Strings en Delphi 2010

Unicode

- A partir de la versión 2009, Delphi incorporó el soporte para caracteres UNICODE.
- Las aplicaciones en general utilizaban un alfabeto de 26 caracteres.
- Con UNICODE pueden representarse todos los lenguajes.

¿Qué es UNICODE?

- Es una representación numérica **única** para cada carácter sin importar el idioma, la plataforma o el programa.
- Adoptado por Apple, HP, IBM, JustSystem, Microsoft, Oracle, SAP, Sun, Sybase, Unisys y otros …

www.unicode.org

UTF (Unicode Transformation Format)

- Un mismo número puede representarse de distintas formas.
- El estándar define tres formatos dependiendo de la cantidad de bits usados para representar la parte inicial del conjunto
  - **UTF-8**, **UTF-16** y **UTF-32.**

Lo más fácil hubiera sido que todos utilizaran 4 bytes pero esto busca ahorrar memoria y tiempo de procesamiento.

Tipos CHAR

Delphi soporta

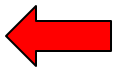
- AnsiChar : representación de 8 bits
  - **256 símbolos distintos** interpretados según la página de código (**code page**) indicada.
- WideChar: representación de 16 bits
  - **64K símbolos distintos**

El tipo **Char** se interpreta como **WideChar**

Char como tipo ordinal

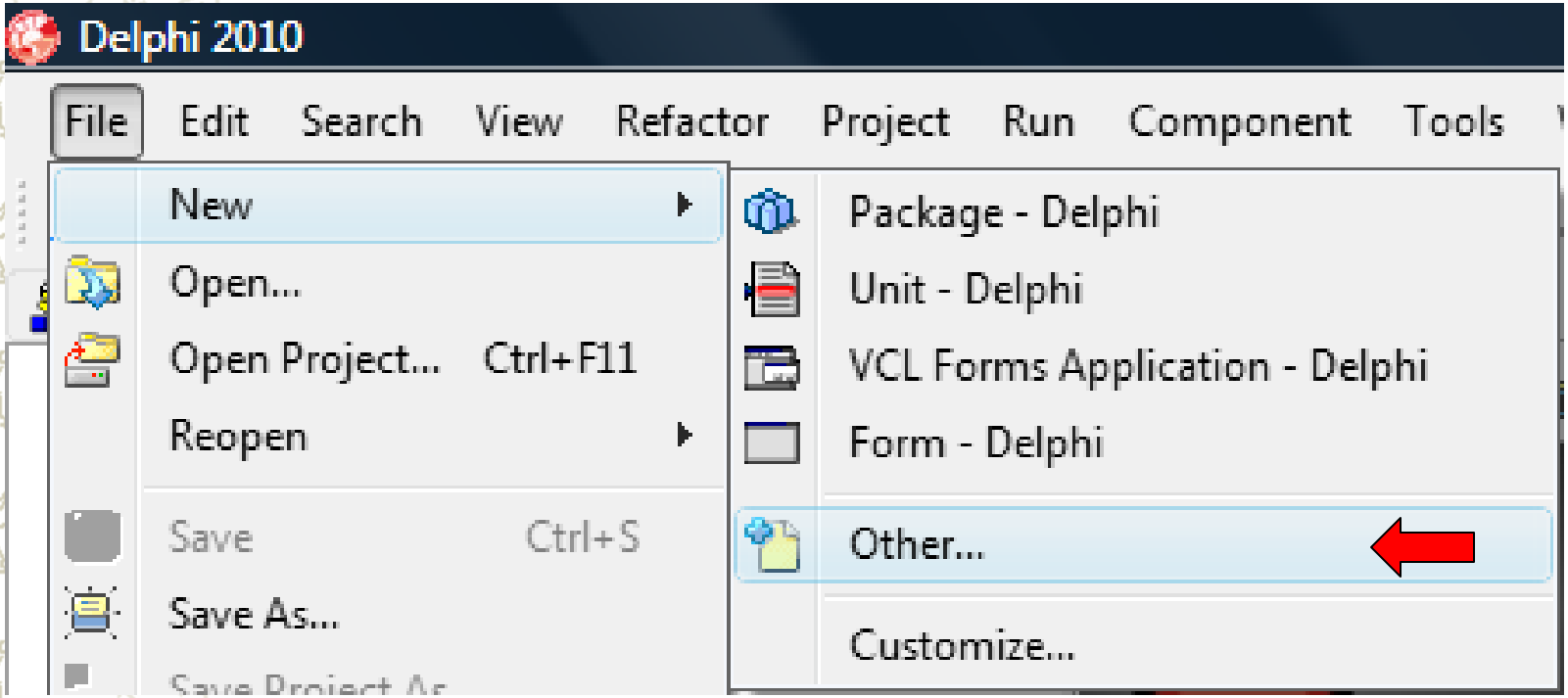
- El tipo Char sigue siendo un ordinal. Puede utilizarse **inc**, **dec** o como índice en el **for**.

```
Var ch : Char;
Begin
    ch := 'a';
    inc(ch, 100);
    for ch := #90 to #95 do
        writeln(ch);
end
```

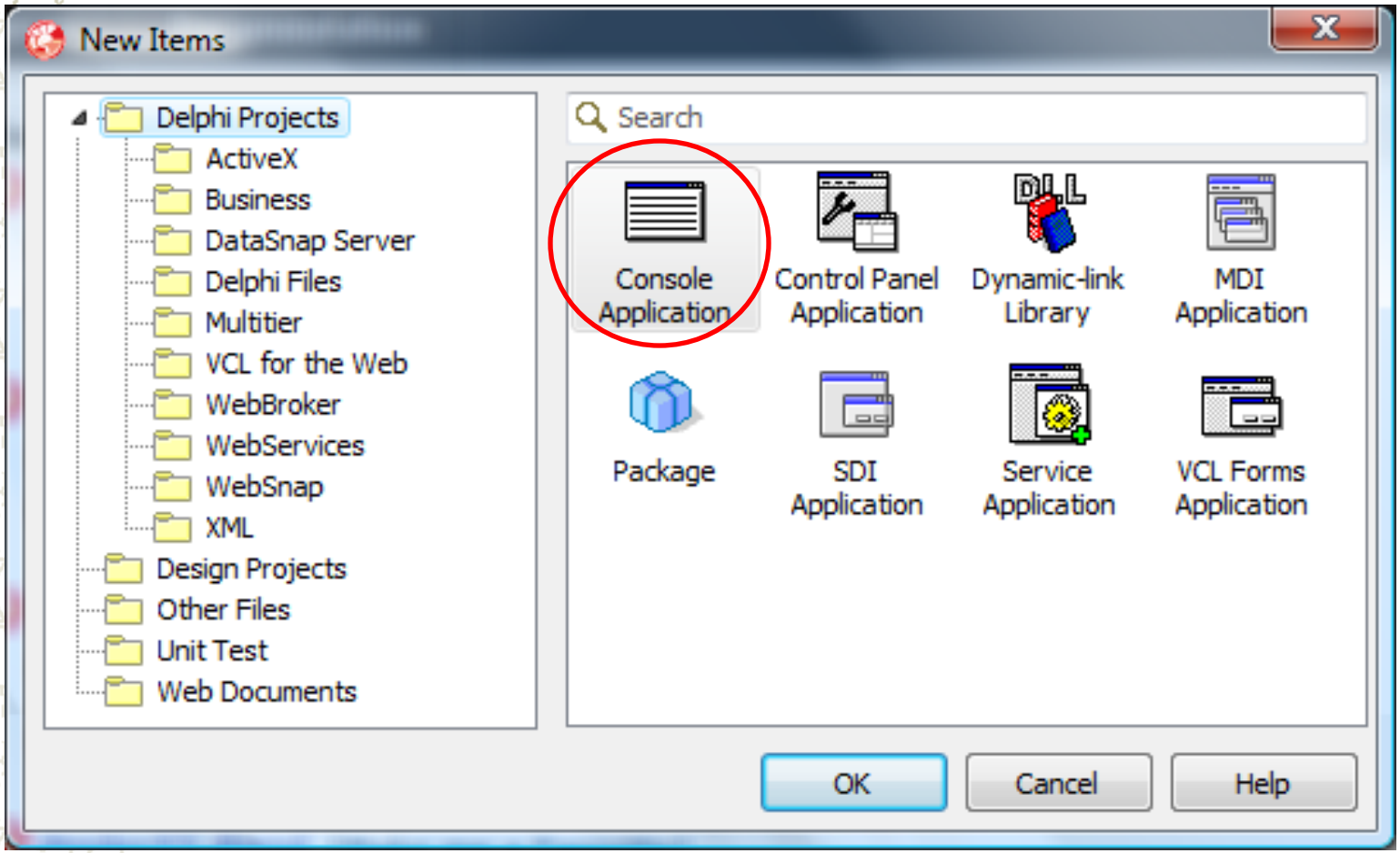


Sintaxis válida en una **aplicación de consola**

Aplicación de Consola



Aplicación de Consola



WideChar.dpr

```
program wideChar01;
{$APPTYPE CONSOLE}
uses SysUtils;
Var ch : Char;
Begin
        ch := 'a';
        inc(ch, 100);
        for ch := #90 to #95 do
                writeln(ch);
end.
```

Set of Char

WideChar02.dpr

```
Var CharSet : Set of Char;
begin
   CharSet := ['a', 'b', 'c'];
   if 'a' in CharSet  then
      writeln('pertenece');
end.
```

W1050: WideChar reduced to byte char in set expressions. Consider using ‘CharInSet’ function in ‘SysUtils’ unit.

Set of Char

WideChar03.dpr

```
Var CharSet : Set of AnsiChar;
begin
   CharSet := ['a', 'b', 'c'];
   if AnsiChar('a‘) in CharSet  then
      writeln('pertenece');
end.
```

De esta forma se eliminan los warning

Tipos de Strings

- Delphi posee los siguientes tipos de datos string
  - **ShortString** (los usados en Programación)
  - **AnsiString** : Usa caracteres de 8 bits.
  - **UnicodeString**
  - **WideString**

  } Usa caracteres Unicode (16 bits)

La palabra reservada **String** funciona como un identificador de tipo genérico que por defecto se interpreta como **UnicodeString**

Tipos de Strings

- Delphi posee los siguientes tipos de datos string
  - **ShortString** (los usados en Programación)
  - **AnsiString**
  - **UnicodeString**
  - **WideString**

Son dinámicos. Pueden tener hasta 2GB de longitud

La palabra reservada **String** funciona como un identificador de tipo genérico que por defecto se interpreta como **UnicodeString**

Tipos de Strings

- Los elementos de un string pueden ser accedidos como un arreglo de caracteres con índice entre 1 y la cant.máxima de caracteres que contiene.

```
var MiString : string;
begin
   MiString := 'Esto es un ejemplo';
   MiString[4] := 'e';
   writeln( MiString );
end;
```

¿Qué imprime?

Tipos de Strings

- Los elementos de un string pueden ser accedidos como un arreglo de caracteres con índice entre 1 y la cant.máxima de caracteres que contiene.

```
var MiString : string;
begin
   MiString := 'Esto es un ejemplo';
   MiString[4] := 'e';
   writeln( MiString );
end;
```

Puede reemplazarlo por ShortString, AnsiString, UnicodeString o WideString

Reference Counting

- Los tipos
  - **AnsiString**
  - **UnicodeString**

  utilizan una representación basada en un contador de referencia.
- Esto no ocurre con el tipo **WideString**.

Reference Counting

- **Variables AnsiString y UnicodeString**
  - Utiliza un puntero de 4 bytes.
  - Cuando la variable está vacía vale **nil** y el string no ocupa lugar en memoria.
  - Cuando la variable no está vacía, apunta a un área de memoria dinámicamente alocada.

    Esta memoria se reserva en la heap. Su manejo es automático y no requiere código adicional.

Reference Counting

- Representación

-12	-10	-8	-4	Valor
Code Page	Tamaño del elemento	Contador de Referencia	Longitud del String	Contenido del String

Para distinguir AnsiString (8 bits) de UnicodeString (16 bits)

Información

- Pueden utilizarse las siguientes funciones para obtener información de estas variables
  - **StringCodePage** : el valor por defecto 1200
  - **StringElementSize** : el valor por defecto es 2
  - **StringRefCount**
  - **Length**

UnicodeString01.dpr

Reference Counting

- Como las variables de tipo String (Ansi y Unicode) son punteros, dos o más de ellos pueden referenciar el mismo valor sin consumir memoria adicional.
- Cuando una variable es destruida o se le asigna un nuevo valor, el contador de referencia del viejo string (el valor previo de la variable) es decrementado y el contador de referencia del nuevo valor (si lo hay) es incrementado.
- Si el contador de referencia de un string llega a cero, su memoria es liberada.
- Este proceso se llama ***reference-counting***.

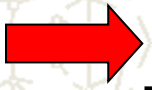
Ejemplo

```
Var Tex1, Tex2, Tex3 : String;
Begin
      Tex1 := 'Ejemplo';

      Tex2 := Tex1;
      Tex3 := Tex2;

      Tex2 := 'Nuevo';

      Tex2 := 'Otro texto';
end;
```



Memoria HEAP

Ejemplo

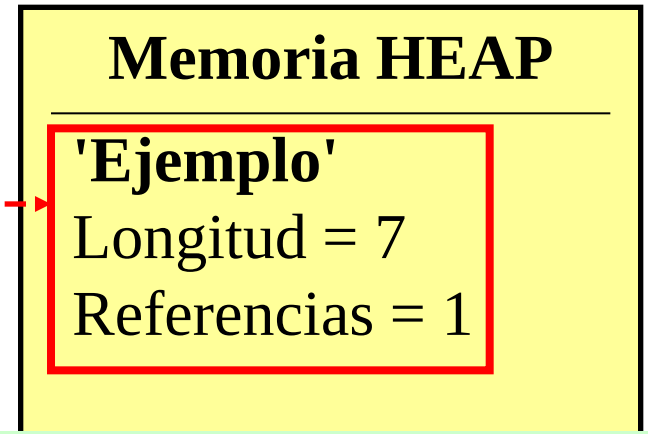
```
Var Tex1, Tex2, Tex3 : String;
Begin

     Tex1 := 'Ejemplo';

     Tex2 := Tex1;
     Tex3 := Tex2;

     Tex2 := 'Nuevo';

     Tex2 := 'Otro texto
end;
```



Note que **Tex1** no contiene el string sino un **puntero** al lugar donde está almacenado

Ejemplo

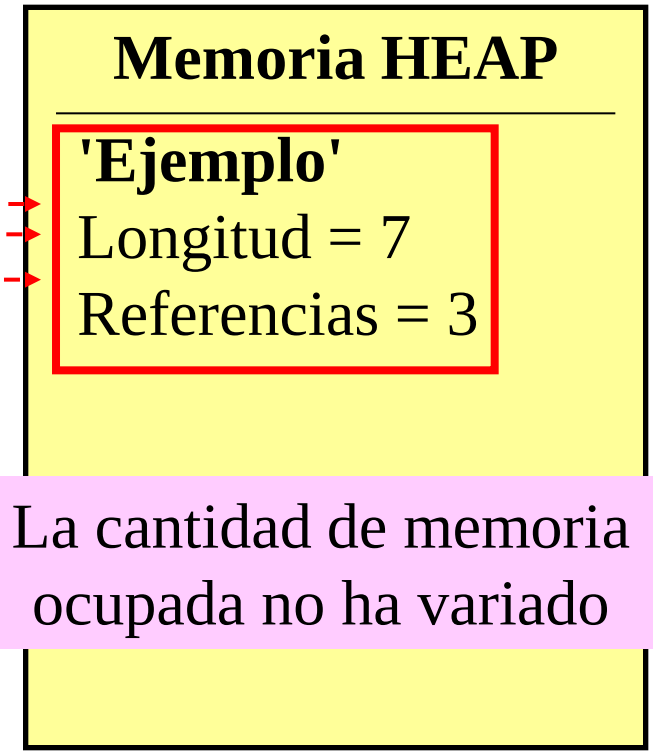
```
Var Tex1, Tex2, Tex3 : String;
Begin

     Tex1 := 'Ejemplo';

     Tex2 := Tex1;
     Tex3 := Tex2;

     Tex2 := 'Nuevo';

     Tex2 := 'Otro texto';
end;
```



La cantidad de memoria ocupada no ha variado

Ejemplo

```
Var Tex1, Tex2, Tex3 : String;
Begin

     Tex1 := 'Ejemplo';

     Tex2 := Tex1;
     Tex3 := Tex2;

     Tex2 := 'Nuevo';

     Tex2 := 'Otro texto';
end;
```

Memoria HEAP

'Ejemplo'
Longitud = 7
Referencias = 2

'Nuevo'
Longitud = 5
Referencias = 1

Ejemplo

```
Var Tex1, Tex2, Tex3:String;
Begin

     Tex1 := 'Ejemplo';

     Tex2 := Tex1;
     Tex3 := Tex2;

     Tex2 := 'Nuevo';

     Tex2 := 'Otro texto';

end;
```

Memoria HEAP

'Ejemplo'
Longitud = 7
Referencias = 2

'Nuevo'
Longitud = 5
Referencias = 0

'Otro texto'
Longitud = 10
Referencias = 1

Declarando e inicializando Strings

- La siguiente instrucción declara un UnicodeString:

```
S: string;
```

- Estos strings son inicializados automáticamente con el valor nulo.

- Puede utilizar la variable EmptyStr para testear si el string está vacío

```
if S = EmptyStr then
```

  - `if S = '' then`

Declarando e inicializando Strings

- Un string vacío no tiene un valor válido. Intentar indexar un string vacío es equivalente a acceder a una posición de memoria inexistente y producirá un error.

```
var
   S: string;
begin
   S[i];     // error de acceso
   // sentencias
end;
```

Declarando e inicializando Strings

- La asignación de un valor string constante (o cualquier expresión que retorne un string) a una variable modificará su longitud dinámicamente.
- Ejemplos

```
MyString := 'Hello world!';
MyString := 'Hello ' + 'world';
MyString := MyString + '!';
MyString := ' ';          { un blanco }
MyString := '';           { string vacío }
```

Dimensionando Strings

- La función **Length** retorna la cant.de caracteres del string.
- El procedure **SetLength** ajusta la longitud del string.

```
S := 'No es nulo';

//setea dinámicamente la long.de S a 100
SetLength(S, 100);
```

- SetLength preserva los caracteres existentes en el string pero el contenido del nuevo espacio es indefinido.
- Luego del SetLength, S es la única referencia al string, es decir, es un string cuyo contador de referencia tiene valor 1.

Declarando strings

- Recuerde que declarar un string como

```
var S: string[n];
```

declara implícitamente un ShortString, no un UnicodeString de longitud n.

- Un UnicodeString de longitud n se declara así:

```
var S: String;
begin
    SetLength(S, n);
```

Indique las similitudes y diferencias de estas declaraciones

Longitud de un ShortString

```
program Produce;
var
   str : String[30];
   len : Integer;
begin
   str := 'texto';
   len := Ord(str[0]);
   writeln('La longitud de str es ',len);
end.
```

El tipo **ShortString** agrega un byte inicial (posición cero) para guardar la longitud.

Elem.0 inaccesible en String

```
program Produce;
var
   str : String;
   len : Integer;
begin
 str := 'texto';
 len := ord(str[0])
end.
```

ERROR!

```
program Produce;
var
   str : String;
   len : Integer;
begin
   str := 'texto';
   len := Length(str);
end.
```

Los Strings NO poseen el elemento 0 conteniendo la longitud.

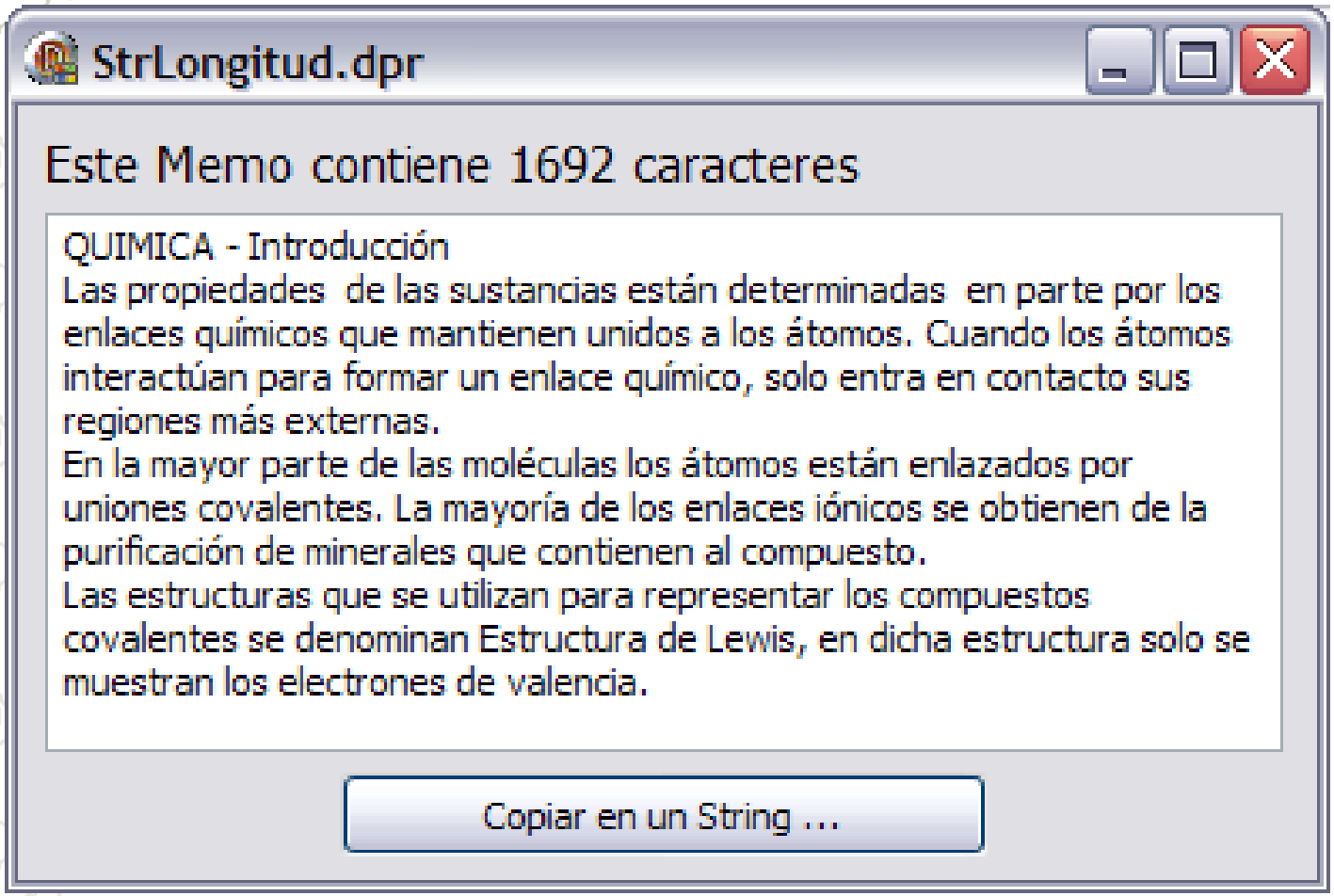
Ejercicio : ¿Qué imprime?

La función **SizeOf** retorna el tamaño de una variable en bytes

```
Var Str1 : String;
    Str2 : String[50];
    Str3 : ShortString;
begin
  writeln('Tamaño en bytes de Str1 =',SizeOf(Str1));
  writeln('Tamaño en bytes de Str2 =',SizeOf(Str2));
  writeln('Tamaño en bytes de Str3 =',SizeOf(Str3));

  SetLength(Str1, 20);
  writeln('Tamaño en bytes de Str1 =',SizeOf(Str1));
end;
```

Ejercicio


StrLongitud.dpr
Este Memo contiene 1692 caracteres
QUIMICA - Introducción
Las propiedades de las sustancias están determinadas en parte por los enlaces químicos que mantienen unidos a los átomos. Cuando los átomos interactúan para formar un enlace químico, solo entra en contacto sus regiones más externas.
En la mayor parte de las moléculas los átomos están enlazados por uniones covalentes. La mayoría de los enlaces iónicos se obtienen de la purificación de minerales que contienen al compuesto.
Las estructuras que se utilizan para representar los compuestos covalentes se denominan Estructura de Lewis, en dicha estructura solo se muestran los electrones de valencia.
Copiar en un String ...

```
procedure TForm4.Button1Click(Sender: TObject);
Var Str1 : ShortString;
    Str2 : string[200];
    Str3 : AnsiString;
    Str4,Str5 : String;
begin
   setLength(Str5, 300);
   Str1 := Memo1.text;
   Str2 := Str1;
   Str3 := Memo1.text;
   Str4 := Str3;
   Str5 := Memo1.text;
   ShowMessage(IntToStr(length(Str1))+' - '+
               IntToStr(length(Str2))+' - '+
               IntToStr(length(Str3))+' - '+
               IntToStr(length(Str4))+' - '+
               IntToStr(length(Str5)));
end;
```

Length(Memo1.text) = 1692

¿Se produce algún tipo de error?

```
procedure TForm4.Button1Click(Sender: TObject);
Var Str1 : ShortString;
    Str2 : string[200];
    Str3 : AnsiString;
    Str4,Str5 : String;
begin
   setLength(Str5, 300);
   Str1 := Memo1.text;
   Str2 := Str1;
   Str3 := Memo1.text;
   Str4 := Str3;
   Str5 := Memo1.text;
   ShowMessage(IntToStr(length(Str1))+' - '+
   IntToStr(StringElementSize(Str2)*length(Str2))+'-'+
   IntToStr(StringElementSize(Str3)*length(Str3))+'-'+
   IntToStr(StringElementSize(Str4)*length(Str4))+'-'+
   IntToStr(StringElementSize(Str5)*length(Str5)));
end;
```

Length(Memo1.text) = 1692

Ejercicio : ¿Qué imprime?

```
Var TextoA, TextoB : String;
begin
    TextoA := 'Primer Valor';
    TextoB := TextoA;
    TextoA := 'Segundo Valor';
    Writeln(TextoA, ' - ', TextoB);
end;
```

¿Por qué, si TextoA y TextoB son punteros, no comparten el mismo texto?

Ejercicio : ¿Qué imprime?

```
Var Texto : ShortString;
    Cadena : String[100];
begin
   Texto := 'Una línea';
   writeln(ord(Texto[0]));
   writeln(length(Texto));

   Texto[0] := Char(25);
   writeln(length(Texto));

   SetLength(Texto, 500);
   writeln(length(Texto));
end;
```

9

25

244 (500 mod 256)