

Excepciones

- ✖ La información que se ingresa al programa siempre es un string.
- ✖ Cuando se trata de valores numéricos deben ser convertidos antes de ser usados.
- ✖ Los procesos de conversión producen error si el texto ingresado no contiene un número.
- ✖ ¿Cómo se valida?

Ejemplo Calcu_Simple.dpr

- Implementar una calculadora sencilla que permita sumar dos números enteros.

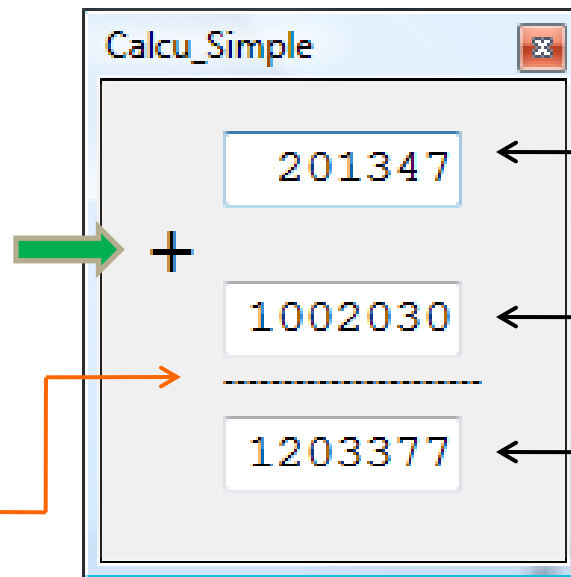
TSpeedButton
(paleta Additional)

Propiedades

Caption

Flat

TLabel
(paleta Standard)



TEdit
(paleta Standard)

Ejemplo : OnClick del botón

```
Procedure TForm1.SpeedButton1Click(Sender: TObject );  
Var Nro1, Nro2 : Integer;  
begin  
    Nro1 := StrToInt(Edit1.Text);  
    Nro2 := StrToInt(Edit2.Text);  
    Edit3.text := IntToStr(Nro1 + Nro2 );  
end;
```

Ver que si Edit1 y/o Edit2 no contienen un número, la función StrToInt dará error y el programa abortará.

Seguimiento del programa

```
30 procedure TForm1.SpeedButton1Click(Sender: TObject);  
  var nro1,nro2:integer;  
  begin  
    nro1 := StrToInt(Edit1.text);  
    nro2 := StrToInt(Edit2.text);  
    edit3.text := Int Edit2.text : 'HOLA' ;  
  end;
```

Puede introducir un punto de parada en el programa clickeando sobre los puntos azules (ver F4 y F7).

Muestra el valor de propiedades y variables

¿Qué creen que va a pasar?

Manejo de Excepciones

- ✱ Una **excepción** es un evento que ocurre por un error del programa en tiempo de ejecución y es generada para indicarle al programador que han ocurrido errores que impiden la normal ejecución del programa (ej: división por cero).
- ✱ El proceso que permite resolver los problemas generados por una excepción se denomina el **manejador** de dicha excepción (ej: qué hacer cuando aparece la división por cero).

Manejo de Excepciones

- ✖ Desventajas al NO utilizar manejadores de excepciones:
 - Oscurece el código desarrollado, ya que se mezclan las instrucciones dedicadas a resolver el problema con las requeridas para evitar los errores.
 - Disminuye la eficiencia del código pues las validaciones son realizadas independientemente de si se trata de una situación de error o no.

Manejo de Excepciones

✶ En Pascal

```
if Z <> 0 then
    X := Y / Z
else ResolverDivisionPorCero;
```

Hay que leer el código para saber que parte resuelve el problema y que parte resuelve el error

✶ En Delphi

```
try
    X := Y/Z;
except
    on EZeroDivide do ResolverDivisionPorCero;
end;
```

Código libre de errores

Manejador que resuelve el problema

Manejo de Excepciones

✶ En Pascal

```
if Z <> 0 then
```

```
    X := Y / Z
```

```
else ResolverDivisionPorCero;
```

La condición $Z \neq 0$
se evalúa siempre

✶ En Delphi

```
try
```

```
    X := Y/Z;
```

```
except
```

```
    on EZeroDivide do ResolverDivisionPorCero;
```

```
end;
```

Este es el único código que
se ejecuta. No se evalúa
ninguna condición.

Sintaxis try- except

Try

{ Bloque de instrucciones que se desea proteger }

except

on (Tipo de excepción 1) **do**
Manejador_Excepcion_1;

on (Tipo de excepción 2) **do**
Manejador_Excepcion_2;

...

else Manejador_para_las_demas;

end;

Tipos de excepciones predefinidos

Excepción	Descripción
<i>EAccessViolation</i>	Acceso a memoria inválido
<i>EConvertError</i>	Error en la conversión de un string u objeto.
<i>EDivByZero</i>	División entera por cero.
<i>EIntOverflow</i>	El entero calculado es demasiado grande.
<i>EInvalidGraphic</i>	Se ha intentado trabajar con un formato de archivo gráfico desconocido.
<i>ERangeError</i>	Error de rango.
<i>EZeroDivide</i>	División de punto flotante por cero.

Ejemplo Calcu_Simple.dpr

```
procedure TForm1.BitBtn1Click(Sender: TObject);
Var Nro1, Nro2 : Integer;
begin
  Try
    Nro1 := StrToInt(Edit1.Text);
    Nro2 := StrToInt(Edit2.Text);
    Edit3.text := IntToStr(Nro1 + Nro2 );
  except
    on EConvertError do begin
      Edit3.text := '';
      ShowMessage('Operandos Inválidos');
    end;
  end;
end;
```

Funciones de Conversión

✱ Dado un número lo convierten en un string

- **IntToStr**

- **FloatToStr** ←

Para que la calculadora
opere con valores reales

✱ Dado un string lo convierten en un número

- **StrToInt**

- **StrToFloat** ←

Strings en Pascal

- ✚ Revisemos los procesos definidos en Pascal para manejo de Strings
- ✚ Funciones
 - Length, Copy y Pos
- ✚ Procedimiento
 - Delete

Función Length

- ✱ Retorna la cantidad de caracteres que contiene el string

```
Var linea : string;  
    long   : integer;  
begin  
    linea := 'Esto es un ejemplo.';  
    long  := length(linea);
```

El valor de **long** será 19

Función Copy

↓ Sintaxis

COPY(String, inicio, cantidad)

retorna un substring de longitud **cantidad**
formado por los caracteres de **String**
comenzando en la posición indicada por **inicio**
(inclusive).

Función Copy

✶ Retorna un substring de un string dado

```
Var linea : string;  
miniLinea : string;  
begin  
  linea := 'Esto es un ejemplo.';  
  miniLinea := copy(linea, 4, 6);
```

El valor de **miniLinea** será 'o es u'

Función POS

✱ Retorna la posición de un substring dentro de un string dado. Si no lo encuentra devuelve cero.

```
Var linea : string;  
    ubicacion : integer;  
begin  
    linea := 'Esto es un ejemplo.';  
    ubicacion := POS('e', linea);
```

El valor de **ubicacion** será **6**

Procedimiento Delete

↓ Sintaxis

DELETE(String, inicio, cantidad)

Borra de **String** tantos caracteres como indica **cantidad** a partir de la posición indicada por **inicio** (inclusive).

Procedimiento Delete

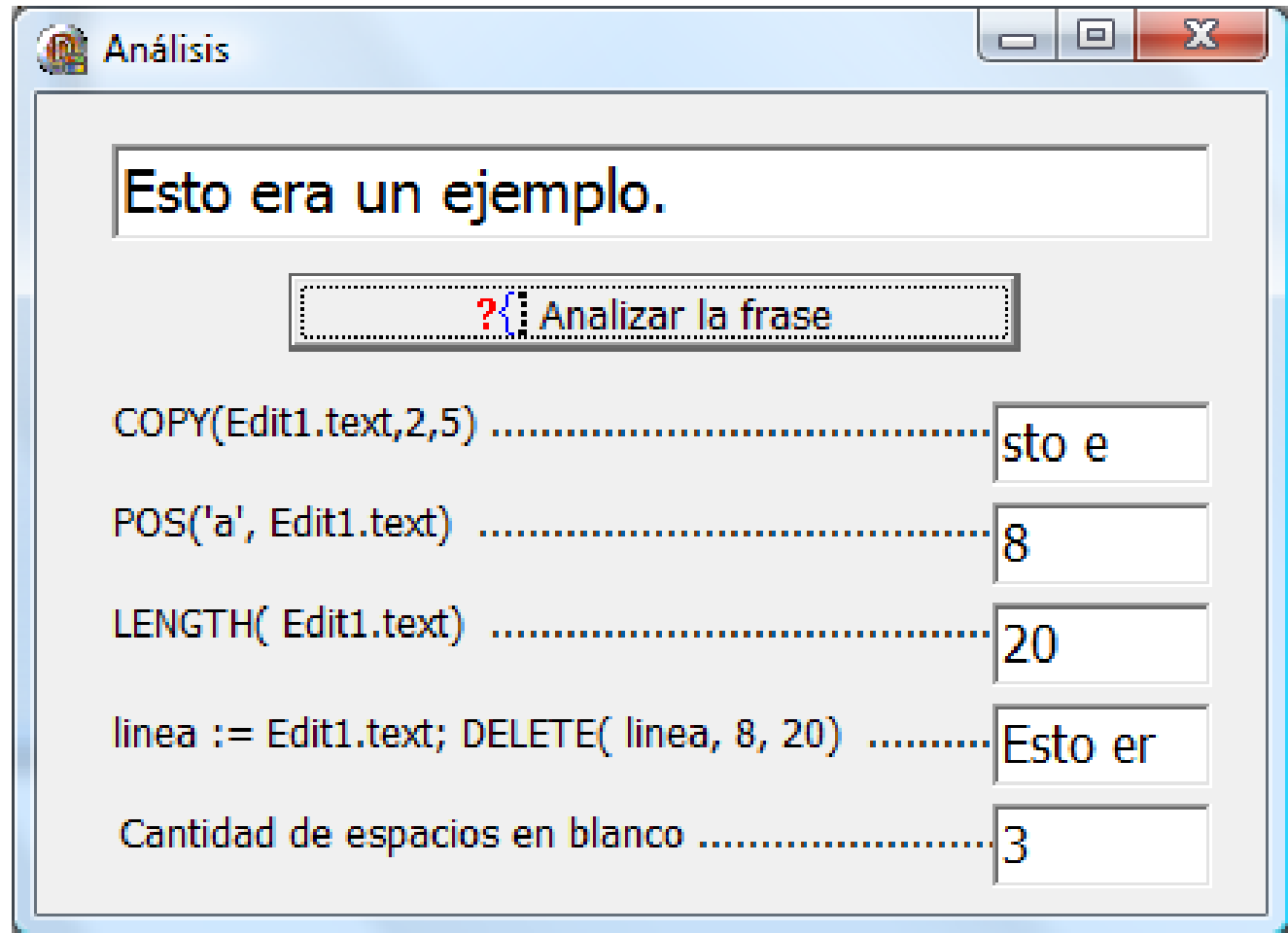
- ✳ Permite borrar caracteres de un string dado.

```
Var linea : string;  
    ubicacion : integer;  
begin  
    linea := 'Esto es un ejemplo.';  
    delete( linea, 2, 11);
```

El valor de **linea** será 'Ejemplo.'

Ejemplo

Analisis.dpr



The screenshot shows a Delphi application window titled "Análisis". Inside the window, there is a text box containing the sentence "Esto era un ejemplo.". Below the text box is a button with a red question mark icon and the text "Analizar la frase". Below the button, there are five rows of text, each followed by a dotted line and a text box containing the result of a string operation:

COPY(Edit1.text,2,5)	sto e
POS('a', Edit1.text)	8
LENGTH(Edit1.text)	20
linea := Edit1.text; DELETE(linea, 8, 20)	Esto er
Cantidad de espacios en blanco	3

Unidades en Pascal

- ✦ Una unidad consiste de declaraciones de tipos, constantes, variables y procesos (funciones y procedimientos). Cada unidad se define en un archivo con extensión **.pas**

- ✦ **Sintaxis**

Unit nombre_de_la_unidad;

interface

{ declaraciones públicas }

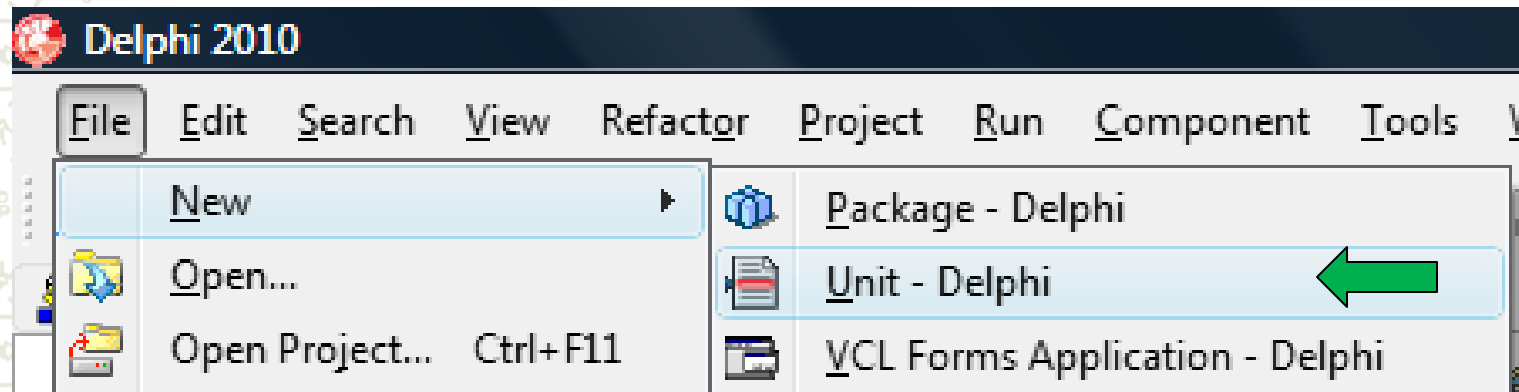
implementation

{ área privada. Lo aquí declarado sólo es conocido dentro de la unidad }

end

Unidades en Pascal

- ✶ Cierre la aplicación anterior (**File \ Close All**).
- ✶ Abra únicamente una unidad nueva



Unit Unit1;
interface

Declaración pública

Function CantVocales(S : String) : integer;

implementation

Function CantVocales(S : String) : integer;

var auxi, : integer;

i : integer;

begin

auxi := 0;

for i:= 1 to length(S) do

if S[i] in ['a', 'e', 'i', 'o', 'u'] then

auxi := auxi + 1;

CantVocales := auxi;

end;

end.

Lo aquí declarado es
privado. Sólo lo
conoce esta unidad

Unit Unit1;

interface

Function CantVocales(S : String) : integer;

implementation

Function CantVocales(S : String) : integer;

var auxi, : integer;

i : integer;

begin

auxi := 0;

for i:= 1 to length(S) do

if **S[i]** in ['a', 'e', 'i', 'o', 'u'] then
auxi := auxi + 1;

CantVocales := auxi;

end;

end.

Acceso a cada letra
del string. El 1er.
elem. tiene índice 1

Unidades en Pascal

- ✳ Para cambiar el nombre de la unidad debe salvarla con otro nombre mediante la opción

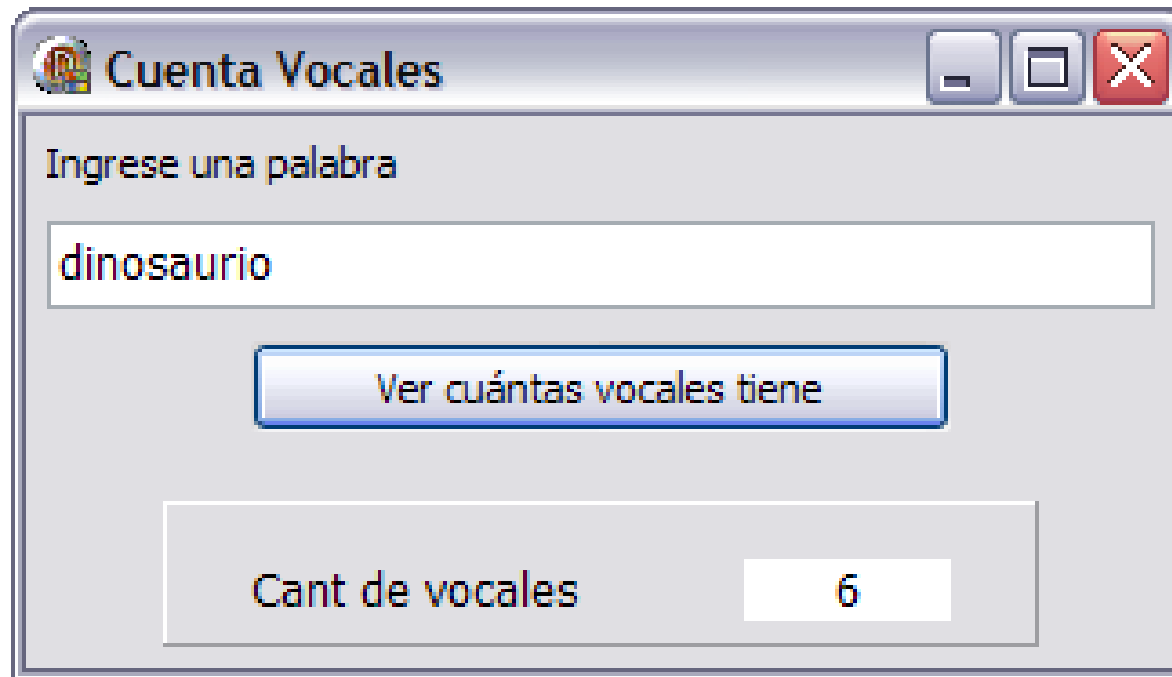
File \ Save as ...



El nombre del archivo debe coincidir con el nombre de la unidad

Ejemplo Unidades.dpr

✦ Utilice la unidad anterior en la siguiente aplicación



The screenshot shows a Windows application window titled "Cuenta Vocales". Inside the window, there is a text input field containing the word "dinosaurio". Below the input field is a button labeled "Ver cuántas vocales tiene". At the bottom of the window, there is a label "Cant de vocales" followed by a text box displaying the number "6".

La Unidad del Formulario

```
unit Unit1;
```

```
interface
```

```
uses Windows, Messages, SysUtils, Classes, Graphics,  
    Controls, Forms, Dialogs;
```

```
type
```

```
    TForm1 = class(TForm)  
    private  
        { Private declarations }  
    public  
        { Public declarations }  
    end;
```

```
var
```

```
    Form1: TForm1;
```

```
implementation
```

```
{ $R *.DFM }
```

```
end.
```

Definición de la clase
a la que pertenece el
formulario

type

TForm1 = **class**(TForm)

Label1: TLabel;

Edit1: TEdit;

Button1: TButton;

Panel1: TPanel;

Label2: TLabel;

Label3: TLabel;

private

{ Private declarations }

public

{ Public declarations }

end;

Para Delphi

Esta es la zona donde Delphi agrega código automáticamente

Para el Programador

En estas secciones podemos declarar lo que vamos a utilizar en el programa.

Así se ve la clase **TForm1**, a la que pertenece el formulario, luego de pegar todas las componentes.

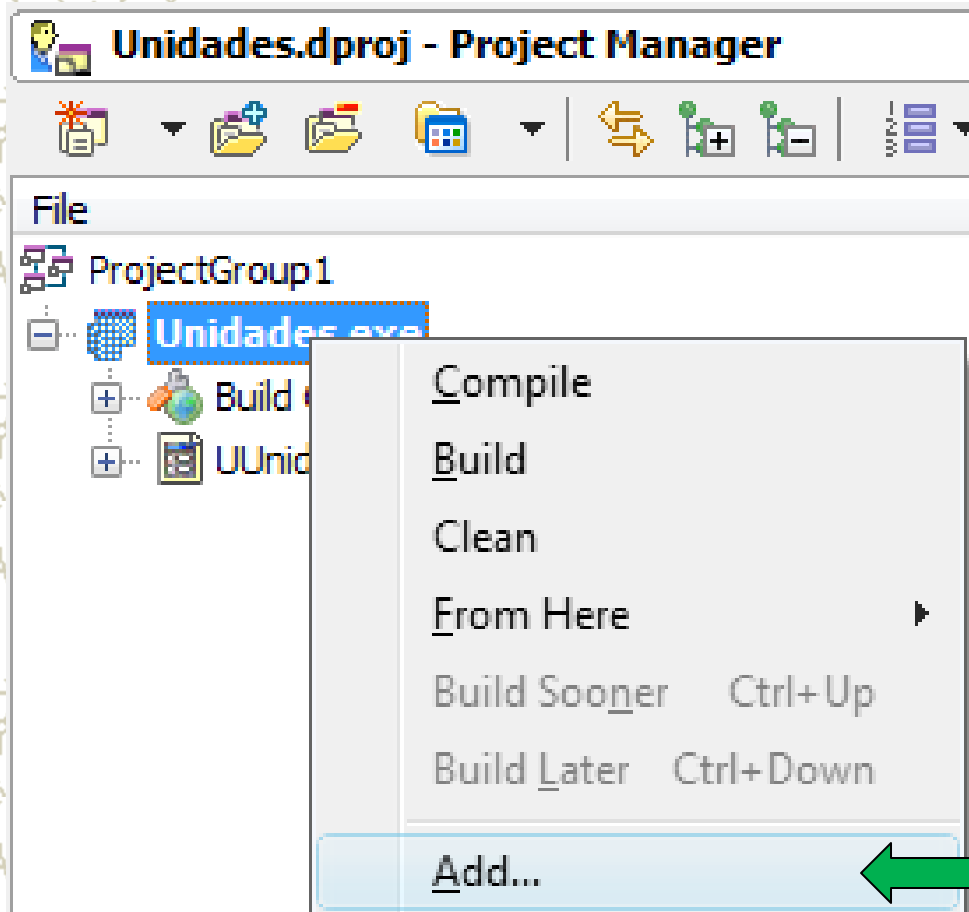
Invocación de la función

- ✦ En el OnClick del botón debe realizarse el llamado a la función

```
procedure TForm2.Button1Click(Sender: TObject);  
begin  
    label3.caption := IntToStr( CantVocales(edit1.text) );  
    panel1.visible := true;  
end;
```

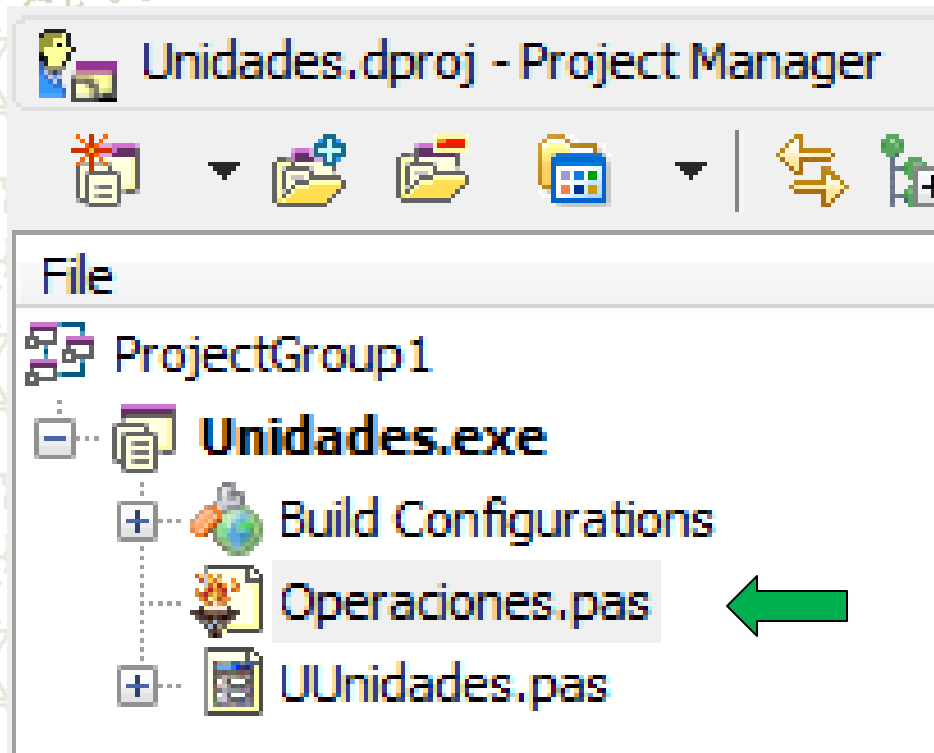
Falta indicar dónde está definida
la función **CantVocales**

Agregando la unidad al proyecto



⚡ Seleccione la
unidad
operaciones.pas

Agregando la unidad al proyecto



La unidad **operaciones.pas** ha sido agregada.

Cláusula Uses

- ✦ Esta cláusula está ubicada dentro de sección de interface antes de la declaración de la clase del formulario.
- ✦ Allí debe agregarse el nombre de la unidad (archivo.pas) que contiene a la función a utilizar.
 - Pude usarse **File \ Use Unit ...**

Verifique si su aplicación funciona

Ejercicio adicional

- ✳ Defina una unidad que permita trabajar sobre un vector de 100 números enteros declarado de la siguiente forma

```
CONST N = 100;  
type TVector = record  
    nro : array[1..N] of integer;  
    long : integer;  
end;
```

Ejercicio adicional

- Las operaciones a realizar sobre el vector son las siguientes

```
function Promedio(V : TVector) : real;
```

```
Procedure Rango(V : TVector;
```

```
    var min,max:integer);
```

```
Procedure BorrarTodo( var V : TVector );
```

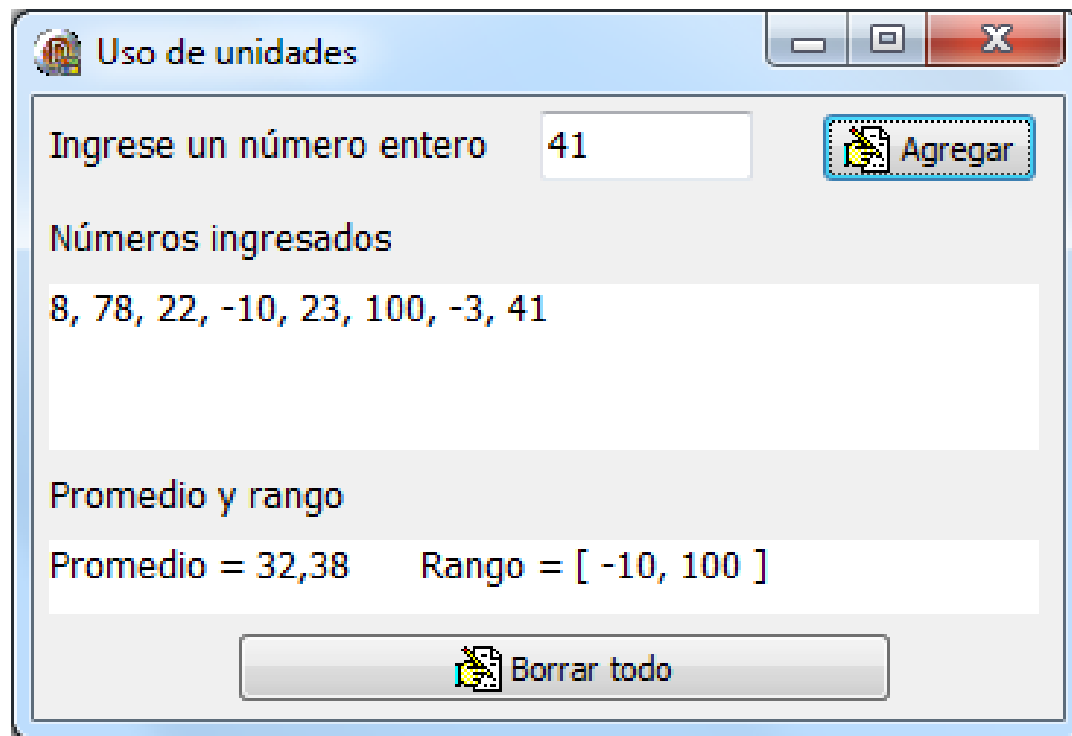
```
Procedure Agregar(var V : TVector;
```

```
    n : integer);
```

Ejercicio adicional

- Utilice la unidad anterior desde una aplicación con la siguiente interfaz

EjercUnidades.exe



The screenshot shows a Windows application window titled "Uso de unidades". It features a text input field with the value "41" and an "Agregar" button. Below this, a list of numbers is displayed: "8, 78, 22, -10, 23, 100, -3, 41". At the bottom, the application shows the calculated "Promedio y rango" as "Promedio = 32,38" and "Rango = [-10, 100]". A "Borrar todo" button is located at the bottom of the window.

Uso de unidades

Ingrese un número entero 41 Agregar

Números ingresados

8, 78, 22, -10, 23, 100, -3, 41

Promedio y rango

Promedio = 32,38 Rango = [-10, 100]

Borrar todo

Resumen

✶ Excepciones

- Sintaxis

✶ Funciones de conversión

- De número a String
 - **IntToStr, FloatToStr**
- De String a número
 - **StrToInt, StrToFloat**

✶ Manejo de Strings

- Como vector de caracteres.
- Funciones : **Length, Pos y Copy.**
- Procedimiento **Delete**

✶ Unidades

- Sintaxis
- Clausula **Uses**