una letra 1 vez, otra letra 2 veces, otra letra 3 veces, otra letra 4 veces y asi tantas como suficientes letras tenga la palabra, bien de forma estricta como telele, con 1 t, 2 eles y 3 es o de forma no tan estricta con los acentos como tacatá, con 1 c, 2 tes y 3 aes si bien una esta acentuada.

En este programa se puede observar como se puede en lenguaje Tol: a) Leer y escribir ficheros planos de texto con ReadFile(), WriteFile y AppendFile y a convertir esos textos en conjuntos con Tokenizer(). b) Realizar bifurcaciones con las funciones If() y Case(). c) Recorrer y evaluar funciones sobre conjuntos con EvalSet() y For(). d) Seleccionar determinados elementos de un conjunto con Select() o a hacer que todos sean diferentes con Unique(). e) Ordenar y clasificar conjuntos con Sort() y Classify(). f) Transponer conjuntos tabulares con Traspose(). g) Declarar funciones dentro de funciones, por ejemplo, la declaracion de la funcion local filPth() dentro de la funcion global LemEndAt() o la declaracion de la funcion local graCnt() que cuenta grafemas dentro de la funcion global LemGrapheme(). h) Pasar codigo Tol como parametro de entrada de otras funciones, ver por ejemplo la declaracion de la fumncion LemSelect() y su llamada desde la funcion LemCicle() y otras

declaracion de la funncion LemSelect() y su llamada desde la funcion LemCicle() y otras.

Las funciones de seleccion de este programa Rae.Lemario tienen diversos modos de funcionamiento, a veces seleccionables mediante parametros, como por ejemplo: a) la seleccion a partir de cierta longitud de la palabra, b) la distincion o no entre mayusculas y minusculas, c) la distincion o no entre vocales acentuadas o no acentuadas y con dieresis, etc. Esta parametrizacion no es general en todas las funciones que, a su vez, se pueden ejecutar o no mediante un If() de control. Finalmente, hay una funcion que puede ejecutarse a la terminacion que, con todas las selecciones realizadas por las funciones del programa, construye un nuevo lemario con todos aquellos terminos, del lemario de entrada, que cumplen al menos una de las caracteristicas seleccionadas, este fichero podria considerarse un lemario de palabras curiosas.

Árbol de ficheros

Rae.Lemario seleccion de palabras de un lemario por diferentes caracteristicas

- ← make.tol proceso con diferentes funciones de seleccion de palabras
- ← make.bat mandato de ejecucion del programa de seleccion de palabras
- lemario.inp directorio de lemarios de entrada para seleccion con origen la RAE
  - ← lemario.curioso.txt lemario ejemplo de entrada, mas de 8.000 palabras/lemas curiosos
- lemario.out directorio de ficheros con palabras seleccionadas del lemario entrada
  - → aeiou.con.repeticion.txt palabras con 1 ocurrencia o mas de cada una de las vocales
  - → aeiou.sin.repeticion.txt palabras con 1 ocurrencia de cada una de las vocales
  - → empieza.termina.igual.txt palabras que empiezan y terminan por las mismas 2 letras
  - → letras.repetidas.en.secuencia.estricto.txt con 1 letra 1 vez, otra 2, otra 3,... distinguiendo acentos
  - → letras.repetidas.en.secuencia.no.estricto.txt con 1 letra 1 vez, otra 2, otra 3,... sin considerar acentos
  - → mas.consonantes.que.vocales.txt palabras que tienen un 70% o mas de consonantes que vocales
  - → mas.vocales.que.consonantes.txt palabras que tienen un 70% o mas de vocales que consonantes
  - → palabras.largas.txt palabras largas, en este caso, de mas de 17 letras
  - → palindroma.01.txt palabras que se leen igual a derechas y a izquierdas
  - → palindroma.02.txt palabras que se leen igual a derechas y a izquierdas
  - → palabras.simetricas.txt pares de palabras simetrica una de otra incluido palindromos
  - → termina.en.ab.txt palabras que terminan en ab

→ termina.en.b.txt palabras que terminan en b

→ termina.en.bab.txt palabras que terminan en bab

→ termina.en.c.txt palabras que terminan en c

→ termina.en.f.txt palabras que terminan en f

→ termina.en.g.txt palabras que terminan en g

→ termina.en.h.txt palabras que terminan en h

→ termina.en.j.txt palabras que terminan en j

→ termina.en.k.txt palabras que terminan en k

→ termina.en.m.txt palabras que terminan en m

→ termina.en.p.txt palabras que terminan en p

→ termina.en.t.txt palabras que terminan en t

→ termina.en.u.txt palabras que terminan en u

→ termina.en.v.txt palabras que terminan en v

→ termina.en.x.txt palabras que terminan en x

→ termina.en.y.txt palabras que terminan en y

→ tiene.4.grafemas.txt palabras que tiene 4 grafemas

→ tiene.5.grafemas.txt palabras que tiene 5 grafemas

→ tiene.6.grafemas.txt palabras que tiene 6 grafemas

→ tiene.las.mismas.letras.txt conjuntos de palabras que todas tienen las mismas letras

→ todas.las.letras.2.veces.estricto.txt palabras que tienen 2 veces cada letra de forma estricta

→ todas.las.letras.2.veces.no.estricto.txt palabras que tienen 2 veces cada letra con o sin acentos

→ todos.los.carecteres.diferentes.txt palabras largas en donde no se repite ninguna letra

→ rae_lemario.pdf documento de funciones de seleccion de lemas

Declaraciones

Constantes

- `Text DirInp`
  Directorio para los lemarios de entrada.
- `Text DirOut`
  Directorio para los lemarios de salida.
- `Text FilInp`
  Fichero de entrada con lemas, palabras.

- `Set LemInp`
  Conjunto de palabras, lemas, de entrada.

Funciones

- `Real LemAppend(Text filPth, Text lemTxt)`
  Retorna cierto y escribe la palabra lemTxt en una linea del fichero filPth.
- `Text LemLowCls(Text lemTxt)`
  Retorna un lema en minusculas y sin acentos.
- `Set LemChrSet(Text lemTxt)`
  Retorna el conjunto ordenado de las letras de un lema.
- `Text Lem2Line(Set lemSet)`
  Retorna un texto con un conjunto de lemas semarados por |.
- `Real LemEndAt(Set lemSet, Set endSet, Text filPat)`
  Para el conjunto de terminaciones endSet escribe tantos ficheros de salida como terminaciones y dentro de cada fichero las palabras que terminan en dicha terminacion. Escribe cada palabra en una linea del fichero. Los nombres de los ficheros son similares salvo que cada uno contiene la terminacion. Esta funciones diferencia las letras mayusculas de las minusculas y las acentuadas de las no acentuadas. Las terminaciones puede tener coincidencias, por ejemplo, n y on y con. Retorna el numero total de palabras encontradas, si una palabra coincide con varias terminaciones cuenta tantas veces como coincidencias.
- `Real LemPalindrome01(Set lemSet, Real minChr, Text filNam)`
  Escribe en el fichero de salida filNam todas las palabras palindromas del conjunto de entrada lemSet que tengan minChr o mas letras. Escribe cada palabra en una linea del fichero. Esta funcion diferencia las letras mayusculas de las minusculas y las acentuadas de las no acentuadas. Retorna el numero total de palabras palindromas encontradas. Se trata de una version programada de forma algo clasica y existe otra version programada de una forma mas natural en Tol.
- `Real LemPalindrome02(Set lemSet, Real minChr, Text filNam)`
  Selecciona de lemSet todas las palabras de mas de minChr letras e iguales a su Reverse() y las escribe de golpe en el fichero filNam añadiendo un salto de linea a cada palabra. Esta funcion diferencia las letras mayusculas de las minusculas y las acentuadas de las no acentuadas. Retorna el numero total de palabras palindromas encontradas. Se trata de una version programada en un estilo natural en Tol y existe otra version programada de una forma mas clasica.
- `Real LemAeiou(Set lemSet, Text sinFil, Text conFil)`
  Escribe 2 ficheros: a) en el primero todas las palabras que contienen todas las vocales, una sola vez, sin repeticion, b) y en el segundo las que las tienen una o mas veces, con repeticion. Escribe cada palabra en una linea del fichero. Las vocales da igual que esten en mayusculas, en minusculas, acentuadas o no acentuadas. Retorna el numero total de palabras encontradas de ambas categorias.
- `Real LemSimetricPair(Set lemSet, Text filNam)`

Selecciona de lemSet pares de palabras que una sea la simetrica de otra. Para ello contactena el conjunto de palabras con sus inversas, las clasifica por ser identicas y aquellos conjuntos con 2 ocurrencias indica que habia una simetria, esto se puede hacer porque en el lemario no hay repetidos. Omite las palabras del lemario que tengan guiones, usualmente son sufijos o prefijos. En esta seleccion se incluyen los palindromos que siempre son los simetricos de ellos mismos. Esta funcion diferencia las letras mayusculas de las minusculas y las acentuadas de las no acentuadas. Retorna el numero total de pares de palabras simetricas encontradas.

- `Real LemGELength(Set lemSet, Real minChr, Text filNam)`
  Selecciona de lemSet todas las palabras de mas de minChr letras y las escribe de golpe en el fichero filNam añadiendo un salto de linea a cada palabra. Retorna el numero total de palabras encontradas.

- `Real LemSelect(Set lemSet, Text filNam, Code funSel)`
  Generalizacion de las funciones anteriores que selecciona de lemSet palabras que cumplan una determinada funcion funSel(palabra) y las escribe de golpe en el fichero filNam añadiendo un salto de linea a cada palabra. Retorna el numero total de palabras encontradas.

- `Real LemCicle(Set lemSet, Real numChr, Text filNam)`
  Selecciona de lemSet todas las palabras que empiezan y terminan por las mismas letras, pero revertidas, de forma que puedan formar un circulo y las escribe en el fichero filNam. El numero de caracteres a comparar puede ser 1, 2, ... Utiliza la funcion de seleccion por palabras, una a una, LemSelect(). No selecciona por longitud y si considera diferentes las letras mayusculas de las minusculas y las acentuadas de las que no lo son. Retorna el numero total de palabras encontradas.

- `Real LemGrapheme(Set lemSet, Set numSet, Text filPat)`
  Escribe n ficheros cada uno con las palabras que tienen un numero de grafemas como el que se especifica en la lista numSet. Escribe cada palabra en una linea de cada fichero. Retorna el numero total de palabras encontradas en total.

- `Real LemRatio(Set lemSet, Real vocRat, Text vocFil, Text conFil)`
  Escribe 2 ficheros: a) el primero cuando las vocales son muchas mas que las consonantes y b) el segundo cuando las consonantes son muchas mas que las vocales. Para realizar esta distincion emplea un ratio de vocales sobre el total de las letras, por ejemplo el 60% de vocales, 0.6. Escribe cada palabra en una linea de cada fichero. Retorna el numero total de palabras encontradas de un tipo y del otro.

- `Real LemEquChr(Set lemSet, Real minRep, Text filNam)`
  Selecciona de lemSet conjuntos de lemas con las mismas letras y escribe en el fichero de salida aquellos con minRep o mas elementos. Para estas repeticiones distingue mayusculas de minusculas y acentos. Retorna el numero total conjuntos de palabras encontrados.

- `Real LemAllDifChr(Set lemSet, Real minChr, Text filNam)`

Selecciona de lemSet todas las palabras que tienen todas sus caracteres diferentes y igual o mas numCrh letras y las escribe en el fichero filNam. Utiliza la funcion de seleccion por palabras, una a una, LemSelect(). Considera diferentes las letras mayusculas de las minusculas y las acentuadas de las que no lo son. Pasando a minusculas y eliminado acentos puede hacerse la funcion equivalente que considere que, por ejemplo, la e acentuada es igual que la e. Retorna el numero total de palabras encontradas.

- `Real LemTwice(Set lemSet, Real casSen, Text filNam)`
  Selecciona de lemSet todas las palabras que tienen todos sus letras 2 veces. Utiliza la funcion de seleccion por palabras, una a una, LemSelect(). Dependiendo de casSen: a) Si es falso entonces considera iguales las letras mayusculas de las minusculas y las acentuadas iguales a las no acentudas. b) Si es cierto diferencia las letras mayusculas de minusculas y las acentuadas de las vocales sin acento. Retorna el numero total de palabras encontradas.

- `Real LemSec123(Set lemSet, Real casSen, Text filNam)`
  Selecciona de lemSet todas las palabras que tienen 1 letra 1 vez, otra 2 veces, otra 3 veces, otra 4 veces y asi tantas como letras haya. Utiliza la funcion de seleccion por palabras, una a una, LemSelect(). Dependiendo de casSen: a) Si es falso entonces considera iguales las letras mayusculas de las minusculas y las acentuadas iguales a las no acentudas. b) Si es cierto diferencia las letras mayusculas de minusculas y las acentuadas de las vocales sin acento. Retorna el numero total de palabras encontradas.

- `Real LemJoin(Text filNam)`
  Recupera todas los lemas, las palabras, seleccionadas por sus diferentes caracteristicas, palindromos, simetricas unas de otras, largas, con letras repetidas 2 veces, etc. y crea en filNam un nuevo lemario de palabras curiosas. Retorna el numero total de palabras encontradas.

Proceso

- `Real makEnd`
  Localiza y guarda palabras con cierta terminacion.

- `Real makP01`
  Localiza y guarda palabras palindromas, version 1.

- `Real makP02`
  Localiza y guarda palabras palindromas, version 2.

- `Real mak_5v`
  Localiza y guarda palabras con todas las vocales.

- `Real makSim`
  Localiza y guarda pares de palabras simetricas.

- `Real makLen`
  Localiza y guarda lemas de cierta longitud o mas.

- `Real makCic`
  Lemas que empiezan y terminan con la misma letra.

- `Real makGra`
  Localiza y guarda lemas con varios grafemas.

- `Real makRat`

Con mas vocales que consonantes o viceversa.

- `Real makEqu`
  Busca y escribre palabras con las mismas letras.
- `Real makDif`
  Palabras con todos sus caracteres diferentes.
- `Real makTwi`
  Palabras con todas sus letras 2 veces.
- `Real mak123`
  1 letra repetida 1 vez, otra 2, otra 3, otra 4, ...
- `Real makJoi`
  Une sin repeticion los lemas curiosos encontrados

Constantes

Text DirInp

```
////////////////////////////////////////////////////////////////////////////
Text DirInp = "lemario.inp";
////////////////////////////////////////////////////////////////////////////
PutDescription("Directorio para los lemarios de entrada.", DirInp);
////////////////////////////////////////////////////////////////////////////
```

Text DirOut

```
////////////////////////////////////////////////////////////////////////////
Text DirOut = "lemario.out";
////////////////////////////////////////////////////////////////////////////
PutDescription("Directorio para los lemarios de salida.", DirOut);
////////////////////////////////////////////////////////////////////////////
```

Text FilInp

```
////////////////////////////////////////////////////////////////////////////
Text FilInp = "lemario.curioso.txt";
////////////////////////////////////////////////////////////////////////////
PutDescription("Fichero de entrada con lemas, palabras.", FilInp);
////////////////////////////////////////////////////////////////////////////
```

Set LemInp

```
////////////////////////////////////////////////////////////////////////////
Set LemInp = Select(Tokenizer(ReadFile(DirInp+"/"+FilInp), "\n"),
                    Real(Text lemTxt) { And(Compact(lemTxt)!="",
                                            !TextFind(lemTxt,"-"),
                                            !TextFind(lemTxt,"?")) });
////////////////////////////////////////////////////////////////////////////
PutDescription("Conjunto de palabras, lemas, de entrada.", LemInp);
////////////////////////////////////////////////////////////////////////////
```

Real LemAppend()

```
////////////////////////////////////////////////////////////////////////////
Real LemAppend(Text filPth, // Ruta de un fichero
               Text lemTxt) // Palabra a escribir
////////////////////////////////////////////////////////////////////////////
{ Text AppendFile(filPth, lemTxt+"\n"); TRUE };
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna cierto y escribe la palabra lemTxt en una linea del fichero filPth.",
LemAppend);
////////////////////////////////////////////////////////////////////////////
```

Text LemLowCls()

```
////////////////////////////////////////////////////////////////////////////
Text LemLowCls(Text lemTxt) // Texto de entrada
////////////////////////////////////////////////////////////////////////////
{
    ReplaceTable(ToLower(lemTxt),
                 [[ [["á", "a"]], [["é", "e"]], [["í", "i"]], [["ó", "o"]],
                    [["ú", "u"]], [["ü", "u"]] ]])
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna un lema en minusculas y sin acentos.",
LemLowCls);
////////////////////////////////////////////////////////////////////////////
```

Set LemChrSet()

```
////////////////////////////////////////////////////////////////////////////
Set LemChrSet(Text lemTxt) // Texto de entrada
////////////////////////////////////////////////////////////////////////////
{
  Set  chrSet = For(1, TextLength(lemTxt), Text(Real posTxt)
                    { Sub(lemTxt, posTxt, posTxt) }); // Letras de la palabra
  Sort(chrSet, Real(Text a, Text b) { Compare(a,b) }) // Orden alfabetico
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna el conjunto ordenado de las letras de un lema.",
LemChrSet);
////////////////////////////////////////////////////////////////////////////
```

Text Lem2Line()

```
////////////////////////////////////////////////////////////////////////////
Text Lem2Line(Set lemSet) // Conjunto de entrada
////////////////////////////////////////////////////////////////////////////
{
  Real lemCrd = Card(lemSet);
  Case(
    lemCrd == 0, "",          // Si conjunto vacio -> tira vacia
    lemCrd == 1, lemSet[1],   // 1 lema se pone tal cual
    TRUE,                     // 2 o mas
    {
      Text lemFst = lemSet[1]; // El primero
      Set  lemCic = For(2, lemCrd, Text(Real lemPos)
                        { " | " + lemSet[lemPos] }); // Lemas separados por |
```

```
      lemFst + SetSum(lemCic)
    })
};
////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna un texto con un conjunto de lemas semarados por |.",
Lem2Line);
////////////////////////////////////////////////////////////////////////
```

Real LemEndAt()

```
////////////////////////////////////////////////////////////////////////
Real LemEndAt(Set  lemSet, // Conjunto de lemas de entrada
              Set  endSet, // Conjunto de terminaciones
              Text filPat) // Patron de fichero de salida
////////////////////////////////////////////////////////////////////////
{
  // Retorna la ruta de un fichero de salida para la terminacion endTxt a
  // cambiando en el patron de nombre de fichero filPat _ por la terminación
  Text filPth(Text endTxt) { DirOut + "/" + Replace(filPat, "_", endTxt) };

  // Inicializa todos los ficheros de salida
  Set  EvalSet(endSet, Text(Text endTxt) { WriteFile(filPth(endTxt), "") });

  Set  lemCic = EvalSet(lemSet, Real(Text lemTxt) // Ciclo por palabras
  {
    // Ciclo terminaciones, pueden ocurrir varias
    Set  endCic = EvalSet(endSet, Real(Text endTxt)
    {
      If(! TextEndAt(lemTxt, endTxt), FALSE,
                                      LemAppend(filPth(endTxt), lemTxt))
    });
    SetSum(endCic) // Retorna el numero de terminaciones encontradas
  });
  SetSum(lemCic) // Retorna el numero de palabras encontradas
};
////////////////////////////////////////////////////////////////////////
PutDescription(
"Para el conjunto de terminaciones endSet escribe tantos ficheros de salida
como terminaciones y dentro de cada fichero las palabras que terminan en
dicha terminacion.
Escribe cada palabra en una linea del fichero.
Los nombres de los ficheros son similares salvo que cada uno contiene la
terminacion.
Esta funciones diferencia las letras mayusculas de las minusculas y las
acentuadas de las no acentuadas.
Las terminaciones puede tener coincidencias, por ejemplo, n y on y con.
Retorna el numero total de palabras encontradas, si una palabra coincide
con varias terminaciones cuenta tantas veces como coincidencias.",
LemEndAt);
////////////////////////////////////////////////////////////////////////
```

Real LemPalindrome01()

```
////////////////////////////////////////////////////////////////////////
Real LemPalindrome01(Set  lemSet, // Lemas de entrada para buscar palindromos
                     Real minChr, // Minimo de letras que ha de tener
                     Text filNam) // Nombre del fichero de salida
////////////////////////////////////////////////////////////////////////
{
  Text filPth = DirOut + "/" + filNam; // Ruta del fichero de salida
  Text WriteFile(filPth, ""); // Inicializa el fichero de salida

  Set  lemCic = EvalSet(lemSet, Real(Text lemTxt) // Ciclo por palabras
  {
    Real numChr = TextLength(lemTxt); // Longitud de la palabra
    If(numChr < minChr, FALSE, // Demasiado corta
```

```
    {
      // La mitad de la longitud. Si es impar la letra central es siempre
      // igual a ella misma y por eso el uso de la funcion Floor()
      Real midChr = Floor(numChr / 2);

      Text lftChr = Sub(lemTxt,          1, midChr);
      Text rghChr = Sub(Reverse(lemTxt), 1, midChr);

      If(lftChr != rghChr, FALSE,                    // No es palindromo
                           LemAppend(filPth, lemTxt)) // Es palindromo
    })
  });
  SetSum(lemCic) // Retorna el numero de palabras encontradas
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Escribe en el fichero de salida filNam todas las palabras palindromas del
conjunto de entrada lemSet que tengan minChr o mas letras.
Escribe cada palabra en una linea del fichero.
Esta funcion diferencia las letras mayusculas de las minusculas y
las acentuadas de las no acentuadas.
Retorna el numero total de palabras palindromas encontradas.
Se trata de una version programada de forma algo clasica y
existe otra version programada de una forma mas natural en Tol.",
LemPalindrome01);
////////////////////////////////////////////////////////////////////////////
```

Real LemPalindrome02()

```
////////////////////////////////////////////////////////////////////////////
Real LemPalindrome02(Set  lemSet, // Lemas de entrada para buscar palindromos
                     Real minChr, // Minimo de letras que ha de tener
                     Text filNam) // Nombre del fichero de salida
////////////////////////////////////////////////////////////////////////////
{
  Set  palSet = Select(lemSet, Real(Text lemTxt)
       { And(TextLength(lemTxt) >= minChr, lemTxt == Reverse(lemTxt)) });

  Text WriteFile(DirOut+"/"+filNam,
                 SetSum(EvalSet(palSet, Text(Text palTxt) { palTxt+"\n" })));

  Card(palSet)
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Selecciona de lemSet todas las palabras de mas de minChr letras e iguales a
su Reverse() y las escribe de golpe en el fichero filNam añadiendo un salto
de linea a cada palabra.
Esta funcion diferencia las letras mayusculas de las minusculas y
las acentuadas de las no acentuadas.
Retorna el numero total de palabras palindromas encontradas.
Se trata de una version programada en un estilo natural en Tol y
existe otra version programada de una forma mas clasica.",
LemPalindrome02);
////////////////////////////////////////////////////////////////////////////
```

Real LemAeiou()

```
////////////////////////////////////////////////////////////////////////////
Real LemAeiou(Set  lemSet, // Conjunto de lemas de entrada
              Text sinFil, // Fichero de salida sin repeticion
              Text conFil) // Fichero de salida con repeticion
////////////////////////////////////////////////////////////////////////////
{
  Text sinPth = DirOut + "/" + sinFil; // Ruta para aeiou sin repeticion
  Text conPth = DirOut + "/" + conFil; // Ruta para aeiou con repeticion

  Text WriteFile(sinPth, ""); // Inicializa el fichero sin repeticion
```

```
  Text WriteFile(conPth, ""); // Inicializa el fichero con repeticion

  Set  lemCic = EvalSet(lemSet, Real(Text lemTxt) // Ciclo por palabras
  {
    Text lemLow = LemLowCls(lemTxt); // En minuscula y limpia de acentos

    Real aCnt   = TextOccurrences(lemLow, "a");
    Real eCnt   = TextOccurrences(lemLow, "e");
    Real iCnt   = TextOccurrences(lemLow, "i");
    Real oCnt   = TextOccurrences(lemLow, "o");
    Real uCnt   = TextOccurrences(lemLow, "u");

    Case(

      And(EQ(aCnt,1), EQ(eCnt,1), EQ(iCnt,1), EQ(oCnt,1), EQ(uCnt,1)),
          LemAppend(sinPth, lemTxt), // Sin repeticion

      And(GE(aCnt,1), GE(eCnt,1), GE(iCnt,1), GE(oCnt,1), GE(uCnt,1),
          GE(aCnt+eCnt+iCnt+oCnt+uCnt, 6)),
          LemAppend(conPth, lemTxt), // Con repeticion

      TRUE, FALSE) // No tiene aeiou, nada que hacer
  });
  SetSum(lemCic) // Retorna el numero de palabras encontradas
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Escribe 2 ficheros:
a) en el primero todas las palabras que contienen todas las vocales,
   una sola vez, sin repeticion,
b) y en el segundo las que las tienen una o mas veces, con repeticion.
Escribe cada palabra en una linea del fichero.
Las vocales da igual que esten en mayusculas, en minusculas, acentuadas o
no acentuadas.
Retorna el numero total de palabras encontradas de ambas categorias.",
LemAeiou);
////////////////////////////////////////////////////////////////////////////
```

Real LemSimetricPair()

```
////////////////////////////////////////////////////////////////////////////
Real LemSimetricPair(Set  lemSet, // Lemas de entrada para buscar simetricas
                     Text filNam) // Nombre del fichero de salida
////////////////////////////////////////////////////////////////////////////
{
  Set  revSet = EvalSet(lemSet, Text(Text lemTxt) { Reverse(lemTxt) });

  Set  lemCla = Classify(lemSet << revSet,
                         Real(Text a, Text b) { Compare(a,b) });

  Set  simPar = Select(lemCla, Real(Set claSet)
                { And(Card(claSet) == 2, !TextFind(claSet[1],"-")) });

  Set  simFix = EvalSet(simPar, Text(Set claSet)
  {
    Text parTxt = claSet[1]+" | "+Reverse(claSet[1]);
    Repeat(" ", 12-Floor(TextLength(parTxt)/2)) + parTxt + "\n" // Centra el |
  });

  Text WriteFile(DirOut+"/"+filNam, SetSum(simFix)); // Escribe

  Card(simPar)
};
////////////////////////////////////////////////////////////////////////////
```

```
PutDescription(
"Selecciona de lemSet pares de palabras que una sea la simetrica de otra.
Para ello contactena el conjunto de palabras con sus inversas, las clasifica
por ser identicas y aquellos conjuntos con 2 ocurrencias indica que habia
una simetria, esto se puede hacer porque en el lemario no hay repetidos.
Omite las palabras del lemario que tengan guiones, usualmente son sufijos o
prefijos.
En esta seleccion se incluyen los palindromos que siempre son los simetricos
de ellos mismos.
Esta funcion diferencia las letras mayusculas de las minusculas y
las acentuadas de las no acentuadas.
Retorna el numero total de pares de palabras simetricas encontradas.",
LemSimetricPair);
////////////////////////////////////////////////////////////////////////////
```

Real LemGELength()

```
////////////////////////////////////////////////////////////////////////////
Real LemGELength(Set  lemSet, // Lemas de entrada
                 Real minChr, // Minimo de letras que ha de tener
                 Text filNam) // Nombre del fichero de salida
////////////////////////////////////////////////////////////////////////////
{
  Set  palSet = Select(lemSet, Real(Text lemTxt)
                       { TextLength(lemTxt) >= minChr });

  Text WriteFile(DirOut+"/"+filNam,
                 SetSum(EvalSet(palSet, Text(Text palTxt) { palTxt+"\n" })));

  Card(palSet)
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Selecciona de lemSet todas las palabras de mas de minChr letras y las
escribe de golpe en el fichero filNam añadiendo un salto de linea a cada
palabra.
Retorna el numero total de palabras encontradas.",
LemGELength);
////////////////////////////////////////////////////////////////////////////
```

Real LemSelect()

```
////////////////////////////////////////////////////////////////////////////
Real LemSelect(Set  lemSet, // Lemas de entrada
               Text filNam, // Nombre del fichero de salida
               Code funSel) // Funcion de seleccion Real funSel(Text)
////////////////////////////////////////////////////////////////////////////
{
  Set  palSet = Select(lemSet, funSel);

  Text WriteFile(DirOut+"/"+filNam,
                 SetSum(EvalSet(palSet, Text(Text palTxt) { palTxt+"\n" })));

  Card(palSet)
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Generalizacion de las funciones anteriores que selecciona de lemSet palabras
que  cumplan una determinada funcion funSel(palabra) y las escribe de golpe en
el fichero filNam añadiendo un salto de linea a cada palabra.
Retorna el numero total de palabras encontradas.",
LemSelect);
////////////////////////////////////////////////////////////////////////////
```

Real LemCicle()

```
////////////////////////////////////////////////////////////////////////////
Real LemCicle(Set  lemSet, // Lemas de entrada
              Real numChr, // Numero de caracteres a comparar
              Text filNam) // Nombre del fichero de salida
////////////////////////////////////////////////////////////////////////////
{
  LemSelect(lemSet, filNam, Real(Text lemTxt)
  {
    Text txtIni = Sub(lemTxt,1,numChr);
    Text txtRev = Sub(Reverse(lemTxt),1,numChr);
    txtIni == txtRev
  })
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Selecciona de lemSet todas las palabras que empiezan y terminan por las
mismas letras, pero revertidas, de forma que puedan formar un circulo y
las escribe en el fichero filNam.
El numero de caracteres a comparar puede ser 1, 2, ...
Utiliza la funcion de seleccion por palabras, una a una, LemSelect().
No selecciona por longitud y si considera diferentes las letras mayusculas de
las minusculas y las acentuadas de las que no lo son.
Retorna el numero total de palabras encontradas.",
LemCicle);
////////////////////////////////////////////////////////////////////////////
```

Real LemGrapheme()

```
////////////////////////////////////////////////////////////////////////////
Real LemGrapheme(Set  lemSet, // Conjunto de lemas de entrada
                 Set  numSet, // Conjunto de numero de grafemas para aparecer
                 Text filPat) // Patrol del nombre del fichero de salida
////////////////////////////////////////////////////////////////////////////
{
  // Retorna la ruta de un fichero de salida para el numero de grafemas
  // cambiando en el patron de nombre de fichero filPat _ por el numero
  Text filPth(Real numGra)
  { DirOut + "/" + Replace(filPat, "_", FormatReal(numGra, "%.0lf")) };

  // Inicializa todos los ficheros de salida
  Set  EvalSet(numSet, Text(Real numGra) { WriteFile(filPth(numGra), "") });

  Real graCnt(Text lemTxt) // Cuenta grafemas
  {
    Text lemRep = ReplaceTable(ToLower(lemTxt),
      [[ [["á", "_"]], [["é", "_"]], [["i", "_"]], [["í", "_"]],
         [["ó", "_"]], [["ú", "_"]], [["ü", "_"]], [["ñ", "_"]] ]]);
    TextOccurrences(lemRep,   "_") // Cuenta _
  };

  Set  graCic = EvalSet(lemSet, Set(Text lemTxt)   // Ciclo por palabras
                { [[ graCnt(lemTxt), lemTxt ]] }); // Cuenta y palabra

  Set  graCla = Classify(graCic, Real(Set a, Set b) // Clasifica de + a -
                         { Compare(b[1], a[1]) });

  Set  graWri = EvalSet(graCla, Real(Set graTab)
  {
    If(!(graTab[1][1] <: numSet), FALSE, // El 1º sin nº de grafemas deseado
    {
      SetSum(EvalSet(graTab, Real(Set graRow)  // Tabla [numero grafemas;lema]
             { LemAppend(filPth(graRow[1]), graRow[2]) }))
    })
  });
  SetSum(graWri) // Retorna el numero de palabras encontradas
};
////////////////////////////////////////////////////////////////////////////
```

```
PutDescription(
"Escribe n ficheros cada uno con las palabras que tienen un numero de grafemas
como el que se especifica en la lista numSet.
Escribe cada palabra en una linea de cada fichero.
Retorna el numero total de palabras encontradas en total.",
LemGrapheme);
////////////////////////////////////////////////////////////////////////////
```

Real LemRatio()

```
////////////////////////////////////////////////////////////////////////////
Real LemRatio(Set  lemSet, // Conjunto de lemas de entrada
              Real vocRat, // Ratio de vocales
              Text vocFil, // Fichero para mas vocales
              Text conFil) // Fichero para mas consonantes
////////////////////////////////////////////////////////////////////////////
{
  Text vocPth = DirOut + "/" + vocFil; // Ruta para mas vocales
  Text conPth = DirOut + "/" + conFil; // Ruta para mas consonantes

  Text WriteFile(vocPth, ""); // Inicializa el fichero de mas vocales
  Text WriteFile(conPth, ""); // Inicializa el fichero de mas vocales

  Real graCnt(Text lemTxt) // Cuenta vocales
  {
    Text lemRep = ReplaceTable(ToLower(lemTxt),
      [[ [["a", "_"]], [["á", "_"]], [["e", "_"]], [["é", "_"]],
         [["i", "_"]], [["í", "_"]], [["o", "_"]], [["ó", "_"]],
         [["u", "_"]], [["ú", "_"]], [["ü", "_"]]  ]]);
    TextOccurrences(lemRep,   "_") // Cuenta _
  };

  Set  lemCic = EvalSet(lemSet, Real(Text lemTxt) // Ciclo por palabras
  {
    Real lemRat =  graCnt(lemTxt) / TextLength(lemTxt); // Ratio vocales

    Case(
      lemRat >= vocRat,     LemAppend(vocPth, lemTxt), // muchas vocales
      lemRat <= (1-vocRat), LemAppend(conPth, lemTxt), // muchas consonantes
      TRUE, FALSE)          // Proporciones centrales no se guardan
  });
  SetSum(lemCic) // Retorna el numero de palabras encontradas
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Escribe 2 ficheros:
a) el primero cuando las vocales son muchas mas que las consonantes y
b) el segundo cuando las consonantes son muchas mas que las vocales.
Para realizar esta distincion emplea un ratio de vocales sobre el total de
las letras, por ejemplo el 60% de vocales, 0.6.
Escribe cada palabra en una linea de cada fichero.
Retorna el numero total de palabras encontradas de un tipo y del otro.",
LemRatio);
////////////////////////////////////////////////////////////////////////////
```

Real LemEquChr()

```
////////////////////////////////////////////////////////////////////////////
Real LemEquChr(Set  lemSet, // Lemas de entrada
               Real minRep, // Minimo de mismas letras para salir elegidos
               Text filNam) // Nombre del fichero de salida
////////////////////////////////////////////////////////////////////////////
{
  Set  lemTab = EvalSet(lemSet, Set(Text lemTxt) // Pares [letras; lema]
                { SetOfText(SetSum(LemChrSet(lemTxt)), lemTxt) });

  Set  lemCla = Classify(lemTab, // Clasificar por mismos caracteres
```

```
                        Real(Set a, Set b) { Compare(a[1], b[1]) });

  // Seleccionar los conjuntos con minRep o mas lemas de las mismas letras
  Set  lemSel = Select(lemCla, Real(Set claSet) { Card(claSet) >= minRep });

  Set  lemSrt = Sort(lemSel, Real(Set a, Set b) // Primero los mas repetidos
       { Compare(Card(b), Card(a)) });

  Set  lemWri = EvalSet(lemSrt, Text(Set claSet) // Tabla [letras; lema]
  {
    Set lemRow = Traspose(claSet)[2]; // La fila es la segunda columna
    Lem2Line(lemRow)+"\n"             // De fila de palabras a texto
  });

  Text WriteFile(DirOut+"/"+filNam, SetSum(lemWri)); // Escribe

  Card(lemWri) // Retorna el numero de grupos, no el de lemas
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Selecciona de lemSet conjuntos de lemas con las mismas letras y escribe
en el fichero de salida aquellos con minRep o mas elementos.
Para estas repeticiones distingue mayusculas de minusculas y acentos.
Retorna el numero total conjuntos de palabras encontrados.",
LemEquChr);
////////////////////////////////////////////////////////////////////////////
```

Real LemAllDifChr()

```
////////////////////////////////////////////////////////////////////////////
Real LemAllDifChr(Set  lemSet, // Lemas de entrada
                  Real minChr, // Numero minimo de caracteres
                  Text filNam) // Nombre del fichero de salida
////////////////////////////////////////////////////////////////////////////
{
  LemSelect(lemSet, filNam, Real(Text lemTxt)
  {
    Real numChr = TextLength(lemTxt); // Numero de caracteres
    If(numChr < minChr, FALSE,        // Demasiado corta
    {
      Set  chrSet = Unique(LemChrSet(lemTxt)); // Caracteres diferentes
      Card(chrSet) == numChr // Si son iguales -> todos son diferentes
    })
  })
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Selecciona de lemSet todas las palabras que tienen todas sus caracteres
diferentes y igual o mas numCrh letras y las escribe en el fichero filNam.
Utiliza la funcion de seleccion por palabras, una a una, LemSelect().
Considera diferentes las letras mayusculas de las minusculas y las acentuadas
de las que no lo son.
Pasando a minusculas y eliminado acentos puede hacerse la funcion equivalente
que considere que, por ejemplo, la e acentuada es igual que la e.
Retorna el numero total de palabras encontradas.",
LemAllDifChr);
////////////////////////////////////////////////////////////////////////////
```

Real LemTwice()

```
////////////////////////////////////////////////////////////////////////////
Real LemTwice(Set  lemSet, // Lemas de entrada
              Real casSen, // Control de mayusculas/minusculas y acentos
              Text filNam) // Nombre del fichero de salida
////////////////////////////////////////////////////////////////////////////
{
  LemSelect(lemSet, filNam, Real(Text lemTxt)
```

```
  {
    Real numChr = TextLength(lemTxt);        // Numero de caracteres

    Text lemCls = If(casSen, lemTxt, LemLowCls(lemTxt)); // Diferencia o no

    Set  chrCla = Classify(LemChrSet(lemCls), Real(Text a, Text b)
                           { Compare(a, b) }); // Caracteres iguales

    // Selecciona solo las clases de 2 caracteres iguales
    Set  selCla = Select(chrCla, Real(Set chrSet) { Card(chrSet) == 2 });

    EQ(2 * Card(selCla), numChr) // Todos los caracteres 2 veces
  })
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Selecciona de lemSet todas las palabras que tienen todos sus letras 2 veces.
Utiliza la funcion de seleccion por palabras, una a una, LemSelect().
Dependiendo de casSen:
a) Si es falso entonces considera iguales las letras mayusculas de las
   minusculas y las acentuadas iguales a las no acentudas.
b) Si es cierto diferencia las letras mayusculas de minusculas y las
    acentuadas de las vocales sin acento.
Retorna el numero total de palabras encontradas.",
LemTwice);
////////////////////////////////////////////////////////////////////////////
```

Real LemSec123()

```
////////////////////////////////////////////////////////////////////////////
Real LemSec123(Set  lemSet, // Lemas de entrada
               Real casSen, // Control de mayusculas/minusculas y acentos
               Text filNam) // Nombre del fichero de salida
////////////////////////////////////////////////////////////////////////////
{
  LemSelect(lemSet, filNam, Real(Text lemTxt)
  {
    Text lemCls = If(casSen, lemTxt, LemLowCls(lemTxt)); // Diferencia o no

    Set  chrCla = Classify(LemChrSet(lemCls), Real(Text a, Text b)
                           { Compare(a, b) }); // Caracteres iguales

    Set  chrSrt = Sort(chrCla, Real(Set a, Set b) // Por nº de ocurrencias
                           { Compare(Card(a), Card(b)) });

    Real chrCrd = Card(chrSrt); // Numero de conjunto de caracteres iguales

    Set  chrChk = For(1, chrCrd, Real(Real setPos) // Comprueba nº ocurrencias
                  { setPos == Card(chrSrt[setPos]) }); // 1 si igual, 0 si dif

    EQ(SetSum(chrChk), chrCrd) // Todos los caracteres 2 veces
  })
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Selecciona de lemSet todas las palabras que tienen 1 letra 1 vez, otra 2
veces, otra 3 veces, otra 4 veces y asi tantas como letras haya.
Utiliza la funcion de seleccion por palabras, una a una, LemSelect().
Dependiendo de casSen:
a) Si es falso entonces considera iguales las letras mayusculas de las
   minusculas y las acentuadas iguales a las no acentudas.
b) Si es cierto diferencia las letras mayusculas de minusculas y las
    acentuadas de las vocales sin acento.
Retorna el numero total de palabras encontradas.",
LemSec123);
////////////////////////////////////////////////////////////////////////////
```

Real LemJoin()

```
////////////////////////////////////////////////////////////////////////
Real LemJoin(Text filNam) // Nombre del fichero de salida
////////////////////////////////////////////////////////////////////////
{
  Set  filSet = GetDir(DirOut)[1];                          // Ficheros de salida
  Set  txtSet = EvalSet(filSet, Text(Text filNam)           // Leer los ficheros
                { Replace(ReadFile(DirOut+"/"+filNam)+"|", "\n", "|") });

  Text txtAll = SetSum(txtSet);                               // Unir los textos
  Set  lemSet = Tokenizer(txtAll, "|");         // Obtener el conjunto de lemas

  Set  lemCmp = EvalSet(lemSet, Text(Text lemTxt)         // Compactar los lemas
                { Compact(lemTxt) });

  Set  lemUni = Unique(lemCmp);                              // Eliminar repetidos
  Set  lemStr = Sort(lemUni, Real(Text a, Text b)             // Orden alfabetico
                { Compare(LemLowCls(a), LemLowCls(b)) });  // no de chars

  LemSelect(lemStr, filNam, Real(Text lemTxt)           // Escribir lemas buenos
  { And(lemTxt!="", !TextFind(lemTxt,"-"), !TextFind(lemTxt,"?")) })
};
////////////////////////////////////////////////////////////////////////
PutDescription(
"Recupera todas los lemas, las palabras, seleccionadas por sus diferentes
caracteristicas, palindromos, simetricas unas de otras, largas, con letras
repetidas 2 veces, etc. y crea en filNam un nuevo lemario de palabras
curiosas.
Retorna el numero total de palabras encontradas.",
LemJoin);
////////////////////////////////////////////////////////////////////////
```

Real makEnd

```
////////////////////////////////////////////////////////////////////////
Real makEnd = If(FALSE, FALSE, // Cambiar TRUE/FALSE para ejecutar
{
  Text WriteLn("\nRae.Lemario: LemEndAt()...");
  LemEndAt(
    LemInp, // Lemario de entrada
    [["b","ab","bab","c","f","g","h","j","k","m","p","t","u","v","x","y"]],
    "termina.en._.txt")
});
////////////////////////////////////////////////////////////////////////
PutDescription("Localiza y guarda palabras con cierta terminacion.", makEnd);
////////////////////////////////////////////////////////////////////////
```

Real makP01

```
////////////////////////////////////////////////////////////////////////
Real makP01 = If(FALSE, FALSE, // Cambiar TRUE/FALSE para ejecutar
{
  Text WriteLn("\nRae.Lemario: LemPalindrome01()...");
  LemPalindrome01(LemInp, 2, "palindroma.01.txt")
});
////////////////////////////////////////////////////////////////////////
PutDescription("Localiza y guarda palabras palindromas, version 1.", makP01);
////////////////////////////////////////////////////////////////////////
```

Real makP02

```
////////////////////////////////////////////////////////////////////////
Real makP02 = If(FALSE, FALSE, // Cambiar TRUE/FALSE para ejecutar
{
  Text WriteLn("\nRae.Lemario: LemPalindrome02()...");
```

```
  LemPalindrome01(LemInp, 2, "palindroma.02.txt")
});
////////////////////////////////////////////////////////////////////////////
PutDescription("Localiza y guarda palabras palindromas, version 2.", makP02);
////////////////////////////////////////////////////////////////////////////
```

Real mak_5v

```
////////////////////////////////////////////////////////////////////////////
Real mak_5v = If(FALSE, FALSE, // Cambiar TRUE/FALSE para ejecutar
{
  Text WriteLn("\nRae.Lemario: LemAeiou()...");
  LemAeiou(LemInp,                        // Lemario
           "aeiou.sin.repeticion.txt",  // Aeiou 1 sola vez
           "aeiou.con.repeticion.txt")  // Aeiou 1 o mas veces
});
////////////////////////////////////////////////////////////////////////////
PutDescription("Localiza y guarda palabras con todas las vocales.", mak_5v);
////////////////////////////////////////////////////////////////////////////
```

Real makSim

```
////////////////////////////////////////////////////////////////////////////
Real makSim = If(FALSE, FALSE, // Cambiar TRUE/FALSE para ejecutar
{
  Text WriteLn("\nRae.Lemario: LemSimetricPair()...");
  LemSimetricPair(LemInp,                      // Lemario
                  "palabras.simetricas.txt") // Salida
});
////////////////////////////////////////////////////////////////////////////
PutDescription("Localiza y guarda pares de palabras simetricas.", makSim);
////////////////////////////////////////////////////////////////////////////
```

Real makLen

```
////////////////////////////////////////////////////////////////////////////
Real makLen = If(FALSE, FALSE, // Cambiar TRUE/FALSE para ejecutar
{
  Text WriteLn("\nRae.Lemario: LemGELength()...");
  LemGELength(LemInp,                       // Lemario
              17,                           // Estas letras o mas
              "palabras.largas.txt")        // Salida
});
////////////////////////////////////////////////////////////////////////////
PutDescription("Localiza y guarda lemas de cierta longitud o mas.", makLen);
////////////////////////////////////////////////////////////////////////////
```

Real makCic

```
////////////////////////////////////////////////////////////////////////////
Real makCic = If(FALSE, FALSE, // Cambiar TRUE/FALSE para ejecutar
{
  Text WriteLn("\nRae.Lemario: LemCicle()...");
  LemCicle(LemInp,                        // Lemario
           2,                             // 2 letras a coincidir
           "empieza.termina.igual.txt") // Salida
});
////////////////////////////////////////////////////////////////////////////
PutDescription("Lemas que empiezan y terminan con la misma letra.", makCic);
////////////////////////////////////////////////////////////////////////////
```

Real makGra

```
////////////////////////////////////////////////////////////////////////
Real makGra = If(FALSE, FALSE, // Cambiar TRUE/FALSE para ejecutar
{
  Text WriteLn("\nRae.Lemario: LemGrapheme()...");
  LemGrapheme(LemInp,              // Lemario
              [[4,5,6]],           // numeros de grafemas
            "tiene._.grafemas.txt") // Salida
});
////////////////////////////////////////////////////////////////////////
PutDescription("Localiza y guarda lemas con varios grafemas.", makGra);
////////////////////////////////////////////////////////////////////////
```

Real makRat

```
////////////////////////////////////////////////////////////////////////
Real makRat = If(FALSE, FALSE, // Cambiar TRUE/FALSE para ejecutar
{
  Text WriteLn("\nRae.Lemario: LemRatio()...");
  LemRatio(LemInp,                              // Lemario
           0.7,                                 // Ratio de vocales 70%
           "mas.vocales.que.consonantes.txt", // >= % de vocales
           "mas.consonantes.que.vocales.txt") // >= % de consonantes
});
////////////////////////////////////////////////////////////////////////
PutDescription("Con mas vocales que consonantes o viceversa.", makRat);
////////////////////////////////////////////////////////////////////////
```

Real makEqu

```
////////////////////////////////////////////////////////////////////////
Real makEqu = If(FALSE, FALSE, // Cambiar TRUE/FALSE para ejecutar
{
  Text WriteLn("\nRae.Lemario: LemEquChr()...");
  LemEquChr(LemInp,                             // Lemario
            4,                                  // 4 o mas palabras en el grupo
            "tiene.las.mismas.letras.txt")    // Fichero de salida
});
////////////////////////////////////////////////////////////////////////
PutDescription("Busca y escribre palabras con las mismas letras.", makEqu);
////////////////////////////////////////////////////////////////////////
```

Real makDif

```
////////////////////////////////////////////////////////////////////////
Real makDif = If(FALSE, FALSE, // Cambiar TRUE/FALSE para ejecutar
{
  Text WriteLn("\nRae.Lemario: LemAllDifChr()...");
  LemAllDifChr(LemInp,                                  // Lemario
               10,                                      // 10 o mas caracteres
               "todos.los.carecteres.diferentes.txt") // Fichero de salida
});
////////////////////////////////////////////////////////////////////////
PutDescription("Palabras con todos sus caracteres diferentes.", makDif);
////////////////////////////////////////////////////////////////////////
```

Real makTwi

```
////////////////////////////////////////////////////////////////////////
```

```
Real makTwi = If(FALSE, FALSE, // Cambiar TRUE/FALSE para ejecutar
{
  Text WriteLn("\nRae.Lemario: LemTwice(strict)...");
  LemTwice(LemInp,                                    // Lemario
           TRUE,                                      // Estricto
           "todas.las.letras.2.veces.estricto.txt");  // Fichero de salida

  Text WriteLn("\nRae.Lemario: LemTwice(not strict)...");
  LemTwice(LemInp,                                    // Lemario
           FALSE,                                     // Estricto
           "todas.las.letras.2.veces.no.estricto.txt") // Fichero de salida
});
////////////////////////////////////////////////////////////////////////////
PutDescription("Palabras con todas sus letras 2 veces.", makTwi);
////////////////////////////////////////////////////////////////////////////
```

Real mak123

```
////////////////////////////////////////////////////////////////////////////
Real mak123 = If(FALSE, FALSE, // Cambiar TRUE/FALSE para ejecutar
{
  Text WriteLn("\nRae.Lemario: LemSec123(strict)...");
  LemSec123(LemInp,                                     // Lemario
            TRUE,                                       // Estricto
            "letras.repetidas.en.secuencia.estricto.txt");   // Salida

  Text WriteLn("\nRae.Lemario: LemSec123(not strict)...");
  LemSec123(LemInp,                                     // Lemario
            FALSE,                                      // Estricto
            "letras.repetidas.en.secuencia.no.estricto.txt") // Salida
});
////////////////////////////////////////////////////////////////////////////
PutDescription("1 letra repetida 1 vez, otra 2, otra 3, otra 4, ...", mak123);
////////////////////////////////////////////////////////////////////////////
```

Real makJoi

```
////////////////////////////////////////////////////////////////////////////
Real makJoi = If(FALSE, FALSE, // Cambiar TRUE/FALSE para ejecutar
{
  Text WriteLn("\nRae.Lemario: LemJoin()...");
  LemJoin("lemario.curioso.txt") // Salida
});
////////////////////////////////////////////////////////////////////////////
PutDescription("Une sin repeticion los lemas curiosos encontrados", makJoi);
////////////////////////////////////////////////////////////////////////////
```