make.tol de SHi.SyntaxHighlight

SHi.SyntaxHighlight es una libreria con funciones que permiten construir generadores de sintaxis realzada (syntax highlight). Por ejemplo, el programa Dct.Write, que genera contenidos para el sitio web lazytol.com, esta basado, entre otras, en esta libreria. Esta libreria de funciones esta fundamentalmente orientada para el realce de sintaxis del lenguaje de programacion Tol (Time Oriented Language), pero tambien sirve para otros lenguajes como por ejemplo: a) Sql (Structured Query Language), b) el lenguaje de los ficheros de especificacion de Gnuplot, c) el de los ficheros de mandatos de Windows de Microsoft, d) Css (Cascading Style Sheets), e) Html (HyperText Markup Language), f) Xml (eXtensible Markup Language) y g) Javascript. Para el realce de la sintaxis emplea los colores de los textos: a) negro para el codigo fuente, b) verde para los comentarios y c) azul para los textos.

SHi.SyntaxHighlight tambien es capaz de realzar sintaxis de algunos lenguajes embebidos en otros, en este caso lo que emplea es el color de fondo del texto, backgraound, en especial el rosa. Ejemplos de lenguajes embebidos en otros cuya sintaxis se puede realzar son: a) el lenguaje Javascript embebido dentro de codigo fuente Html o b) el propio lenguaje Tol embevido dentro de Html. En otros casos la forma de embeber un codigo de un lenguaje en otro lenguaje es mediante textos, por ejemplo, como cuando se introducen sentencias Sql dentro del lenguaje Tol que se ponen como textos. En estos casos el codigo embebido aparece den color azul de los textos.

Se trata de un realce de sintaxis basico orientado a generar la documentacion de los programas Tol, de su codigo fuente, ficheros auxiliares y de otros lenguajes de programacion que conviven con Tol, como Slq, Html, Xml, etc. en 2 formatos diferentes: a) como paginas web en Html y b) como documentos en formato Pdf. Este programa SHi.SyntaxHighlight realiza: a) una serie de pruebas de las funciones de sintaxsis realzada de su libreria y b) tambien permiten comprender como es la informacion que retorna la funcion Parse() del lenguaje Tol, si bien esta funcion es realmente empleada en otros programas que pueden basarse en esta libreria, como por ejemplo, Dct.Writer.

Árbol de ficheros

- SHi.SyntaxHighlight funciones de sintaxis realzada de codigo
  - ← make.tol programa de test de las funciones de sintaxis realzada en Tol
  - ← make.bat mandato de ejecucion del programa de test de realce de sintaxis
  - tol directorios que contienen fichero de codigo fuente Tol
    - cmm funciones comunes de manejo y generacion de textos y Html
      - ← txt.tol funciones para el manejo y la transformacion de textos
      - ← htm.tol funciones para generar codigo fuente en lenguaje Html
    - app funciones especificas de aplicacion de sintaxis realzada
      - ← shi.tol sintaxis realzada de código fuente en Tol, Xml, Html, etc.
    - ← inc.tol fichero con ordenes de inclusion de otros ficheros Tol

code.inp directorio con ficheros de codigo en diversos lenguajes para pruebas

- ← arr.js codigo fuente en lenguaje Javascript
- ← cmm.css ejemplo de Css, Cascading Style Sheets
- ← edi.sql lenguaje Sql, Structured Query Language
- ← gif.gpl fichero de especificacion de mandatos Gnuplot
- ← gpl.tol ejemplo de Tol invocando a Gnuplot
- ← map.xml ejemplo de codigo Xml, eXtensible Markup Language
- ← pdf.bat ficheros de mandatos de Windows de Microsoft
- ← see.htm lenguaje Html, HyperText Markup Language
- ← sql.tol ejemplo de lenguaje Tol con codigo Sql embebido

code.out directorio de resultados con el codigo de entrada realzado

- ← code.css Css para el realce de la sintaxis
- ← codeseed.htm semilla, template, para los resultados
- → codetest.htm fichero de salida con el resultado

→ resultado.html codigos fuente realzado en varios lenguajes de programacion

→ shi_syntaxhighlight.pdf documento de funciones de la libreria de realce de sintaxis

Declaraciones

Inclusiones

- `Set allInc`
  Inclusion de las funciones comunes y de aplicacion.

Pruebas

- `Real timIni`
  Para controlar tiempos.
- `Set tstSpl`
  Test de las funciones TxtSplitBy1Tag(), TxtSplitBy2Tag() y TxtSplitBy2Fast().
- `Text tstShi`
  Prueba de la funcion SHiTolPre() con codigo de Tol, Sql, Gnuplot, ficheros de mandatos, Css, Html, Xml y Javascript.
- `Real tstPar`
  Pruebas con la funcion Parse().

Set allInc

```
////////////////////////////////////////////////////////////////////////////
Set  allInc = Include("tol/inc.tol");
////////////////////////////////////////////////////////////////////////////
PutDescription("Inclusion de las funciones comunes y de aplicacion.", allInc);
////////////////////////////////////////////////////////////////////////////
```

Real timIni

```
////////////////////////////////////////////////////////////////////////
Real timIni = Copy(Time);
////////////////////////////////////////////////////////////////////////
PutDescription("Para controlar tiempos.", timIni);
////////////////////////////////////////////////////////////////////////
```

Set tstSpl

```
////////////////////////////////////////////////////////////////////////
Set tstSpl = {
  Text WriteLn("\nTxtSplitBy1Tag() test incluyendo correctos+incorrectos:");
  Set View(SetOfSet(
    TxtSplitBy1Tag("alfa|beta|alfa|beta|alfa", "|"),
    TxtSplitBy1Tag("alfa|beta|alfa|beta|", "|"),
    TxtSplitBy1Tag("alfa|beta|alfa|beta", "|"),
    TxtSplitBy1Tag("alfa|beta|alfa|", "|"),
    TxtSplitBy1Tag("alfa|beta|alfa", "|"),
    TxtSplitBy1Tag("alfa|beta|", "|"),
    TxtSplitBy1Tag("alfa|beta", "|"),
    TxtSplitBy1Tag("|beta|", "|"),
    TxtSplitBy1Tag("|beta", "|"),
    TxtSplitBy1Tag("alfa", "|")), "");

  Text WriteLn("\nTxtSplitBy2Tag() test incluyedo correctos+incorrectos:");
  Set View(SetOfSet(
    TxtSplitBy2Tag("alfa<beta>alfa<beta>alfa", "<", ">"),
    TxtSplitBy2Tag("alfa<beta>alfa<beta>", "<", ">"),
    TxtSplitBy2Tag("alfa<beta>alfa<beta", "<", ">"),
    TxtSplitBy2Tag("alfa<beta>alfa<", "<", ">"),
    TxtSplitBy2Tag("alfa<beta>alfa", "<", ">"),
    TxtSplitBy2Tag("alfa<beta>", "<", ">"),
    TxtSplitBy2Tag("alfa<beta", "<", ">"),
    TxtSplitBy2Tag("<beta>", "<", ">"),
    TxtSplitBy2Tag("<beta", "<", ">"),
    TxtSplitBy2Tag("alfa", "<", ">")), "");

  Text WriteLn("\nTxtSplitBy2Fast() test incluyendo correctos+incorrectos:");
  Set View(SetOfSet(
    TxtSplitBy2Fast("alfa<beta>alfa<beta>alfa", "<", ">"),
    TxtSplitBy2Fast("alfa<beta>alfa<beta>", "<", ">"),
    TxtSplitBy2Fast("alfa<beta>alfa<beta", "<", ">"),
    TxtSplitBy2Fast("alfa<beta>alfa<", "<", ">"),
    TxtSplitBy2Fast("alfa<beta>alfa", "<", ">"),
    TxtSplitBy2Fast("alfa<beta>", "<", ">"),
    TxtSplitBy2Fast("alfa<beta", "<", ">"),
    TxtSplitBy2Fast("<beta>", "<", ">"),
    TxtSplitBy2Fast("<beta", "<", ">"),
    TxtSplitBy2Fast("alfa", "<", ">")), "")
};
////////////////////////////////////////////////////////////////////////
PutDescription(
"Test de las funciones TxtSplitBy1Tag(), TxtSplitBy2Tag() y
TxtSplitBy2Fast().",
tstSpl);
////////////////////////////////////////////////////////////////////////
```

Text tstShi

```
////////////////////////////////////////////////////////////////////////
Text tstShi =
{
  Text WriteLn("\nSHiRemStrCodSet() test:");
```

```
  Text tolCod = ReadFile("code.inp/gpl.tol");
  Text tolSql = ReadFile("code.inp/sql.tol");
  Text sqlCod = ReadFile("code.inp/edi.sql");
  Text gplCod = ReadFile("code.inp/gif.gpl");
  Text cmdCod = ReadFile("code.inp/pdf.bat");
  Text cssCod = ReadFile("code.inp/cmm.css");
  Text xmlCod = ReadFile("code.inp/map.xml");
  Text htmCod = ReadFile("code.inp/see.htm");
  Text jscCod = ReadFile("code.inp/arr.js");

  Text htmSee = ReadFile("code.out/codeseed.htm");

  Text WriteFile("code.out/codetest.htm", Replace(htmSee, "_COD_",
    SHiTolPre(tolCod)+"\n<hr />\n"+
    SHiTolPre(tolSql)+"\n<hr />\n"+
    SHiSqlPre(sqlCod)+"\n<hr />\n"+
    SHiGplPre(gplCod)+"\n<hr />\n"+
    SHiCmdPre(cmdCod)+"\n<hr />\n"+
    SHiCssPre(cssCod)+"\n<hr />\n"+
    SHiJscPre(jscCod)+"\n<hr />\n"+
    SHiXmlPre(xmlCod)+"\n<hr />\n"+
    SHiEmbPre(htmCod)+"\n<hr />\n"+
    " "))
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Prueba de la funcion SHiTolPre() con codigo de Tol, Sql, Gnuplot, ficheros
de mandatos, Css, Html, Xml y Javascript.",
tstShi);
////////////////////////////////////////////////////////////////////////////
```

Real tstPar

```
////////////////////////////////////////////////////////////////////////////
Real tstPar =
{
  Text WriteLn("\nparse: begin\n");

  Text varCod = "
  ////////////////////////////////////////////////////////////////////////
  Text XxxMiniVariable = 1;
  ////////////////////////////////////////////////////////////////////////
  PutDescription(\"Vale 1, variable minima.\", XxxMiniVariable);
  ////////////////////////////////////////////////////////////////////////
  ";

  Set  varPar = Parse(varCod)[3]; // Acesso al arbol del parse

  Real varChk = varPar[1][1] == "=";        // Comprobar que es variable
  Text varCla = If(varChk, "variable", "no es variable");

  Text varTyp = varPar[1][3][1][1];         // Tipo
  Text varNam = varPar[1][3][1][3][1][1];   // Nombre
  Text varDes = varPar[2][3][1][1];         // Descripcion

  Text WriteLn(varTyp+" "+varNam+" = ...; // "+varDes+" ["+varCla+"]");

  Set  varTre = // Arbol similar al del parse, pero creado a mano
  [[
    [[ "=", "binary",
      [[
        [[ "Text", "type",
          [[
            [[ "XxxMiniVariable", "argument", Empty ]]
          ]]
        ]],
        [[ "1", "argument", Empty ]]
      ]]
    ]],
    [[ "PutDescription", "function",
      [[
        [[ "Vale 1, variable minima.", "argument", Empty ]] ,
```

```
      [[ "XxxMiniVariable", "argument", Empty ]]
    ]]
  ]]
]];

Text funCod = "
////////////////////////////////////////////////////////////////////////
Real XxxMiniFunction(Text argOne, Real argTwo)
////////////////////////////////////////////////////////////////////////
{ 1 };
////////////////////////////////////////////////////////////////////////
PutDescription(\"Retorna 1, funcion minima.\", XxxMiniFunction);
////////////////////////////////////////////////////////////////////////
";

Set  funPar = Parse(funCod)[3]; // Acesso al arbol del parse

Real funChk = funPar[1][1] == "#F#";      // Comprobar que es funcion
Text funCla = If(funChk, "funcion", "no es funcion");

Text funTyp = funPar[1][3][1][1];         // Tipo
Text funNam = funPar[1][3][1][3][1][1];   // Nombre
Text funDes = funPar[2][3][1][1];         // Descripcion

Text WriteLn(funTyp+" "+funNam+"(...) {...}; // "+funDes+" ["+funCla+"]");

Set  argSet = funPar[1][3][1][3][1][3];   // Conjunto de argumentos
Real argNum = Card(argSet);               // Numero de argumentos

Set  argCic = For(1, argNum, Text(Real argPos)
{
  Set  argInf = argSet[argPos]; // Informacion del argumento
  Text argTyp = argInf[1];
  Text argNam = argInf[3][1][1];

  Case(EQ(argPos,      1), "("+argTyp+" "+argNam+", ",
       EQ(argPos, argNum),     argTyp+" "+argNam+")",
       TRUE,                   argTyp+" "+argNam+", ")
});
Text argTxt = SetSum(argCic);

Text WriteLn(funTyp+" "+funNam+argTxt+" {...}; // "+funDes+" ["+funCla+"]");

Set  funTre = // Arbol similar al del parse, pero creado a mano
[[
  [[ "#F#", "binary",
    [[
      [[ "Real", "type",
        [[
          [[ "XxxMiniFunction", "function",
            [[
              [[ "Text", "type",
                [[
                  [[ "argOne", "argument", Empty ]]
                ]]
              ]],
              [[ "Real", "type",
                [[
                  [[ "argTwo", "argument", Empty ]]
                ]]
              ]]
            ]]
          ]]
        ]]
      ]],
      [[ "1", "argument", Empty ]]
    ]]
  ]] ,
  [[ "PutDescription", "function",
    [[
      [[ "Retorna 1, funcion minima.", "argument", Empty ]] ,
      [[ "XxxMiniFunction", "argument", Empty ]]
```

```
      ]]
    ]]
  ]];

 TRUE
};
////////////////////////////////////////////////////////////////////////////
PutDescription("Pruebas con la funcion Parse().",tstPar);
////////////////////////////////////////////////////////////////////////////
```

txt.tol de SHi.SyntaxHighlight

Funciones de textos.

Declaraciones

Funciones

- `Real TxtBeginStrict(Text txtInp, Text txtIni)`
  Version estricta de TextBeginWith(), da falso siempre que txtIni sea nulo.
- `Real TxtEndStrict(Text txtInp, Text txtEnd)`
  Version estricta de TextEndAt(), da falso siempre que txtEnd sea nulo.
- `Text TxtBetween2Tag(Text inpTxt, Text tagIni, Text tagEnd, Real cmpFlg)`
  Retorna un subtexto entre la primera ocurrencia de tagIni y tagEnd. Si tagIni o tagEnd no aparecen retorna la tira vacia. Si cmpFlg es cierto entonces aplica la funcion Compact() al texto que retorna. Por ejemplo: TxtBetween2Tag('a b [[c]] d [[e]] f', '[[', ']]', TRUE) retorna 'c'.
- `Set TxtSplitBy1Tag(Text txtInp, Text tagBrk)`
  Retorna un conjunto de textos resultado de cortar el texto de entrada por un unico tag tagBrk incluyendo dicho tagBrk al inicio y al final de cada texto que enmarca, las ocurrencias impares de tagBrk al inicio del texto y las impares al final. Tiene mas sentido cuando el numero de ocurrencias de tagBrk es par. Si el numero de ocurrencias de tagbrk es impar funciona como si al final del texto txtInp hubiera una ultima ocurrencia. Fue una funcion recursiva, pero la versión 2.0.1 de Tol se caia con textos grandes, ahora es una funcion iterativa. TxtSplitBy1Tag(aaa|::|bbb|---|ccc, |) -> [aaav, |::|, bbb, |---|, ccc].
- `Set TxtSplitBy2Tag(Text txtInp, Text tagIni, Text tagEnd)`
  Retorna un conjunto de textos resultado de cortar el texto de entrada por un dos tags de inicio y de fin incluyendo dichos tags al inicio y al final de cada texto que enmarcan. Los tags tagIni y tabEnd no han de ser nulos. Tiene mas sentido cuando a cada tag de inicio le corresponde uno de final. Es una funcion recursiva. Si funciona correctamente si los tags inicial y final son iguales. TxtSplitBy2Tag(aaa<::>bbb<--->ccc, <, >) -> [aaav, <::>, bbb, <--->, ccc].
- `Set TxtSplitBy2Fast(Text txtInp, Text tagIni, Text tagEnd)`
  Retorna un conjunto de textos resultado de cortar el texto de entrada por dos tags de inicio y de fin incluyendo dichos tags al inicio y al final de cada texto que enmarcan. Los tags tagIni y tabEnd no han de ser nulos ni iguales, si son iguales lo correcto seria utilizar la funcion TxtSplitBy1Tag(). No es una funcion recursiva, es mas rapida que TxtSplitBy2Tag() pero menos resistente a la reiteracion de tags de inicio con un unico final. No funciona correctamente si los tags inicial y final son iguales. Tiene mas sentido cuando a cada tag de inicio le corresponde uno de final. TxtSplitBy2Fast(aaa<::>bbb<--->ccc, <, >) -> [aaav, <::>, bbb, <--->, ccc].
- `Set TxtTokenizer(Text txtInp, Text tagBrk)`

Retorna un conjunto de textos resultado de cortar el texto de entrada por un unico tag tagBrk no incluyendo el tag tagBrk dentro de los textos. Se soporta en la funcion Tol Tokenizer() que rompe por un unico caracter. Usa como caracter interno de corte el 7 (bell), esperando que no aparezca.

Real TxtBeginStrict()

```
////////////////////////////////////////////////////////////////////////////
Real TxtBeginStrict(Text txtInp, // Texto de entrada
                    Text txtIni) // Texto inicial
////////////////////////////////////////////////////////////////////////////
{ If(txtIni=="", FALSE, TextBeginWith(txtInp, txtIni)) };
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Version estricta de TextBeginWith(), da falso siempre que txtIni sea nulo.",
TxtBeginStrict);
////////////////////////////////////////////////////////////////////////////
```

Real TxtEndStrict()

```
////////////////////////////////////////////////////////////////////////////
Real TxtEndStrict(Text txtInp, // Texto de entrada
                  Text txtEnd) // Texto final
////////////////////////////////////////////////////////////////////////////
{ If(txtEnd=="", FALSE, TextEndAt(txtInp, txtEnd)) };
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Version estricta de TextEndAt(), da falso siempre que txtEnd sea nulo.",
TxtEndStrict);
////////////////////////////////////////////////////////////////////////////
```

Text TxtBetween2Tag()

```
////////////////////////////////////////////////////////////////////////////
Text TxtBetween2Tag(Text inpTxt, // Texto de entrada
                    Text tagIni, // Tag inicial
                    Text tagEnd, // Tag final
                    Real cmpFlg) // Si true aplica Compact()
////////////////////////////////////////////////////////////////////////////
{
  Real posIni = TextFind(inpTxt, tagIni);
  Text result = If(LE(posIni,0), "",
  {
    Real lenIni = TextLength(tagIni);
    Real posSub = posIni + lenIni;
    Real posEnd = TextFind(inpTxt, tagEnd, posSub);
    If(LE(posEnd, 0), "", Sub(inpTxt, posSub, posEnd-1))
  });
  If(cmpFlg, Compact(result), result)
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna un subtexto entre la primera ocurrencia de tagIni y tagEnd.
Si tagIni o tagEnd no aparecen retorna la tira vacia.
Si cmpFlg es cierto entonces aplica la funcion Compact() al texto que retorna.
Por ejemplo:
  TxtBetween2Tag('a b [[ c ]] d [[ e ]] f', '[[', ']]', TRUE)
  retorna 'c'.",
TxtBetween2Tag);
////////////////////////////////////////////////////////////////////////////
```

Set TxtSplitBy1Tag()

```
////////////////////////////////////////////////////////////////////////////
Set TxtSplitBy1Tag(Text txtInp, // Texto de entrada
                   Text tagBrk) // Tag por el que se corta
////////////////////////////////////////////////////////////////////////////
{
  Set txtTok = TxtTokenizer(txtInp, tagBrk);
  For(1, Card(txtTok), Text(Real posTok) // Ciclo para impares y pares
  { If(posTok%2, txtTok[posTok], tagBrk+txtTok[posTok]+tagBrk) })
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna un conjunto de textos resultado de cortar el texto de entrada por un
unico tag tagBrk incluyendo dicho tagBrk al inicio y al final de cada texto
que enmarca, las ocurrencias impares de tagBrk al inicio del texto y
las impares al final.
Tiene mas sentido cuando el numero de ocurrencias de tagBrk es par.
Si el numero de ocurrencias de tagbrk es impar funciona como si al final del
texto txtInp hubiera una ultima ocurrencia.
Fue una funcion recursiva, pero la versión 2.0.1 de Tol se caia con textos
grandes, ahora es una funcion iterativa.
TxtSplitBy1Tag(aaa|::|bbb|---|ccc, |) -> [aaav, |::|, bbb, |---|, ccc].",
TxtSplitBy1Tag);
////////////////////////////////////////////////////////////////////////////
```

Set TxtSplitBy2Tag()

```
////////////////////////////////////////////////////////////////////////////
Set TxtSplitBy2Tag(Text txtInp, // Texto de entrada
                   Text tagIni, // Tag inicial por el que se corta
                   Text tagEnd) // Tag final por el que se corta
////////////////////////////////////////////////////////////////////////////
{
  Real posIni = TextFind(txtInp, tagIni);
  If(LE(posIni,0), SetOfText(txtInp), // Nada que cortar
  {
    Real lenTxt = TextLength(txtInp); // Longitud del texto de entrada
    Real lenIni = TextLength(tagIni); // Longitud del tag inicial
    Real lenEnd = TextLength(tagEnd); // Longitud del tag final
    Real posSub = posIni + lenIni;
    Real posEnd = TextFind(txtInp, tagEnd, posSub);
    Case(
      Or(And(EQ(posIni,1),LE(posEnd,0)), // Ini en posicion 1 pero no termina
         And(EQ(posIni,1),EQ(posEnd+lenEnd-1,lenTxt))), // Justo ini y final
         SetOfText(txtInp),
      Or(And(GT(posIni,1),LE(posEnd,0)), // Ini en posicion >1 pero no termina
         And(GT(posIni,1),EQ(posEnd+lenEnd-1,lenTxt))), // Justo al final
         SetOfText(
           Sub(txtInp, 1,      posIni-1),
           Sub(txtInp, posIni, lenTxt)),
      And(EQ(posIni,1),GT(posEnd,1)), // Inicia en posicion 1, termina y sigue
        SetOfText(
          Sub(txtInp, 1,       posEnd+lenEnd-1)) <<
          TxtSplitBy2Tag(Sub(txtInp, posEnd+lenEnd, lenTxt), tagIni, tagEnd),
      TRUE,                              // Inicia posicion >1, termina y sigue
        SetOfText(
          Sub(txtInp, 1,       posIni-1),
          Sub(txtInp, posIni,  posEnd+lenEnd-1)) <<
          TxtSplitBy2Tag(Sub(txtInp, posEnd+lenEnd, lenTxt), tagIni, tagEnd))
  })
};
////////////////////////////////////////////////////////////////////////////
```

```
PutDescription(
"Retorna un conjunto de textos resultado de cortar el texto de entrada por un
dos tags de inicio y de fin incluyendo dichos tags al inicio y al final de
cada texto que enmarcan.
Los tags tagIni y tabEnd no han de ser nulos.
Tiene mas sentido cuando a cada tag de inicio le corresponde uno de final.
Es una funcion recursiva.
Si funciona correctamente si los tags inicial y final son iguales.
TxtSplitBy2Tag(aaa<::>bbb<--->ccc, <, >) -> [aaav, <::>, bbb, <--->, ccc].",
TxtSplitBy2Tag);
////////////////////////////////////////////////////////////////////////////
```

Set TxtSplitBy2Fast()

```
////////////////////////////////////////////////////////////////////////////
Set TxtSplitBy2Fast(Text txtInp, // Texto de entrada
                    Text tagIni, // Tag inicial por el que se corta
                    Text tagEnd) // Tag final por el que se corta
////////////////////////////////////////////////////////////////////////////
{
  Text chrBrk = Char(7); // Caracter auxiliar de corte que se espera unico
  Text repEnd = Replace(txtInp, tagEnd, tagEnd+chrBrk);
  Set  txtSet = Tokenizer(repEnd, chrBrk);

  Set  cicSet = EvalSet(txtSet, Set(Text txtTok)
  {

    Text repIni = Replace(txtTok, tagIni, chrBrk+tagIni);
    Set  tokSet = Tokenizer(repIni, chrBrk);
    Select(tokSet, Real(Text tokTxt) { tokTxt != "" }) // Elimina los vacios
  });
  BinGroup("<<", cicSet) // De conjunto de pares a conjunto lineal
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna un conjunto de textos resultado de cortar el texto de entrada por
dos tags de inicio y de fin incluyendo dichos tags al inicio y al final de
cada texto que enmarcan.
Los tags tagIni y tabEnd no han de ser nulos ni iguales, si son iguales lo
correcto seria utilizar la funcion TxtSplitBy1Tag().
No es una funcion recursiva, es mas rapida que TxtSplitBy2Tag() pero menos
resistente a la reiteracion de tags de inicio con un unico final.
No funciona correctamente si los tags inicial y final son iguales.
Tiene mas sentido cuando a cada tag de inicio le corresponde uno de final.
TxtSplitBy2Fast(aaa<::>bbb<--->ccc, <, >) -> [aaav, <::>, bbb, <--->, ccc].",
TxtSplitBy2Fast);
////////////////////////////////////////////////////////////////////////////
```

Set TxtTokenizer()

```
////////////////////////////////////////////////////////////////////////////
Set TxtTokenizer(Text txtInp, // Texto de entrada
                 Text tagBrk) // Tag por el que se corta
////////////////////////////////////////////////////////////////////////////
{ Tokenizer(Replace(txtInp, tagBrk, Char(7)), Char(7)) };
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna un conjunto de textos resultado de cortar el texto de entrada por un
unico tag tagBrk no incluyendo el tag tagBrk dentro de los textos.
Se soporta en la funcion Tol Tokenizer() que rompe por un unico caracter.
Usa como caracter interno de corte el 7 (bell), esperando que no aparezca.",
TxtTokenizer);
////////////////////////////////////////////////////////////////////////////
```

htm.tol de SHi.SyntaxHighlight

Funciones de basicas de Html (HyperText Markup Language).

Declaraciones

Funciones

- `Text HtmAsc2Xml(Text codInp)`

  Retorna el codigo resultado de reconvertir &, < y >.

Text HtmAsc2Xml()

```
////////////////////////////////////////////////////////////////////////////
Text HtmAsc2Xml(Text codInp) // Codigo de entrada
////////////////////////////////////////////////////////////////////////////
{
  ReplaceTable(codInp, [[ [["&", "&amp;"]],
                          [["<", "&lt;"]],
                          [[">", "&gt;"]] ]], 1)
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna el codigo resultado de reconvertir &, < y >.",
HtmAsc2Xml);
////////////////////////////////////////////////////////////////////////////
```

shi.tol de SHi.SyntaxHighlight

Funciones de sintaxis realzada (syntax highlight).

Declaraciones

Constantes

- `Text SHiRemOpn`

  Inicio tipico de un comentario de bloque.

- `Text SHiRemCls`

  Cierre tipico de un comentario de bloque.

- `Text SHiQuo`

  Comillas dobles.

- `Text SHiRemHtm`

  Span para un comentario.

- `Text SHiQu1Htm`

  Span para un texto de primer nivel.

- `Text SHiQu2Htm`
  Span para un subtexto de segundo nivel.
- `Text SHiTagHtm`
  Span para el inicio de una etiqueta.
- `Text SHiCodEmb`
  Span para el codigo embebido.
- `Text SHiEndHtm`
  Cierre de un span.

Funciones

- `Set SHiRemStrCodSet(Text txtInp, Text remOpn, Text remCls, Text remLin, Text strQu1, Text strQu2)`
  Retorna un conjunto de textos resultado de separar los comentarios de bloque, los comentarios de linea, los textos entrecomillados y el codigo.
- `Text SHiRemStrCodHtm(Text txtInp, Text remOpn, Text remCls, Text remLin, Text strQu1, Text strQu2)`
  Retorna el codigo Html resultado de aplicar sintaxis realzada al codigo de entrada txtImp. El ReplaceTable() para transformar < y > no se puede aplicar desde el principio porque a la hora de realzar codigo Html son los tags delimitadores mas importantes, por eso hay que hacerlo trozo a trozo de codigo.
- `Text SHiPre(Text txtInp)`
  Retorna el codigo Html resultado de enmarcar txtImp entre las etiquetas <pre><code> y </code></pre>.
- `Text SHiRemStrCodPre(Text txtInp, Text remOpn, Text remCls, Text remLin, Text strQu1, Text strQu2)`
  Retorna el codigo Html resultado de aplicar sintaxis realzada al codigo de txtImp enmarcado todo el resultado entre las etiquetas <pre><code> y </code></pre>.
- `Text SHiTolHtm(Text codInp)`
  Retorna el codigo Html resultado de aplicar sintaxis realzada al codigo Tol de entrada que recibe como parametro pero sin enmarcar entre las etiquetas <pre><code> y </code> </pre>.
- `Text SHiTolPre(Text codInp)`
  Retorna el codigo Html resultado de aplicar sintaxis realzada al codigo Tol de entrada que recibe como parametro.
- `Text SHiJscHtm(Text codInp)`
  Retorna el codigo Html resultado de aplicar sintaxis realzada al codigo Javascript de entrada que recibe como parametro pero sin enmarcar entre las etiquetas <pre><code> y </code></pre>.
- `Text SHiJscPre(Text codInp)`
  Retorna el codigo Html resultado de aplicar sintaxis realzada al codigo Javascript de entrada que recibe como parametro.
- `Text SHiSqlPre(Text codInp)`

Retorna el codigo Html resultado de aplicar sintaxis realzada al codigo Sql de entrada que recibe como parametro.

- `Text SHiGplPre(Text codInp)`
  Retorna el codigo Html resultado de aplicar sintaxis realzada al codigo Gnuplot de entrada que recibe como parametro.
- `Text SHiCmdPre(Text codInp)`
  Retorna el codigo Html resultado de aplicar sintaxis realzada al codigo de mandatos para el Cmd de Dos que recibe como parametro.
- `Text SHiCssPre(Text codInp)`
  Retorna el codigo Html resultado de aplicar sintaxis realzada al codigo Cascade style sheet que recibe como parametro.
- `Text SHiXmlHtm(Text codInp)`
  Retorna el codigo Html resultado de aplicar sintaxis realzada al codigo Xml que recibe como parametro pero sin enmarcar en las etiquetas <pre><code> y </code></pre> Emplea la version no recursiva TxtSplitBy2Fast(), aunque menos robusta que TxtSplitBy2Tag(), para evitar caidas en fichero Xml de cierta longitud.
- `Text SHiXmlPre(Text codInp)`
  Retorna el codigo Html resultado de aplicar sintaxis realzada al codigo Xml que recibe como parametro.
- `Text SHiEmbHtm(Text codInp)`
  Retorna el codigo Html resultado de aplicar sintaxis realzada al codigo Html basico que recibe como parametro pero sin enmarcar en las etiquetas <pre><code> y </code></pre>.
- `Text SHiEmbPre(Text codInp)`
  Retorna el codigo Html resultado de aplicar sintaxis realzada al codigo Xml que recibe como parametro.
- `Text SHiAscPre(Text codInp)`
  Retorna el codigo Html del contenido Ascii que recibe como parametro. Unicamente cambia los caracteres especiales con HtmAsc2Xml() y pone las etiquetas <pre><code> y </code> </pre>
- `Text SHiCompact(Text codHtm)`
  Retorna el codigo Html resultado de eliminar las lineas en blanco antes y despues del <pre><code> y </code></pre>.
- `Text SHiPreCode(Text filPth, Text filTxt)`
  Retorna el codigo Html resultado de aplicar sintaxis realzada al codigo filTxt que se la leido, total o parcialmente, de fichero filPth. Recibe el nombre o ruta del fichero para saber su extension y de ahi el lenguaje de programacion.
- `Text SHiPreFile(Text filPth)`
  Retorna el codigo Html resultado de aplicar sintaxis realzada a todo el codigo del fichero que recibe como parametro.

Constantes

Text SHiRemOpn

```
////////////////////////////////////////////////////////////////////////////
Text SHiRemOpn = "/"+"*";
////////////////////////////////////////////////////////////////////////////
PutDescription("Inicio tipico de un comentario de bloque.", SHiRemOpn);
////////////////////////////////////////////////////////////////////////////
```

Text SHiRemCls

```
////////////////////////////////////////////////////////////////////////////
Text SHiRemCls = "*"+"/";
////////////////////////////////////////////////////////////////////////////
PutDescription("Cierre tipico de un comentario de bloque.", SHiRemCls);
////////////////////////////////////////////////////////////////////////////
```

Text SHiQuo

```
////////////////////////////////////////////////////////////////////////////
Text SHiQuo = Char(34);
////////////////////////////////////////////////////////////////////////////
PutDescription("Comillas dobles.", SHiQuo);
////////////////////////////////////////////////////////////////////////////
```

Text SHiRemHtm

```
////////////////////////////////////////////////////////////////////////////
Text SHiRemHtm = "<span class='CodRem'>";
////////////////////////////////////////////////////////////////////////////
PutDescription("Span para un comentario.", SHiRemHtm);
////////////////////////////////////////////////////////////////////////////
```

Text SHiQu1Htm

```
////////////////////////////////////////////////////////////////////////////
Text SHiQu1Htm = "<span class='CodTxt'>";
////////////////////////////////////////////////////////////////////////////
PutDescription("Span para un texto de primer nivel.", SHiQu1Htm);
////////////////////////////////////////////////////////////////////////////
```

Text SHiQu2Htm

```
////////////////////////////////////////////////////////////////////////////
Text SHiQu2Htm = "<span class='CodSub'>";
////////////////////////////////////////////////////////////////////////////
PutDescription("Span para un subtexto de segundo nivel.", SHiQu2Htm);
////////////////////////////////////////////////////////////////////////////
```

Text SHiTagHtm

```
////////////////////////////////////////////////////////////////////////////
Text SHiTagHtm = "<span class='CodTag'>";
////////////////////////////////////////////////////////////////////////////
PutDescription("Span para el inicio de una etiqueta.", SHiTagHtm);
```

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

Text SHiCodEmb

```
////////////////////////////////////////////////////////////////////////////
Text SHiCodEmb = "<span class='CodEmb'>";
////////////////////////////////////////////////////////////////////////////
PutDescription("Span para el codigo embebido.", SHiCodEmb);
////////////////////////////////////////////////////////////////////////////
```

Text SHiEndHtm

```
////////////////////////////////////////////////////////////////////////////
Text SHiEndHtm = "</span>";
////////////////////////////////////////////////////////////////////////////
PutDescription("Cierre de un span.", SHiEndHtm);
////////////////////////////////////////////////////////////////////////////
```

Set SHiRemStrCodSet()

```
////////////////////////////////////////////////////////////////////////////
Set SHiRemStrCodSet(Text txtInp, // Texto de entrada
                    Text remOpn, // Apertura de los comentarios de bloque
                    Text remCls, // Cierre de los comentarios de bloque
                    Text remLin, // Apertura de los comentarios de linea
                    Text strQu1, // Delimitador de textos alto nivel (ie "")
                    Text strQu2) // Delimitacor de textos bajo nivel (ie '')
////////////////////////////////////////////////////////////////////////////
{
  Set  setBlk = If(remOpn == "", [[txtInp]], // No hay comentarios de bloque
                   TxtSplitBy2Tag(txtInp, remOpn, remCls)); // Comenta bloque

  Set  setQu1 = BinGroup("<<", EvalSet(setBlk, Set(Text txtCod)
  {
    If(TxtBeginStrict(txtCod, remOpn), [[txtCod]], // Era comentario de bloque
       TxtSplitBy1Tag(txtCod, strQu1))   // Romper por textos entrecomillados
  })); // De conjunto de conjuntos a conjunto lineal

  Set  setQu2 = If(strQu2 == "", setQu1, // Si no se emplea
  {
    BinGroup("<<", EvalSet(setQu1, Set(Text txtCod)
    {
      If(TxtBeginStrict(txtCod, remOpn), [[txtCod]], // Era comentario bloque
      If(TxtBeginStrict(txtCod, strQu1), [[txtCod]], // Quote alto nivel
         TxtSplitBy1Tag(txtCod, strQu2)))  // Romper textos y entrecomillados
    }))  // De conjunto de conjuntos a conjunto lineal
  });

  Set  setLin = If(remLin == "", setQu2, // Si no se emplea
  {
    BinGroup("<<", EvalSet(setQu2, Set(Text txtCod)
    {
      If(TxtBeginStrict(txtCod, remOpn), [[txtCod]], // Era comentario bloque
      If(TxtBeginStrict(txtCod, strQu1), [[txtCod]], // Quote alto nivel
      If(TxtBeginStrict(txtCod, strQu2), [[txtCod]], // Quote bajo nivel
         TxtSplitBy2Tag(txtCod, remLin, "\n"))))  // Romper comentarios linea
    })) // De conjunto de conjuntos a conjunto lineal
  });

  setLin
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna un conjunto de textos resultado de separar los comentarios de bloque,
los comentarios de linea, los textos entrecomillados y el codigo.",
```

```
SHiRemStrCodSet);
////////////////////////////////////////////////////////////////////////
```

Text SHiRemStrCodHtm()

```
////////////////////////////////////////////////////////////////////////
Text SHiRemStrCodHtm(Text txtInp, // Texto de entrada
                     Text remOpn, // Apertura de los comentarios de bloque
                     Text remCls, // Cierre de los comentarios de bloque
                     Text remLin, // Apertura de los comentarios de linea
                     Text strQu1, // Delimitador de textos alto nivel (ie "")
                     Text strQu2) // Delimitacor de textos bajo nivel (ie '')
////////////////////////////////////////////////////////////////////////
{
  Set  setBlk = SHiRemStrCodSet(txtInp,remOpn,remCls,remLin,strQu1,strQu2);
  Set  setHtm = EvalSet(setBlk, Text(Text txtCod)
  {
    Case(
      TxtBeginStrict(txtCod, remOpn),
        SHiRemHtm + HtmAsc2Xml(txtCod) + SHiEndHtm,

      TxtBeginStrict(txtCod, strQu1),
        SHiQu1Htm + HtmAsc2Xml(txtCod) + SHiEndHtm,

      TxtBeginStrict(txtCod, strQu2),
        SHiQu2Htm + HtmAsc2Xml(txtCod) + SHiEndHtm,

      TxtBeginStrict(txtCod, remLin),
        SHiRemHtm + Replace(HtmAsc2Xml(txtCod), "\n", SHiEndHtm+"\n"),

      TRUE,
        HtmAsc2Xml(txtCod))
  });
  SetSum(setHtm)
};
////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna el codigo Html resultado de aplicar sintaxis realzada al codigo de
entrada txtImp.
El ReplaceTable() para transformar < y > no se puede aplicar desde el
principio porque a la hora de realzar codigo Html son los tags delimitadores
mas importantes, por eso hay que hacerlo trozo a trozo de codigo.",
SHiRemStrCodHtm);
////////////////////////////////////////////////////////////////////////
```

Text SHiPre()

```
////////////////////////////////////////////////////////////////////////
Text SHiPre(Text txtInp) // Texto de entrada
////////////////////////////////////////////////////////////////////////
{ "\n<pre><code>" + txtInp + "</code></pre>\n" };
////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna el codigo Html resultado de enmarcar txtImp entre las etiquetas
<pre><code> y </code></pre>.",
SHiPre);
////////////////////////////////////////////////////////////////////////
```

Text SHiRemStrCodPre()

```
////////////////////////////////////////////////////////////////////////
Text SHiRemStrCodPre(Text txtInp, // Texto de entrada
                     Text remOpn, // Apertura de los comentarios de bloque
                     Text remCls, // Cierre de los comentarios de bloque
                     Text remLin, // Apertura de los comentarios de linea
```

```
                    Text strQu1, // Delimitador de textos alto nivel (ie "")
                    Text strQu2) // Delimitacor de textos bajo nivel (ie '')
////////////////////////////////////////////////////////////////////////////
{ SHiPre(SHiRemStrCodHtm(txtInp,remOpn,remCls,remLin,strQu1,strQu2)) };
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna el codigo Html resultado de aplicar sintaxis realzada al codigo de
txtImp enmarcado todo el resultado entre las etiquetas <pre><code> y
</code></pre>.",
SHiRemStrCodPre);
////////////////////////////////////////////////////////////////////////////
```

Text SHiTolHtm()

```
////////////////////////////////////////////////////////////////////////////
Text SHiTolHtm(Text codInp) // Codigo de entrada Tol
////////////////////////////////////////////////////////////////////////////
{ SHiRemStrCodHtm(codInp, SHiRemOpn, SHiRemCls, "//", SHiQuo, "") };
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna el codigo Html resultado de aplicar sintaxis realzada al codigo
Tol de entrada que recibe como parametro pero sin enmarcar entre las
etiquetas <pre><code> y </code></pre>.",
SHiTolHtm);
////////////////////////////////////////////////////////////////////////////
```

Text SHiTolPre()

```
////////////////////////////////////////////////////////////////////////////
Text SHiTolPre(Text codInp) // Codigo de entrada Tol
////////////////////////////////////////////////////////////////////////////
{ SHiPre(SHiTolHtm(codInp)) };
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna el codigo Html resultado de aplicar sintaxis realzada al codigo
Tol de entrada que recibe como parametro.",
SHiTolPre);
////////////////////////////////////////////////////////////////////////////
```

Text SHiJscHtm()

```
////////////////////////////////////////////////////////////////////////////
Text SHiJscHtm(Text codInp) // Codigo de entrada Javascript
////////////////////////////////////////////////////////////////////////////
{ SHiRemStrCodHtm(codInp, SHiRemOpn, SHiRemCls, "//", SHiQuo, "'") };
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna el codigo Html resultado de aplicar sintaxis realzada al codigo
Javascript de entrada que recibe como parametro pero sin enmarcar entre las
etiquetas <pre><code> y </code></pre>.",
SHiJscHtm);
////////////////////////////////////////////////////////////////////////////
```

Text SHiJscPre()

```
////////////////////////////////////////////////////////////////////////////
Text SHiJscPre(Text codInp) // Codigo de entrada Javascript
////////////////////////////////////////////////////////////////////////////
{ SHiPre(SHiJscHtm(codInp)) };
////////////////////////////////////////////////////////////////////////////
```

```
PutDescription(
"Retorna el codigo Html resultado de aplicar sintaxis realzada al codigo
Javascript de entrada que recibe como parametro.",
SHiJscPre);
////////////////////////////////////////////////////////////////////////////
```

Text SHiSqlPre()

```
////////////////////////////////////////////////////////////////////////////
Text SHiSqlPre(Text codInp) // Codigo de entrada Sql
////////////////////////////////////////////////////////////////////////////
{ SHiRemStrCodPre(codInp, "", "", "--", "'", "") };
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna el codigo Html resultado de aplicar sintaxis realzada al codigo
Sql de entrada que recibe como parametro.",
SHiSqlPre);
////////////////////////////////////////////////////////////////////////////
```

Text SHiGplPre()

```
////////////////////////////////////////////////////////////////////////////
Text SHiGplPre(Text codInp) // Codigo de entrada Gnuplot
////////////////////////////////////////////////////////////////////////////
{ SHiRemStrCodPre(codInp, "", "", "#", "'", "") };
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna el codigo Html resultado de aplicar sintaxis realzada al codigo
Gnuplot de entrada que recibe como parametro.",
SHiGplPre);
////////////////////////////////////////////////////////////////////////////
```

Text SHiCmdPre()

```
////////////////////////////////////////////////////////////////////////////
Text SHiCmdPre(Text codInp) // Codigo de entrada de mandatos Dos
////////////////////////////////////////////////////////////////////////////
{ SHiRemStrCodPre(codInp, "", "", "::", SHiQuo, "") };
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna el codigo Html resultado de aplicar sintaxis realzada al codigo
de mandatos para el Cmd de Dos que recibe como parametro.",
SHiCmdPre);
////////////////////////////////////////////////////////////////////////////
```

Text SHiCssPre()

```
////////////////////////////////////////////////////////////////////////////
Text SHiCssPre(Text codInp) // Codigo de entrada Cascade style sheet
////////////////////////////////////////////////////////////////////////////
{ SHiRemStrCodPre(codInp, SHiRemOpn, SHiRemCls, "", SHiQuo, "'") };
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna el codigo Html resultado de aplicar sintaxis realzada al codigo
Cascade style sheet que recibe como parametro.",
SHiCssPre);
////////////////////////////////////////////////////////////////////////////
```

Text SHiXmlHtm()

```
////////////////////////////////////////////////////////////////////////
Text SHiXmlHtm(Text codInp) // Codigo Xml
////////////////////////////////////////////////////////////////////////
{
  Set  setBlk = TxtSplitBy2Tag(codInp, "<!--", "-->"); // Romper x comments

  Set  setTag = BinGroup("<<", EvalSet(setBlk, Set(Text txtCod)
  {
    If(TxtBeginStrict(txtCod, "<!--"), [[txtCod]], // Era comentario
       TxtSplitBy2Fast(txtCod, "<", ">")) // Romper x etiquetas, no recursivo
  })); // De conjunto de conjuntos a conjunto lineal

  Set  setHtm = EvalSet(setTag, Text(Text txtCod)
  {
    Case(
      TxtBeginStrict(txtCod, "<!--"), // Comentario
        SHiRemHtm + HtmAsc2Xml(txtCod) + SHiEndHtm,

      Or(TxtBeginStrict(txtCod, "<"), TxtEndStrict(Compact(txtCod), ">")),
        SHiTagHtm + SHiJscHtm(txtCod) + SHiEndHtm, // Etiqueta

      TRUE,
        HtmAsc2Xml(txtCod))
  });
  SetSum(setHtm)
};
////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna el codigo Html resultado de aplicar sintaxis realzada al codigo
Xml que recibe como parametro pero sin enmarcar en las etiquetas
<pre><code> y </code></pre>
Emplea la version no recursiva TxtSplitBy2Fast(), aunque menos robusta que
TxtSplitBy2Tag(), para evitar caidas en fichero Xml de cierta longitud.",
SHiXmlHtm);
////////////////////////////////////////////////////////////////////////
```

Text SHiXmlPre()

```
////////////////////////////////////////////////////////////////////////
Text SHiXmlPre(Text codInp) // Codigo Xml
////////////////////////////////////////////////////////////////////////
{ SHiPre(SHiXmlHtm(codInp)) };
////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna el codigo Html resultado de aplicar sintaxis realzada al codigo
Xml que recibe como parametro.",
SHiXmlPre);
////////////////////////////////////////////////////////////////////////
```

Text SHiEmbHtm()

```
////////////////////////////////////////////////////////////////////////
Text SHiEmbHtm(Text codInp) // Codigo Html con Javscript y Tol embebido
////////////////////////////////////////////////////////////////////////
{
  Set  setTol = TxtSplitBy2Tag(codInp, "<"+"{", "}"+">"); // Tol embebido

  Set  setJsc = BinGroup("<<", EvalSet(setTol, Set(Text txtCod)
  {
    If(TxtBeginStrict(txtCod, "<"+"{"), [[txtCod]], // Es Tol embebido
       TxtSplitBy2Tag(txtCod, "<script", "</script>"))   // Javascript
  })); // De conjunto de conjuntos a conjunto lineal
```

```
  Set  setHtm = EvalSet(setJsc, Text(Text txtCod)
  {
    Case(
      TxtBeginStrict(txtCod, "<"+"{"), // Tol embebido
        SHiCodEmb + SHiTolHtm(txtCod) + SHiEndHtm,

      TxtBeginStrict(txtCod, "<script"), // Javascript embebido
        SHiCodEmb + SHiJscHtm(txtCod) + SHiEndHtm,

      TRUE, // Resto de codigo con tags Xml / Html
        SHiXmlHtm(txtCod))
  });
  SetSum(setHtm)
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna el codigo Html resultado de aplicar sintaxis realzada al codigo
Html basico que recibe como parametro pero sin enmarcar en las etiquetas
<pre><code> y </code></pre>.",
SHiEmbHtm);
////////////////////////////////////////////////////////////////////////////
```

Text SHiEmbPre()

```
////////////////////////////////////////////////////////////////////////////
Text SHiEmbPre(Text codInp) // // Codigo Html con Javscript y Tol embebido
////////////////////////////////////////////////////////////////////////////
{ SHiPre(SHiEmbHtm(codInp)) };
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna el codigo Html resultado de aplicar sintaxis realzada al codigo
Xml que recibe como parametro.",
SHiEmbPre);
////////////////////////////////////////////////////////////////////////////
```

Text SHiAscPre()

```
////////////////////////////////////////////////////////////////////////////
Text SHiAscPre(Text codInp) // // Fichero Ascii
////////////////////////////////////////////////////////////////////////////
{ SHiPre(HtmAsc2Xml(codInp)) };
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna el codigo Html del contenido Ascii que recibe como parametro.
Unicamente cambia los caracteres especiales con HtmAsc2Xml() y pone las
etiquetas <pre><code> y </code></pre>",
SHiAscPre);
////////////////////////////////////////////////////////////////////////////
```

Text SHiCompact()

```
////////////////////////////////////////////////////////////////////////////
Text SHiCompact(Text codHtm) // // Codigo Html ya autogenerado
////////////////////////////////////////////////////////////////////////////
{
  ReplaceTable(codHtm, [[ [["<pre><code> ",    "<pre><code>"]],
                          [["<pre><code>\n",   "<pre><code>"]],
                          [[" </code></pre>",  "</code></pre>"]],
                          [["\n</code></pre>", "</code></pre>"]] ]])
};
////////////////////////////////////////////////////////////////////////////
```

```
PutDescription(
"Retorna el codigo Html resultado de eliminar las lineas en blanco antes y
despues del <pre><code> y </code></pre>.",
SHiCompact);
////////////////////////////////////////////////////////////////////////////
```

Text SHiPreCode()

```
////////////////////////////////////////////////////////////////////////////
Text SHiPreCode(Text filPth, // Nombre o ruta del fichero para la extension
                Text filTxt) // Codigo ya leido del fichero, todo o parte
////////////////////////////////////////////////////////////////////////////
{
  Text filExt = ToLower(GetFileExtension(filPth));
  Text shiCod = Case(
    filExt == "tol",  SHiTolPre(filTxt), // Time Oriented Language
    filExt == "bat",  SHiCmdPre(filTxt), // Command bat
    filExt == "css",  SHiCssPre(filTxt), // Cascade style sheet
    filExt == "gpl",  SHiGplPre(filTxt), // Gnuplot
    filExt == "js",   SHiJscPre(filTxt), // Javascript
    filExt == "sql",  SHiSqlPre(filTxt), // Sql
    filExt == "xml",  SHiXmlPre(filTxt), // Xml
    filExt == "htm",  SHiEmbPre(filTxt), // Htm
    filExt == "html", SHiEmbPre(filTxt), // Html
    filExt == "age",  SHiEmbPre(filTxt), // Html
    TRUE,             SHiAscPre(filTxt)  // Ascii, limpiar y con pre y code
  );
  SHiCompact(shiCod)
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna el codigo Html resultado de aplicar sintaxis realzada al codigo
filTxt que se la leido, total o parcialmente, de fichero filPth.
Recibe el nombre o ruta del fichero para saber su extension y de ahi el
lenguaje de programacion.",
SHiPreCode);
////////////////////////////////////////////////////////////////////////////
```

Text SHiPreFile()

```
////////////////////////////////////////////////////////////////////////////
Text SHiPreFile(Text filPth) // Ruta del fichero de entrada
////////////////////////////////////////////////////////////////////////////
{ SHiPreCode(filPth, ReadFile(filPth)) };
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna el codigo Html resultado de aplicar sintaxis realzada a todo el
codigo del fichero que recibe como parametro.",
SHiPreFile);
////////////////////////////////////////////////////////////////////////////
```

inc.tol de SHi.SyntaxHighlight

Inclusion de ficheros de funciones comunes y basicas y de funciones especificas de aplicacion

Declaraciones

Inclusiones comunes

- `Set txtInc`
  Incluir funciones de textos.
- `Set htmInc`
  Incluir funciones de Html.

Inclusiones de aplicación

- `Set shiInc`
  Incluir funciones de sintaxis realzada.

Set txtInc

```
////////////////////////////////////////////////////////////////////////////
Set  txtInc = Include("cmm/txt.tol");
////////////////////////////////////////////////////////////////////////////
PutDescription("Incluir funciones de textos.", txtInc);
////////////////////////////////////////////////////////////////////////////
```

Set htmInc

```
////////////////////////////////////////////////////////////////////////////
Set  htmInc = Include("cmm/htm.tol");
////////////////////////////////////////////////////////////////////////////
PutDescription("Incluir funciones de Html.", htmInc);
////////////////////////////////////////////////////////////////////////////
```

Set shiInc

```
////////////////////////////////////////////////////////////////////////////
Set  shiInc = Include("app/shi.tol");
////////////////////////////////////////////////////////////////////////////
PutDescription("Incluir funciones de sintaxis realzada.", shiInc);
////////////////////////////////////////////////////////////////////////////
```

gpl.tol de SHi.SyntaxHighlight

Funciones para crear graficos polares en estrella con Gnuplot, son especificas de esta aplicacion, no son generales. Depende del directorio de instalacion de Gnuplot.

Declaraciones

Constantes

- `Text GplExe`
  Ruta directorios donde esta instalado Gnuplot, depende de la instalacion. Notese la necesidad de comillas dentro de comillas.
- `Text GplPolar2x20`
  Prefijo para los ficheros Gnuplot de programacion (.gpl) y datos (.dar).
- `Text GplDat`
  Plantilla de datos especifica para crear de graficos polares en estrella. Permite generar graficos de 2 series de datos de 20 preguntas, cada 18º, con respuestas en el rango del 0 al 5. Los datos son etiquetas que seran reemplazados por los valores que correspondan en cada llamada.

Funciones

- `Real PdfPolar2x20(Text outPth, Set repTab)`
  Retorna TRUE si puede crear el fichero grafico Png de ruta outPth.

Constantes

Text GplExe

```
////////////////////////////////////////////////////////////////////////////
Text GplExe = Q(FilDos(Q("%BIN%/gnuplot/bin/pgnuplot")));
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Ruta directorios donde esta instalado Gnuplot, depende de la instalacion.
Notese la necesidad de comillas dentro de comillas.",
GplExe);
////////////////////////////////////////////////////////////////////////////
```

Text GplPolar2x20

```
////////////////////////////////////////////////////////////////////////////
Text GplPolar2x20 = "polar2x20";
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Prefijo para los ficheros Gnuplot de programacion (.gpl) y datos (.dar).",
GplPolar2x20);
////////////////////////////////////////////////////////////////////////////
```

Text GplDat

```
////////////////////////////////////////////////////////////////////////////
Text GplDat =
" 0, a01, c01
 18, a02, c02
 36, a03, c03
 54, a04, c04
 72, a05, c05
 90, a06, c06
108, a07, c07
126, a08, c08
```

```
144, a09, c09
162, a10, c10
180, a11, c11
198, a12, c12
216, a13, c13
234, a14, c14
252, a15, c15
270, a16, c16
288, a17, c17
306, a18, c18
324, a19, c19
342, a20, c20
360, a01, c01
";
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Plantilla de datos especifica para crear de graficos polares en estrella.
Permite generar graficos de 2 series de datos de 20 preguntas, cada 18º,
con respuestas en el rango del 0 al 5.
Los datos son etiquetas que seran reemplazados por los valores que
correspondan en cada llamada.",
GplDat);
////////////////////////////////////////////////////////////////////////////
```

Real PdfPolar2x20()

```
////////////////////////////////////////////////////////////////////////////
Real PdfPolar2x20(Text outPth, // Ruta del fichero png de salida
                  Set  repTab) // Tabla de reemplazamiento con datos a pintar
////////////////////////////////////////////////////////////////////////////
{
  Text seePth = CtrDir+"/semilla/"+
                GplPolar2x20+".see"; // Semilla de programacion Gnuplot
  Text gplPth = GplPolar2x20+".gpl"; // Programacion Gnuplot
  Text datPth = GplPolar2x20+".dat"; // Fichero de datos para Gnuplot

  Text WriteFile(datPth, ReplaceTable(GplDat, repTab)); // Escribe datos
  Text seeTxt = ReadFile(seePth); // Lee la semilla de programacion
  Text WriteFile(gplPth, Replace(seeTxt,"web/shi_syntaxhighlight/gpl_tol.html"

  Text cmdTxt = GplExe+" "+gplPth;   // Ejecutable Gnuplot y programa .gpl

  Real cmdExe = System(cmdTxt);
  Text cmdMsg = "    "+If(cmdExe, "Plot OK", "Plot ERROR")+" -> ";
  Text WriteLn(cmdMsg+outPth);
  cmdExe
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna TRUE si puede crear el fichero grafico Png de ruta outPth.",
PdfPolar2x20);
////////////////////////////////////////////////////////////////////////////
```

sql.tol de SHi.SyntaxHighlight

Funciones con los queries basicos de la aplicacion

Declaraciones

Funciones

- `Text SqlLitNam(Text ctrCod, Text domCod, Text anyCod)`
  Retorna el nombre de algo dado un periodo, su dominio y su codigo.
- `Text SqlResAnd(Text ctrCod, Text prgCod, Text gruCod, Text curCod, Text prfCod, Text asiCod, Text preCod)`
  Retorna una serie de condiciones Sql para la tabla Respuestas enlazadas con el operador and. Aquellos argumentos de entrada cuyo valor sea nulo no apareceran en la serie de condiciones.
- `Set SqlResLst(Text ctrCod, Text prgCod, Text gruCod, Text curCod, Text prfCod, Text asiCod, Text preCod)`
  Retorna la lista de respuestas que cumplen unas determinadas condiciones de control, programa, grupo, curso, profesor, asignatura y pregunta. Aquellos campos para los que su codigo sea nulo quedan libres en el query. El campo Respuesta es de texto pues se admite la respuesta ?, pero en este querie el campo Respuesta se convierte a entero CInt().

Text SqlLitNam()

```
////////////////////////////////////////////////////////////////////////////
Text SqlLitNam(Text ctrCod, // Control, periodo en el que se realiza
               Text domCod, // Dominio
               Text anyCod) // Codigo de para localizar el nombre
////////////////////////////////////////////////////////////////////////////
{
  Text sqlTxt = "
    select Etiqueta
    from   Literal
    where
      Control = '"+ctrCod+"' and "+
    " Dominio = '"+domCod+"' and "+
    " Codigo  = '"+anyCod+"'; ";

  Set sqlSet = DBTable(sqlTxt);

  If(EQ(Card(sqlSet),1), sqlSet[1][1], "ERROR") // Solo puede haber 1
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna el nombre de algo dado un periodo, su dominio y su codigo.",
SqlLitNam);
////////////////////////////////////////////////////////////////////////////
```

Text SqlResAnd()

```
////////////////////////////////////////////////////////////////////////////
Text SqlResAnd(Text ctrCod, // Control, periodo
```

```
                Text prgCod, // Codigo de programa (opcional)
                Text gruCod, // Codigo de grupo (opcional)
                Text curCod, // Codigo de curso (opcional)
                Text prfCod, // Codigo del profesor (opcional)
                Text asiCod, // Codigo de la asignatura (opcional)
                Text preCod) // Codigo de pregunta
////////////////////////////////////////////////////////////////////////////
{
  Text ctrQry = If(ctrCod=="", "", "Control    = '"+ctrCod+"'");
  Text prgQry = If(prgCod=="", "", "Programa   = '"+prgCod+"'");
  Text gruQry = If(gruCod=="", "", "Grupo      = '"+gruCod+"'");
  Text curQry = If(curCod=="", "", "Curso      = '"+curCod+"'");
  Text prfQry = If(prfCod=="", "", "Profesor   = '"+prfCod+"'");
  Text asiQry = If(asiCod=="", "", "Asignatura = '"+asiCod+"'");
  Text preQry = If(preCod=="", "", "Pregunta   = '"+preCod+"'");

  Set  wheSet = [[ctrQry, prgQry, gruQry, curQry, prfQry, asiQry, preQry]];
  Set  wheSel = Select(wheSet, Real(Text txtQry) { txtQry != "" });

  Set2Txt(wheSel, "", "", " and ", " and ", "", "", "", "")
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna una serie de condiciones Sql para la tabla Respuestas enlazadas con
el operador and.
Aquellos argumentos de entrada cuyo valor sea nulo no apareceran en la
serie de condiciones.",
SqlResAnd);
////////////////////////////////////////////////////////////////////////////
```

Set SqlResLst()

```
////////////////////////////////////////////////////////////////////////////
Set SqlResLst(Text ctrCod, // Control, periodo
              Text prgCod, // Codigo de programa (opcional)
              Text gruCod, // Codigo de grupo (opcional)
              Text curCod, // Codigo de curso (opcional)
              Text prfCod, // Codigo del profesor (opcional)
              Text asiCod, // Codigo de la asignatura (opcional)
              Text preCod) // Codigo de pregunta
////////////////////////////////////////////////////////////////////////////
{
  Text wheQry = SqlResAnd(ctrCod,prgCod,gruCod,curCod,prfCod,asiCod,preCod);

  Text sqlTxt = "
    select CInt(Respuesta)
    from   Respuesta
    where
      Respuesta >= '1' and
      Respuesta <= '5' and
    " + wheQry + ";";

//Text WriteLn(sqlTxt);

  Set sqlSet = DBTable(sqlTxt);

  If(Card(sqlSet), Traspose(sqlSet)[1], Empty)
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna la lista de respuestas que cumplen unas determinadas condiciones de
control, programa, grupo, curso, profesor, asignatura y pregunta.
Aquellos campos para los que su codigo sea nulo quedan libres en el query.
El campo Respuesta es de texto pues se admite la respuesta ?,
pero en este querie el campo Respuesta se convierte a entero CInt().",
SqlResLst);
////////////////////////////////////////////////////////////////////////////
```