

Estructura: program · unit · library · package

```
program MiApp; // fichero .dpr (proyecto)
{$APPTYPE CONSOLE}
uses
  System.SysUtils,
  UMiUnidad in 'UMiUnidad.pas';
begin
  try
    Writeln('Hola, Delphi');
  except
    on E: Exception do
      Writeln(E.ClassName, ': ', E.Message);
    end;
  end.

unit UMiUnidad; // fichero .pas (una unidad = un módulo)
interface // sección PÚBLICA (visible al usar la unidad)
uses System.Classes;
const MaxItems = 100;
type TModo = (mLectura, mEscritura);
TCosa = class ... end; // declaración (sin cuerpos)
var Contador: Integer;
procedure Hacer(const S: string);
implementation // sección PRIVADA
uses System.SysUtils; // aquí: evita referencias circulares
{$R *.dfm} // recurso del formulario (VCL) / *.fmx (FMX)
procedure Hacer(const S: string);
begin
  ...
end;
initialization // al cargar la unidad (opcional)
finalization // al descargar (solo si hay initialization)
end.

library MiDll; // DLL
function Suma(A, B: Integer): Integer; stdcall;
begin Result := A + B; end;
exports Suma name 'Suma';
begin end.
package MiPaquete; requires rtl, vcl; contains UMiUnidad; end. // .dpk
```

Palabras reservadas (64 + at, on)

No se pueden redefinir ni usar como identificador. at (raise ... at) y on (except on ...) se tratan como reservadas. private protected public published automated son reservadas dentro de una declaración de clase. Para usar una como identificador: &type, &begin.

and array as asm begin case class const constructor destructor dispinterface div

do downto else end except exports file finalization finally for function goto if

implementation in inherited initialization inline interface is label library mod

nil not object of or packed procedure program property raise record repeat

resourcestring set shl shr string then threadvar to try type unit until uses var

while with xor

Directivas (contextuales)

Solo tienen significado en ciertas posiciones (fin de una declaración, dentro de una clase, etc.); fuera de ahí son identificadores normales.

absolute abstract assembler automated cdecl contains default delayed deprecated

dispid dynamic experimental export external far final forward helper implements

index inline library local message name near nodefault noreturn operator out

overload override package pascal platform private protected public published read

readonly reference register reintroduce requires resident safecall sealed static

stdcall stored strict unsafe varargs virtual winapi write writeonly

- Aviso (hinting):** deprecated 'usa X', experimental, platform, library → generan warning al usarse.
- Convención de llamada:** register (defecto), stdcall, cdecl, safecall, pascal, winapi (= stdcall en Win32).
- Obsoletas:** far, near, resident, local, assembler, export. **Nuevas:** noreturn (13).
- Atributos del compilador:** [weak], [unsafe], [volatile], [ref], [Result], [HPPGEN(...)].

Operadores y precedencia

Nivel	Operadores	Tipo
1 (alta)	@ not ^(deref) . [] ()	unarios / primarios
2	* / div mod and shl shr as	multiplicativos
3	+ - or xor	aditivos
4 (baja)	= <> < > <= >= in is	relacionales

- Asignación :=. No existen +=, ++: usar Inc(I), Dec(I, 2), I := I + 1.
- / siempre devuelve real; div y mod solo enteros. and/or cortocircuitan por defecto ({\$B-}).
- Bits:** and or xor not shl shr sobre enteros. **Cadenas:** + concatena; comparación = < ... por código.
- Conjuntos:** + unión, - diferencia, * intersección, in pertenencia, <= subconjunto, >= superconjunto.
- Clases:** Obj is TClase (comprueba), Obj as TClase (cast seguro, lanza EInvalidCast), TClase(Obj) (cast duro).
- Punteros:** @X dirección, P^ desreferencia. =/<> comparan punteros, objetos e interfaces por referencia.
- Delphi 13:** Obj is not TClase, X not in [1..5], expresión condicional if C then A else B.

Literales

```
42      -7      $FF (hex)      %1011 (binario, 11+)      1_000_000 (separador, 11+)
3.14    1.5E-5    2.0          // 2 es entero; 2.0 real
'A'     #65      #$41 ^A (Ctrl-A) 'It's' 'línea 1'#13#10'línea 2'
''      (cadena vacía) nil      True / False slineBreak
S := '''          // string multilínea (12+): triple comilla + salto de línea
      SELECT * FROM Clientes
      WHERE Id = :Id
      ''';
[1, 2, 5..9] (* conjunto *) [1, 2, 3] (* constructor de array dinámico *)
```

Los literales de cadena pueden superar 255 caracteres desde Delphi 12. #13#10 = CRLF; #9 = tab; #0 = nulo.

Comentarios y directivas de compilador

```
// comentario de línea      { de bloque }      (* también de bloque *)
{DEFINE LOGS} {UNDEF LOGS}
{IFDEF MSWINDOWS} ... {ELSE} ... {ENDIF}      {IFDEF X} ... {ENDIF}
{$IF Defined(ANDROID) or Defined(IOS)} ... {$ELSEIF ...} ... {$ENDIF}
{$IF CompilerVersion >= 37} ... {$IFEND}      // 37 = Delphi 13, 36 = 12, 35 = 11
{$R *.dfm} {$R iconos.res} {$I defs.inc}      // recursos / include
{$RANGECHECKS ON} {$R+} {$OVERFLOWCHECKS ON} {$Q+} // comprobaciones (debug)
{$WARN SYMBOL_DEPRECATED OFF} {$WARNINGS OFF} {$HINTS OFF}
{$SCOPEDENUMS ON} {$POINTERMATH ON} {$ZEROBASEDSTRINGS OFF} {$BOOLEVAL OFF}
{$M+} / {$TypeInfo ON} (RTTI published) {$RTTI EXPLICIT METHODS([vcPublic])}
{$REGION 'Nombre'} ... {$ENDREGION}      {$ALIGN 8} {$MINENUMSIZE 4}
{$TEXTBLOCK native | CRLF | LF}      // saltos de línea en strings multilinea (12)
{$PUSHOPT} ... {$POPOPT}      // guarda/restaura opciones (13)
{$MESSAGE 'Info'} {$MESSAGE WARN 'Ojo'} {$MESSAGE ERROR 'No'} {$MESSAGE FATAL 'x'}
```

Símbolos predefinidos: MSWINDOWS WIN32 WIN64 LINUX LINUX64 MACOS OSX64 IOS ANDROID POSIX CPUX86 CPUX64 CPUARM CPUARM64 CPU32BITS CPU64BITS CONSOLE DEBUG RELEASE LLVM VER370 (13) VER360 (12) VER350 (11) VER340 (10.4) VER330 (10.3). Constantes: CompilerVersion (37.0 = 13), RTLVersion.

Identificadores y convenciones

- **No** distingue mayúsculas/minúsculas (Nombre = nombre). Empiezan por letra o `_`; 255 caracteres significativos.
- Prefijos habituales: TClase, IInterfaz, EExcepción, PUntero, FCampo privado, AParámetro, LLocal, GGlobal, OnEvento, TxxEvent.
- Unidades con espacio de nombres: System., System.Classes, Vcl.Forms, FMX.Forms, Winapi.Windows, Data.DB, FireDAC.*. Calificar: Vcl.Graphics.TBitmap.
- Identificadores implícitos: Self (instancia o clase actual), Result (retorno de función), inherited.
- Ámbito: bloque (begin...end con var inline), rutina, clase, unidad. Lo declarado en implementation es privado de la unidad.
- Nombres de unidad = nombre de fichero (.pas). Formulario: .pas + .dfm (VCL) / .fmx (FMX). Proyecto: .dpr + .dproj.

Enteros

Tipo	Bytes	Rango	Notas
ShortInt	1	-128 .. 127	Int8
SmallInt	2	-32 768 .. 32 767	Int16
Integer	4	-2 147 483 648 .. 2 147 483 647	Int32; resultado por defecto de la aritmética
Int64	8	-9,2E18 .. 9,2E18	operaciones en 64 bits solo si un operando es Int64
Byte	1	0 .. 255	UInt8
Word	2	0 .. 65 535	UInt16
Cardinal	4	0 .. 4 294 967 295	UInt32
UInt64	8	0 .. 1,8E19	
NativeInt / NativeUInt	4 / 8	según plataforma	= Integer en 32 bits, Int64 en 64 bits (tamaño de puntero)
LongInt / LongWord	4 / 8		4 bytes en Windows (32 y 64) y en 32 bits; 8 en POSIX 64 (iOS, macOS, Android, Linux)
FixedInt / FixedUInt	4		siempre 32 bits en todas las plataformas

Desbordamiento silencioso salvo `{Q+}`; rango fuera de límites salvo `{R+}`. `High(Integer) + 1` → se desborda; usar `Int64(I) + 1`.

Reales

Tipo	Bytes	Rango / precisión	Notas
Single	4	1,5E-45 .. 3,4E38 · 7-8 dígitos	IEEE 754 simple
Double	8	5,0E-324 .. 1,7E308 · 15-16 dígitos	preferido ; Real = Double
Extended	10 / 8 / 16	19-20 dígitos (80 bits) en Win32	10 bytes Win32; en Win64 = Double (8) ; 16 en Linux/macOS Intel. No portable
Currency	8	±922 337 203 685 477,5807	punto fijo, 4 decimales exactos → dinero
Comp	8	±9,2E18	entero de 64 bits tratado como real (legado)
Real48	6		Turbo Pascal; no disponible en Win64

Comparar reales con `SameValue(A, B, Epsilon)` (System.Math), no con `=`. `Round` redondea al par (bancario); `Trunc` trunca.

Booleanos y caracteres

Tipo	Bytes	Notas
Boolean	1	preferido ; solo True/False
ByteBool WordBool LongBool	1 / 2 / 4	compatibilidad C / WinAPI: cualquier valor ≠ 0 es True
Char = WideChar	2	UTF-16 (code unit). <code>Ord('A')</code> = 65, <code>Chr(65)</code> , <code>'A' < 'B'</code>
AnsiChar	1	8 bits según página de códigos
UCS4Char	4	punto de código Unicode completo

`Char` es ordinal: `Succ('A')`, for `C := 'a' to 'z', C in ['0'..'9']`, `CharInSet(C, ['a'..'z'])`, `C.IsDigit` (helper), `UpCase(C)`.

Cadenas

Tipo	Notas
string = UnicodeString	por defecto. UTF-16, dinámica, contada por referencias, copy-on-write, índice 1-based (S[1]), terminada en #0
AnsiString	8 bits con página de códigos (type MiStr = type AnsiString(1252)). Escritorio
UTF8String	= AnsiString(65001); todas las plataformas
RawByteString	AnsiString sin conversión de página de códigos; solo para parámetros const
WideString	BSTR de COM, sin refcount, lenta. Solo para COM/escritorio
ShortString	string[20]: 1.255 AnsiChar, 256 bytes estáticos, legado
PChar / PAnsiChar / PWideChar	punteros a cadenas terminadas en #0 (API). PChar(S) nunca es nil

```
Length(S)  SetLength(S, 10)  S[1]  Copy(S, 2, 3)  Pos('x', S)  Delete(S, 1, 2)  Insert('ab', S, 1)
S := S + 'x';  S := Format('%s=%d (%.2f) %x %-8s|', ['a', 1, 2.5, 255, 'izq']);
S.Length  S.ToUpper  S.Trim  S.Contains('x')  S.StartsWith('a')  S.Split(['.', ','])
S.IndexOf('x')  S.Substring(0, 3)  S.Replace('a', 'b')  S.IsEmpty  string.Join(',', Arr)
// ¡los helpers (TStringHelper) son 0-based: S.Chars[0]; las funciones globales 1-based!
SameText(A, B)  CompareText  AnsiCompareStr  StringReplace(S, 'a', 'b', [rfReplaceAll])
TEncoding.UTF8.GetBytes(S)  TEncoding.UTF8.GetString(Bytes)  // string <-> TBytes
SB := TStringBuilder.Create; SB.Append('a').AppendLine('b'); S := SB.ToString;
```

Ordinales: enumerados, subrangos, conjuntos

```
type
  TColor = (cRojo, cVerde, cAzul);           // Ord(cRojo) = 0; tamaño 1 byte
  TNivel = (nBajo = 1, nMedio = 5, nAlto = 10); // ordinales explícitos
  {$SCOPEDENUMS ON}
  TEstado = (Activo, Inactivo);              // se usa como TEstado.Activo
  {$SCOPEDENUMS OFF}
  TDigito = 0..9;  TMayus = 'A'..'Z';        // subrangos
  TColores = set of TColor;                  // conjunto (máx. 256 elementos)
  TFlags = set of (fLectura, fEscritura);
var C: TColores; Col: TColor;
begin
  C := [cRojo, cAzul]; Include(C, cVerde); Exclude(C, cRojo);
  C := C + [cAzul] - [cRojo];  if cVerde in C then ...;  if C = [] then ...;
  Inc(Col); Dec(Col, 2); Succ(cRojo); Pred(cAzul); Low(TColor); High(TColor);
  Ord(cAzul) = 2;  TColor(1) = cVerde;        // conversión ordinal <-> enum
  for Col := Low(TColor) to High(TColor) do ...;  for Col in C do ...;
  GetEnumName(TypeInfo(TColor), Ord(Col));    // System.TypeInfo: 'cRojo'
  TRttiEnumerationType.GetName<TColor>(Col);  // System.Rtti
end;
```

Arrays

```
var
  A: array[0..9] of Integer;           // estático: índices fijos, en pila
  M: array[1..3, 1..3] of Double;      // multidimensional
  N: array[Byte] of Char;              // indexado por cualquier ordinal
  D: array of string;                  // dinámico: 0-based, en heap, por referencia
  G: TArray<Integer>;                  // = array of Integer (preferido)
  J: array of array of Integer;        // dinámico anidado (jagged)
begin
  SetLength(D, 5);  D[0] := 'a';  SetLength(D, Length(D) + 1);
  SetLength(J, 3, 4);
  G := [1, 2, 3];  G := G + [4, 5];    // constructor y concatenación
  Insert([9], G, 0);  Delete(G, 0, 1); // inserción / borrado
  for I := Low(A) to High(A) do ...;  for X in G do ...;
  B := Copy(G);                        // copia real; B := G comparte la memoria
  G := nil;                            // libera (refcount)
  TArray.Sort<Integer>(G);  TArray.Contains<Integer>(G, 3);  TArray.BinarySearch<Integer>(G, 3, I)
end;
procedure P(const Arr: array of Integer); // open array: acepta cualquier tamaño
procedure Q(const Args: array of const);  // argumentos variados: Q(['a', 1, 2.5])
```

Records

```
type
  TPunto = record
    X, Y: Double;           // los campos van antes que métodos
    constructor Create(AX, AY: Double); // ≥ 1 parámetro obligatorio
    function Dist(const P: TPunto): Double;
    class operator Add(const A, B: TPunto): TPunto; // sobrecarga de operadores
    class operator Implicit(V: Double): TPunto;     // conversión implícita
    class operator Equal(const A, B: TPunto): Boolean;
    class function Origen: TPunto; static;
    property Modulo: Double read GetModulo;
  private
    function GetModulo: Double;
  end;
  TDato = packed record           // packed: sin relleno de alineación
    case Tipo: Byte of           // parte variante (unión)
      0: (I: Integer);
      1: (D: Double);
      2: (B: array[0..7] of Byte);
  end;
  PPunto = ^TPunto;
var P: TPunto;
begin
  P := TPunto.Create(1, 2);  P.X := 3; // tipo VALOR: se copia al asignar/pasar
  P := Default(TPunto);      // todos los campos a cero
  FillChar(P, SizeOf(P), 0); // solo si no tiene campos gestionados
end;
```

Records útiles: TPoint, TRect, TPointF, TRectF, TSize, TGUID, TDateTime (Double), TTimeStamp, TFormatSettings, TValue (RTTI), TVarRec.

Conversión y funciones de uso diario

Función	Uso
IntToStr StrToInt StrToIntDef(S, 0) TryStrToInt(S, I)	entero ↔ string; StrToInt('\$FF') hex
FloatToStr StrToFloat TryStrToFloat FormatFloat('0.00', X) FloatToStrF	dependen de FormatSettings (coma/punto); usar TFormatSettings.Invariant para ficheros
Format('%d %s %.2f %x %8.3f %-10s %%', [...])	formateo tipo printf; %*d ancho dinámico
Round Trunc Ceil Floor Frac Int Abs Sqr Sqrt Power Max Min EnsureRange InRange	System.Math para Ceil/Floor/Power/Max/Min
IntToHex(255, 2) → 'FF' · Random(10) RandomRange Randomize	
UpperCase LowerCase Trim TrimLeft TrimRight QuotedStr StringOfChar('-', 20)	LeftStr RightStr MidStr ContainsText SplitString (System.StrUtils)
Inc Dec Low High Length SetLength SizeOf TypeInfo Default(T) Assigned Supports	intrínsecos del compilador
TPath.Combine TFile.ReadAllText TDirectory.GetFiles	System.IOUtils
TRegEx.IsMatch · TJSONObject.ParseJSONValue · TNetHTTPClient	System.RegularExpressions · System.JSON · System.Net.HttpClient

Punteros y tipos procedimentales

```
type
  PInt = ^Integer;           // puntero tipado
  TOp = function(A, B: Integer): Integer; // tipo procedimental
  TNotifyEvent = procedure(Sender: TObject) of object; // método (evento): puntero + Self
  TFn = reference to function(X: Integer): Integer; // método anónimo (closure)
var P: PInt; Op: TOp; V: Integer; Buf: Pointer;
begin
  P := @V; P^ := 10; Inc(P); // {$POINTERMATH ON} permite P + 1, P[2]
  if Assigned(P) then ...; P := nil; // Assigned(P) = (P <> nil)
  Op := Suma; V := Op(1, 2); // @Suma si hay ambigüedad; Assigned(Op)
  Pointer PByte PInteger PChar PAnsiChar PWideChar PPointer PDouble
  NativeInt(P) + 4 Pointer(N) // aritmética genérica mediante cast
  GetMem(Buf, 1024); try ... finally FreeMem(Buf); end;
end;
```

Alias, tipos distintos y casts

```
type
  TAlias = Integer;           // alias: intercambiable con Integer
  TMetros = type Double;     // tipo DISTINTO: exige cast desde Double; RTTI propia
  PBuf = ^TBuf; TBuf = array[0..255] of Byte; // puntero a tipo declarado después
begin
  M := TMetros(D); D := Double(M); // cast de valor: mismo tamaño en memoria
  I := Integer(Ch); Ch := Char(65); // ordinal <-> ordinal; B := Byte(I) trunca
  P := Pointer(Obj); Obj := TObject(P); // referencias: cast duro, sin comprobación
  if Obj is TForm then F := TForm(Obj); // comprobado: F := Obj as TForm
  S := string(PC); PC := PChar(S); // string <-> puntero (no liberar PC)
end;
```

Variant, fechas y otros tipos

```
var V: Variant; // 16 bytes; contenedor dinámico (COM/DB)
V := 10; V := 'texto'; V := Null; V := Unassigned;
if VarIsNull(V) or VarIsEmpty(V) then ...; VarType(V) = varInteger; VarToStr(V)
OleVariant (automatización COM) TVarRec (array of const) TValue (RTTI)
TDateTime = Double: entero = días desde 30/12/1899; fracción = hora
Now Date Time Tomorrow Yesterday EncodeDate(2026, 10, 24) EncodeTime(10, 30, 0, 0)
FormatDateTime('dd/mm/yyyy hh:nn:ss', Now) DateToStr StrToDate DateTimeToStr
DecodeDate(Dt, Y, M, D) IncDay(Dt) IncMonth DaysBetween YearOf DayOfTheWeek (DateUtils)
TDate / TTime (alias con RTTI) TTimeZone.Local TStopwatch.StartNew (System.Diagnostics)
TGUID: TGUID.NewGuid GUIDToString StringToGUID CreateGUID(G)
TBytes = TArray<Byte> TProc TFunc<T> TPredicate<T> TComparison<T> (System.SysUtils)
```

Variables y constantes

```
const
  MaxLen = 255; // constante verdadera (sin tipo, se sustituye)
  Tasa: Double = 0.21; // constante con tipo (solo modificable con {$J+})
  Dias: array[0..1] of string = ('Lun', 'Mar');
resourcestring
  SError = 'Error %d'; // localizable (recurso de cadena)
var
  Total: Integer = 0; // global con valor inicial (globales y campos → 0/nil/'')
  Buf: array[0..3] of Byte;
  Alias: Integer absolute Buf; // comparte la misma dirección
threadvar
  Ctx: Integer; // una copia por hilo (TLS)
procedure Demo;
var L: Integer; // locales NO inicializadas (salvo tipos gestionados)
begin
  var X := 10; // variable inline (10.3+), tipo inferido Integer
  var S: string := 'a'; // con tipo explícito
  const K = X * 2; // constante inline
  for var I := 0 to 9 do ...; // índice inline
  for var Item in Lista do ...; // ámbito: hasta el end del bloque donde se declara
  if X > 5 then begin var Y := X; ... end;
end;
```

Control de flujo

```
if A > B then Max := A else Max := B;           // sin ';' antes de else
if Cond then begin ... end
else if Otra then ... else ...;
case Dia of                                     // solo ordinales (no strings)
  1, 7: S := 'Fin de semana';
  2..6: S := 'Laborable';
else      S := '?';
end;
for I := 1 to 10 do ...;   for I := 10 downto 1 do ...; // I no se modifica dentro
for Ch in 'Hola' do ...;   for X in Arr do ...;   for Obj in Lista do ...; // for-in
while Cond do begin ... end;
repeat ... until Cond;           // al menos una iteración
Break; Continue; Exit; Exit(Valor); // Exit(Valor) asigna Result y sale
with Obj do begin Prop := 1; end; // evitar: ambigüedad de nombres
Max := if A > B then A else B;    // expresión condicional (13+)
ShowMessage(if N = 1 then 'uno' else 'varios'); // evalúa solo una rama
if Obj is not TButton then ...;   if X not in [1..5] then ...; // 13+
```

Procedimientos y funciones

```
procedure Saluda(const Nombre: string; Veces: Integer = 1); // parámetro por defecto
function Suma(A, B: Integer): Integer; overload;
function Suma(A, B: Double): Double; overload;
function Suma(const A: array of Integer): Integer; overload; // open array
procedure Swap(var A, B: Integer); // por referencia (entrada/salida)
procedure Lee(out Valor: Integer); // solo salida (el compilador lo inicializa)
procedure Mira(const [ref] R: TRecGrande); // const por referencia garantizada
procedure Log(const Args: array of const); // Log(['a', 1, 2.5]) → TVarRec
function Calc(X: Integer): Integer; inline; // sugerencia de inlining
function MessageBeep(uType: Cardinal): LongBool; stdcall; external 'user32.dll';
procedure Adelante(X: Integer); forward; // declaración anticipada
procedure Fallo(const Msg: string); noreturn; // nunca retorna (13+)
```

```
function Exterior(N: Integer): Integer;
  function Interior(X: Integer): Integer; // rutina anidada: ve las locales
  begin Result := X * N; end;
begin
  Result := Interior(2); // Result = valor de retorno
  if N < 0 then Exit(0);
end;
```

Parámetros: **valor** (copia), **const** (solo lectura, sin copia → usar para string/record/interface), **var** (referencia), **out** (salida). Convenciones: register (defecto, 3 registros), stdcall/cdecl (API, DLL), safecall (COM).

Excepciones

```
try
  N := StrToInt(Edit1.Text);
except
  on E: EConvertError do ShowMessage('No es número: ' + E.Message);
  on E: EMiError do begin Log(E); raise; end; // raise sin argumento: relanza
  on Exception do ...; // sin variable
  else ShowMessage('Otro error'); // cualquier otra
end;
try
  F := TFileStream.Create(Ruta, fmOpenRead);
  try ... finally F.Free; end; // finally SIEMPRE se ejecuta
except
  on E: EFOpenError do ...;
end;
raise EMiError.Create('Mensaje');
raise EMiError.CreateFmt('Código %d', [Cod]);
raise Exception.Create('x') at ReturnAddress; // dirección de error personalizada
Exception.RaiseOuterException(ENueva.Create('ctx')); // encadena: E.InnerException
Abort; // EAbort: silenciosa (sin diálogo)
type EMiError = class(Exception); // derivar siempre de Exception
```

Comunes (System.SysUtils): EAbort EAccessViolation EInvalidPointer EOutOfMemory EConvertError EDivByZero ERangeError EIntOverflow EInvalidOp EZeroDivide EOverflow EInOutError EInvalidCast EArgumentException EArgumentOutOfRangeException EArgumentNilException ENotSupportedException ENotImplemented EOSError EListError EStreamError EFOpenError EFCREATEError EAssertionFailed; EDatabaseError (Data.DB). Props: Message ClassName StackTrace InnerException.

Métodos anónimos, helpers, operadores, atributos

```
var Suma := 0;
var Acum: TProc<Integer> := procedure(X: Integer) begin Inc(Suma, X); end; // captura Suma
Acum(5); // por REFERENCIA (vive mientras el closure)
TTask.Run(procedure begin ... end); // TProc, TFunc<T>, TPredicate<T>
type
  TStrHelper = record helper for string // helper de tipo (uno activo por tipo;
    function Invertida: string; // oculta al TStringHelper del RTL)
  end;
  TFormHelper = class helper for TForm // añade métodos sin heredar (sin campos)
    procedure Centrar;
  end;
  TMoney = record
    class operator Add(const A, B: TMoney): TMoney; // + - * / div mod
    class operator Implicit(V: Currency): TMoney; // Explicit, Equal, NotEqual,
    class operator LessThan(const A, B: TMoney): Boolean; // GreaterThan, Negative, Inc...
  end;
  TagAttribute = class(TCustomAttribute) // atributo personalizado
    constructor Create(const AValor: string);
  end;
  [Tag('demo')] TMiClase = class ... end; // sufijo 'Attribute' opcional al usarlo
// Leer: TRttiContext.Create.GetType(TMiClase).GetAttributes (System.Rtti)
Caption := NameOf(Button1); // 'Button1' (13+)
```

Clases

```
type
TAnimal = class(TObject)           // (TObject) opcional
private                             // visible en TODA la unidad
    FNombre: string;
    procedure SetNombre(const Value: string);
strict private                      // solo esta clase
    FId: Integer;
protected                          // + descendientes (y la unidad)
    function GetSonido: string; virtual; abstract;
public
    constructor Create(const ANombre: string);
    destructor Destroy; override;    // SIEMPRE override
    procedure Hablar; virtual;       // redefinible (dynamic: tabla pequeña, más lento)
    class function Especie: string; virtual; // método de clase (Self = la clase)
    class var Contador: Integer;      // campo de clase (compartido)
    class constructor Create;         // una vez por tipo, antes de usarse
    property Nombre: string read FNombre write SetNombre;
published                          // public + RTTI (Object Inspector)
end;
TPerro = class sealed(TAnimal)      // sealed: no heredable; class abstract
protected
    function GetSonido: string; override;
public
    procedure Hablar; override; final; // final: no se puede volver a redefinir
    procedure Saltar; reintroduce;    // oculta un método heredado sin override
    procedure WMSize(var Msg: TWMSize); message WM_SIZE; // handler de mensaje
end;
TAnimalClass = class of TAnimal;    // metaclass (referencia de clase)

procedure TPerro.Hablar;
begin
    inherited;                       // llama a TAnimal.Hablar (mismos args)
    inherited Hablar;                // forma explícita
end;
var A: TAnimal; C: TAnimalClass;
    C := TPerro; A := C.Create('Rex'); // polimórfico si el constructor es virtual
    if A is TPerro then TPerro(A).Saltar; // is + cast duro
    (A as TPerro).Saltar;             // cast comprobado (EInvalidCast)
    A.ClassName A.ClassType A.ClassParent A.InheritsFrom(TAnimal) A.InstanceSize
```

Propiedades

```
property Nombre: string read FNombre write SetNombre; // getter/setter (campo o método)
property Edad: Integer read FEdad write FEdad default 0; // default: solo afecta al .dfm
property Count: Integer read GetCount; // solo lectura
property Items[I: Integer]: string read GetItem write SetItem; default; // array: Obj[0]
property Color: TColor read FColor write FColor stored IsColorStored; // condición de guardado
property Texto: string index 1 read GetStr write SetStr; // varias props, un getter
property OnClick: TNotifyEvent read FOnClick write FOnClick; // evento
class property Instancias: Integer read FInstancias; // de clase (class var/method)
// Obj.Nombre := 'x' llama a SetNombre. No se pueden pasar como parámetro var.
```

Interfaces

```
type
IFigura = interface(IInterface)    // IInterface = IUnknown
    ['{B1A5E5A0-1F0B-4B6E-9C3C-2A5A6C1D7E10}'] // GUID (Ctrl+Shift+G) para as/Supports
    function Area: Double;
    function GetNombre: string;
    property Nombre: string read GetNombre; // props sí; campos no
end;
TCirculo = class(TInterfacedObject, IFigura) // TInterfacedObject: refcount
private
    FR: Double;
    function IFigura.GetNombre = ObtenerNombre; // cláusula de resolución
public
    constructor Create(R: Double);
    function Area: Double;
    function ObtenerNombre: string;
end;
var F: IFigura;
begin
    F := TCirculo.Create(2); // refcount = 1
    ShowMessage(F.Nombre);
    if Supports(Obj, IFigura, F) then ...; F := Obj as IFigura;
end; // F sale de ámbito → refcount 0 → Destroy automático
```

TNoRefCountObject (10.4+) implementa IInterface sin contar; TComponent tampoco cuenta. Delegación: property Impl: TImpl read FImpl implements IFigura; .COM: IDispatch, TAutoObject.

Genéricos

```
type
TPila<T> = class
private
    FItems: TArray<T>;
public
    procedure Push(const Item: T);
    function Pop: T;
    function Max<K>(A, B: K): K; // método genérico
end;
TRepo<T: class, constructor> = class ... end; // restricciones (AND):
// class · record · constructor · IIntf · TBase · interface (13) · unmanaged (13)
var L: TList<Integer>; D: TDictionary<string, TObject>;
begin
    L := TList<Integer>.Create;
    try
        L.Add(1); L.AddRange([2, 3]); L.Sort; L.Reverse;
        if L.Contains(2) then L.Remove(2); L.IndexOf(3); L.Count; L[0];
        for var I in L do ...;
    finally L.Free; end;
    D := TDictionary<string, TObject>.Create; // TryGetValue AddOrSetValue ContainsKey Keys Values
    TArray.Sort<string>(Arr, TComparer<string>.Construct(
        function(const A, B: string): Integer begin Result := CompareText(A, B); end));
end;
```

System.Generics.Collections: TList<T> TObjectList<T> TDictionary<K,V> TObjectDictionary<K,V>
TQueue<T> TStack<T> TThreadList<T> TEnumerable<T> TPair<K,V>; System.Generics.Defaults:
TComparer<T> TEqualityComparer<T>.

Ciclo de vida de un objeto

- ```
Obj := TClase.Create(...) // NewInstance → InitInstance (todo a 0/nil) → cuerpo del
 // constructor (inherited Create primero) → AfterConstruction
Obj.Free // if Obj <> nil then Obj.Destroy → BeforeDestruction →
 // cuerpo del destructor (inherited al final) → FreeInstance
```
- Los objetos son **referencias** (punteros al heap). A := B NO copia el objeto. Una variable local de tipo clase **no** se inicializa a nil.
  - Nunca llames a Destroy directamente: usa **Free** (tolera nil). FreeAndNil(Obj) libera y pone la variable a nil (10.4+: solo acepta TObject).
  - Si el constructor lanza una excepción**, el destructor se ejecuta automáticamente y la memoria se libera: los destructores deben tolerar objetos a medio construir (Free sobre campos nil es seguro).
  - Declara siempre destructor Destroy; override; y termina con inherited;. Los constructores suelen empezar por inherited Create.
  - TComponent: constructor **virtual** Create(AOwner: TComponent); override; → necesario para creación polimórfica y streaming (.dfm).
  - Create puede llamarse sobre una variable existente (Obj.Create) para re-inicializar sin asignar memoria: **no lo hagas**; siempre TClase.Create.
  - Verificar si sigue vivo tras Free es imposible: la referencia queda **colgando**. Por eso FreeAndNil en campos y Assigned antes de usar.

Patrones try/finally

```
// 1 objeto local
var L := TStringList.Create;
try
 L.LoadFromFile('a.txt');
finally
 L.Free; // se ejecuta aunque haya excepción
end;

// Varios objetos: inicializar a nil + un solo finally (o anidar)
var A: TA := nil; var B: TB := nil;
try
 A := TA.Create; B := TB.Create;
finally
 B.Free; A.Free; // orden inverso; Free tolera nil
end;

// Objeto como campo
constructor TAgenda.Create;
begin
 inherited Create;
 FLista := TObjectList<TContacto>.Create(True);
end;
destructor TAgenda.Destroy;
begin
 FLista.Free; // seguro aunque Create fallara (FLista = nil)
 inherited; // SIEMPRE al final
end;
```

Patrones: fábrica, modal, FreeAndNil

```
// Función fábrica: el llamador es el dueño
function CrearConfig(const Ruta: string): TConfig;
begin
 Result := TConfig.Create;
 try
 Result.Cargar(Ruta);
 except
 Result.Free; raise; // no filtrar memoria si falla a medias
 end;
end;

// Formulario modal
var Dlg := TDlgOpciones.Create(nil);
try
 if Dlg.ShowModal = mrOk then Aplicar(Dlg);
finally
 Dlg.Free;
end;
FreeAndNil(FCache); // campos / globales que sobreviven al bloque
if Assigned(FCache) then FCache.Algo; // tras FreeAndNil la comprobación es fiable
```

Propiedad (ownership) y contenedores

- ```
Btn := TButton.Create(Self); // Owner = Self (el form) → lo destruirá él
Btn.Parent := Panel; // Parent = dónde se muestra (≠ Owner)
Frm := TForm2.Create(Application); // vive hasta cerrar la app (o Frm.Release)
Frm := TForm2.Create(nil); try ... finally Frm.Free; end; // sin Owner: tú liberas
L := TObjectList<TCliente>.Create(True); // OwnsObjects: libera sus elementos
L := TObjectList<TCliente>.Create(False); // solo referencias
D := TObjectDictionary<string, TObj>.Create([doOwnsValues]); // doOwnsKeys
SL := TStringList.Create; SL.OwnsObjects := True; SL.AddObject('k', Obj);
Col := TCollection.Create(TMiItem); // TCollection libera sus TCollectionItem
Lista.Extract(Obj); // saca sin liberar; Delete/Remove liberan si Owns
Lista.Clear; // libera todos si OwnsObjects
```
- TComponent.Owner libera a los componentes que posee al destruirse (Components[], ComponentCount). Notification(AComponent, opRemove) avisa a quien lo referencia; FreeNotification.
 - Regla: **quien crea, libera**, salvo que entregue la propiedad (Owner, lista con OwnsObjects, parámetro documentado).
 - TForm.Release pospone la destrucción hasta que terminen los handlers de evento (seguro desde OnClick); Action := caFree en OnClose.

Tipos gestionados (liberación automática)

- `string`, **arrays dinámicos**, **interfaces**, `Variant`, **métodos anónimos** y los **records que los contienen**: contados por referencia; el compilador los inicializa (`''/nil`) y libera al salir de ámbito. Nunca `FreeMem` sobre ellos.
- `SetLength(Arr, 0)` o `Arr := nil` libera un array dinámico; `S := ''` una cadena. Copy-on-write: modificar una copia no afecta a la otra.
- **Records gestionados** (10.4+): `class operator Initialize(out Dest: T) / Finalize(var Dest: T) / Assign(var Dest: T; const [ref] Src: T) → RAII`, punteros inteligentes. En 13 pueden omitirse los parámetros (Self implícito).
- `Default(T)` devuelve el valor cero de cualquier tipo; `Finalize(V)` y `Initialize(V)` manuales solo para memoria cruda (`GetMem`).

```
type
  TAutoFree = record                               // puntero inteligente mínimo
    Obj: TObject;
    class operator Initialize(out Dest: TAutoFree);
    class operator Finalize(var Dest: TAutoFree);
  end;
class operator TAutoFree.Initialize(out Dest: TAutoFree); begin Dest.Obj := nil; end;
class operator TAutoFree.Finalize(var Dest: TAutoFree);   begin Dest.Obj.Free; end;
```

Interfaces y conteo de referencias

```
var O: TCirculo; I: IFigura;
O := TCirculo.Create;           // refcount 0 (aún no hay referencia de interfaz)
I := O;                         // _AddRef → 1
I := nil;                       // _Release → 0 → ¡0 destruido! O queda colgando
// REGLA: no mezclar referencias de objeto e interfaz al mismo TInterfacedObject
F := TCirculo.Create;           // F: IFigura → la vida la gobierna el refcount
TComponent / TNoRefCountObject (10.4) implementan IInterface SIN contar → liberar con Free
[weak] FPadre: IPadre;           // no incrementa; se pone a nil al destruir el objeto
[unsafe] FTmp: IFigura;          // no incrementa; NO se gestiona (como un puntero)
```

Ciclos (A→B→A con interfaces) nunca llegan a 0: rompe el ciclo con `[weak]` (10.1+). `Supports(Obj, IX)` crea una referencia temporal: con `TInterfacedObject` sin refs puede **destruir** el objeto al salir.

Historia: ARC y modelo unificado

- **XE4 – 10.3**: los compiladores móviles (iOS/Android/Linux) usaban **ARC** (conteo automático de objetos); `Free` se convertía en `:= nil`; `DisposeOf` forzaba el destructor.
- **10.4 Sydney (2020)**: ARC **eliminado**; todas las plataformas usan el modelo clásico (`Create/Free`). `DisposeOf` equivale a `Free`; `AUTOREFCOUNT/NEXTGEN` ya no se definen; cadenas 1-based en móvil.
- `TObject.Free`, `TComponent.Owner`, `TInterfacedObject` y los tipos gestionados son hoy el único modelo. `[weak]`/`[unsafe]` siguen existiendo para interfaces.

Memoria de bajo nivel y fugas

```
type PRec = ^TRec; var P: PRec;
New(P); try P^.X := 1; finally Dispose(P); end;           // tipado; gestiona campos
GetMem(Buf, 1024); FreeMem(Buf);                          // crudo; FreeMem sin tamaño
Buf := AllocMem(1024); ReallocMem(Buf, 2048);             // AllocMem → ceros
FillChar(R, SizeOf(R), 0); Move(Src, Dst, N); ZeroMemory(@R, SizeOf(R));
ReportMemoryLeaksOnShutdown := True; // en el .dpr: FastMM avisa de fugas al salir (debug)
GetMemoryManager / SetMemoryManager // FastMM4 (defecto), FastMM5, rpmalloc (13.2, Arm64EC)
AllocMemCount (legado) TMemoryStream.Size/Capacity
```

- **Errores típicos**: usar un objeto tras `Free`; `Free` sobre un objeto que posee otro (doble liberación); olvidar `inherited` en el destructor; liberar en `OnDestroy` lo que no se creó en `OnCreate`; devolver un objeto local liberado; filtrar en bucles sin `try/finally`.
- Detectar fugas: `ReportMemoryLeaksOnShutdown`, `FastMM5/FastMM4 FullDebugMode`, `DeLeaker`, `madExcept`. Doble free: `FreeAndNil + {$D+}`.

Hilos y sincronización

```
TThread.CreateAnonymousThread(
  procedure
  begin
    Trabajo;                                           // en segundo plano
    TThread.Synchronize(nil, procedure begin Label1.Caption := 'ok'; end); // bloquea
    TThread.Queue(nil, procedure begin ... end);       // asíncrono, no bloquea
  end).Start;                                         // FreeOnTerminate = True
type TMiHilo = class(TThread)
  protected procedure Execute; override; // comprobar Terminated en bucles
end;
H := TMiHilo.Create(True); H.FreeOnTerminate := True; H.Start; // o H.WaitFor; H.Free
TTask.Run(procedure begin ... end); F := TTask.Future<Integer>(function: Integer begin ... end)
TParallel.For(0, 99, procedure(I: Integer) begin ... end); // System.Threading
TMonitor.Enter(Obj); try ... finally TMonitor.Exit(Obj); end; // cualquier TObject
CS := TCriticalSection.Create; CS.Enter; ... CS.Leave; // System.SyncObjs
TInterlocked.Increment(N); AtomicIncrement(N); AtomicExchange(A, B);
```

La UI (VCL/FMX) solo se toca desde el hilo principal. Cada hilo tiene su propia `threadvar`. `TThread.CurrentThread.ThreadID`, `TThread.Sleep`, `TEvent`, `TSemaphore`, `TLightweightMREW`.

Árbol de clases principal de la VCL (Windows)

```
TObject // System: raíz de todo (ClassName, Free, InheritsFrom...)
├─ Exception → EAbort · EConvertError · EInvalidCast · EInOutError · EArgumentException
│   ├── EHeapException → EOutOfMemory · EInvalidPointer
│   └── EExternal → EAccessViolation · EIntError (EDivByZero, ERangeError, EIntOverflow)
│       └── EMathError (EInvalidOp, EZeroDivide, EOverflow, EUnderflow)
├─ TInterfacedObject · TNoRefCountObject · TAggregatedObject · TThread · TTask (ITask)
├─ TList · TObjectList (System.Contnrs) · TList<T> · TObjectList<T> · TDictionary<K,V>
├─ TStream → THandleStream → TFileStream · TMemoryStream · TStringStream · TBytesStream
├─ TReader · TWriter · TStringBuilder · TEncoding · TRttiContext
├─ TPersistent // Assign, streaming de propiedades published
│   ├── TStringList
│   ├── TCollection → (TCollectionItem también es TPersistent)
│   ├── TGraphicsObject → TFont · TPen · TBrush
│   ├── TCanvas
│   ├── TInterfacedPersistent → TGraphic → TBitmap · TIcon · TMetafile · TWICImage
│   └── TPicture
│       ├── TJPEGImage · TPngImage · TGIFImage
├─ TComponent // Owner, Name, Tag, Components[], streaming .dfm
│   ├── TApplication · TScreen · TDataModule · TTimer
│   ├── TBaseImageList → TCustomImageList → TImageList
│   ├── TMenu → TMainMenu · TPopupMenu
│   ├── TMenuItem
│   ├── TBasicAction → TContainedAction → TCustomAction → TAction
│   ├── TCustomActionList → TActionList
│   ├── TCommonDialog → TOpenDialog → TSaveDialog · TFontDialog · TColorDialog · TTaskDialog
│   ├── TDataSource · TDataSet → TClientDataSet · TFDDataSet → ... → TFDQuery · TFDTable (FireDAC)
├─ TControl // Left/Top/Width/Height, Parent, Align, Anchors, Visible,
│   // Enabled, Cursor, Hint, OnClick, OnMouse*, Font, Color
│   ├── TGraphicControl // SIN HWND: se pintan sobre el Parent; sin foco ni hijos
│   │   ├── TCustomLabel → TLabel · TImage · TShape · TBevel · TPaintBox · TSpeedButton · TSplitter
│   └─ TWinControl // CON HWND (Handle): foco, hijos, TabOrder, mensajes
│       ├── TButtonControl → TCustomButton → TButton · TBitBtn
│       │   └── TCustomCheckBox → TCheckBox · TRadioButton
│       ├── TCustomEdit → TEdit · TMaskEdit · TLabelEdit · TButtonEdit
│       │   └── TCustomMemo → TMemo · TCustomRichEdit → TRichEdit
│       ├── TCustomListControl → TCustomMultiSelectListControl → TCustomListBox → TListBox
│       │   └── TCustomComboBox → TCustomComboBox → TComboBox
│       │       └── TCustomListBoxEx
│       ├── TCustomControl // Canvas propio + HWND
│       │   ├── TCustomPanel → TPanel · TCustomGroupBox → TGroupBox · ... → TRadioGroup
│       │   ├── TCustomGrid → TCustomDrawGrid → TDrawGrid → TStringGrid · TCustomDBGrid → TDBGrid
│       │   ├── TToolWindow → TToolBar · TCoolBar
│       │   ├── TCustomControlBar → TControlBar
│       │   ├── TCustomTabControl → TTabControl · TPageControl
│       │   ├── TTabSheet
│       │   ├── TCustomListView → TListView · TCustomTreeView → TTreeView · TStatusBar · TProgressBar
│       │   ├── TTrackBar · TUpDown · TDateTimePicker · TMonthCalendar · THeaderControl · TScrollBar
│       │   ├── TStaticText · TCustomTransparentControl · ToleControl → TWebBrowser (ActiveX)
│       └─ TScrollingWinControl → TScrollBox · TCustomFrame → TFrame
│           └─ TCustomForm → TForm · TCustomDockForm
```

Unidades VCL / RTL más usadas

Unidad	Contenido
System System.SysUtils System.Classes	TObject, tipos base · conversiones, excepciones, fechas, ficheros · TComponent, TStream, TString, TThread, TList
System.Generics.Collections System.Generics.Defaults	TList<T>, TDictionary ... · comparadores
System.Math System.StrUtils System.DateUtils System.IOUtils System.Types System.UITypes	matemáticas · cadenas · fechas · TPath/TFile/TDirectory · TPoint/TRect · TColor, TModalResult, mrOk
System.Rtti System.TypeInfo System.JSON System.Threading System.SyncObjs System.Net.HttpClient	RTTI · info de tipos · JSON · tareas paralelas · sincronización · HTTP
Vcl.Forms Vcl.Controls Vcl.StdCtrls Vcl.ExtCtrls Vcl.ComCtrls	TForm, TApplication, TScreen · TControl, TWinControl · botones, edits, listas · TPanel, TImage, TTimer, TShape · TListView, TTreeView, TPageControl, TStatusBar
Vcl.Graphics Vcl.Dialogs Vcl.Menus Vcl.ActnList Vcl.Grids Vcl.Buttons Vcl.Imaging.*	TCanvas, TBitmap, TFont · diálogos, ShowMessage, MessageDlg · menús · acciones · TStringGrid · TSpeedButton, TBitBtn · JPEG/PNG/GIF
Vcl.Themes Vcl.Styles Vcl.DBGrids Vcl.DBCtrls Data.DB FireDAC.Comp.Client	estilos VCL · controles DB · TDataSet, TField · TFDConnection, TFDQuery
Winapi.Windows Winapi.Messages Winapi.ShellAPI Winapi.ActiveX	API Win32, HWND, mensajes WM_*, ShellExecute, COM

Formularios, aplicación y eventos típicos

```
Application.Initialize; Application.MainFormOnTaskbar := True; // .dpr
Application.CreateForm(TFormMain, FormMain); Application.Run;
Application.ProcessMessages; Application.Terminate; Application.OnException := ...;
Application.MessageBox('Texto', 'Título', MB_OK or MB_ICONINFORMATION);
ShowMessage('Hola'); MessageDlg('¿Seguro?', mtConfirmation, [mbYes, mbNo], 0) = mrYes;
InputQuery('Título', 'Nombre:', S); Screen.Cursor := crHourGlass;
// Eventos de un TForm, en orden: OnCreate → OnShow → OnActivate → OnResize/OnPaint ...
// → OnCloseQuery(var CanClose) → OnClose(var Action) → OnDeactivate → OnHide → OnDestroy
procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction); // caNone caHide caFree
Dlg := TDlg.Create(Self); try if Dlg.ShowModal = mrOk then ...; finally Dlg.Free; end;
Frm := TFrm.Create(Application); Frm.Show; // ModalResult := mrOk cierra un modal
B := TButton.Create(Self); B.Parent := Self; B.SetBounds(10, 10, 75, 25); // en ejecución
B.Caption := 'OK'; B.OnClick := BotonClick; // procedure BotonClick(Sender: TObject)
// Eventos: OnClick OnDblClick OnChange OnEnter OnExit OnKeyDown(Key: Word; Shift)
// OnKeyPress(Key: Char) OnMouseDown/Move/Up(Button, Shift, X, Y) OnMouseWheel OnPaint
// OnResize OnTimer OnDragDrop OnContextPopup OnMouseEnter/Leave
// Propiedades: Align (alClient...) Anchors AutoSize Caption/Text Color Font Enabled Visible
// TabOrder TabStop Hint/ShowHint Cursor Tag Margins AlignWithMargins Constraints KeyPreview
procedure WMGetMinMaxInfo(var M: TWMGetMinMaxInfo); message WM_GETMINMAXINFO; // Perform / Post
Canvas.Brush.Color := clRed; Canvas.FillRect(ClientRect); Canvas.TextOut(0, 0, 'Hi'); Invalidate;
TStyleManager.TrySetStyle('Windows11 Modern Dark'); // Vcl.Themes · High DPI: PerMonitorV2
```

Árbol de clases principal de FireMonkey (FMX, multiplataforma)

```
TObject
├─ TInterfacedPersistent → TBitmap · TCanvas (FMX.Graphics)
├─ TPersistent → TBrush · TFont · TGradient · TPosition · TBounds · TPathData
├─ TComponent
│   ├─ TApplication · TScreen · TDataModule · TBindingsList (LiveBindings)
│   ├─ TCustomActionList → TActionList · TAction · TCommonDialog → TOpenDialog · TSaveDialog
│   ├─ TBaseImageList → TCustomImageList → TImageList (FMX.ImgList)
│   └─ TFmxObject (FMX.Types) // raíz FMX: Children, Parent, Clone, Release, AddObject
│       ├─ TTimer · TStyleBook · TMainMenu · TPopupMenu · TGestureManager
│       └─ TAnimation (FMX.Ani) → TFloatAnimation · TColorAnimation · TRectAnimation
│           └─ TBitmapAnimation · TPathAnimation · TGradientAnimation
├─ TEffect (FMX.Effects) → TShadowEffect · TBlurEffect · TGlowlEffect · TInnerGlowEffect
│   └─ TBevelEffect · TReflectionEffect · TFilterEffect → ...
├─ TCommonCustomForm → TCustomForm → TForm · TCustomPopupForm
│   └─ TCustomForm3D → TForm3D
├─ TControl3D (FMX.Types3D) → TShape3D → TCube · TSphere · TPlane · TCylinder · TCone
│   └─ TText3D · TModel3D · TMesh · TCamera · TLight · TDummy
├─ TControl (FMX.Controls) // Position(X,Y: Single), Width/Height, Align: TAlignLayout,
│   // Anchors, Margins, Padding, Opacity, Scale, RotationAngle,
│   // HitTest, ClipChildren, Canvas, Paint, OnClick, OnTap...
│   └─ TShape (FMX.Objects) → TRectangle · TRoundRect · TCircle · TEllipse · TLine
│       └─ TPath · TPie · TArc · TCalloutRectangle
├─ TImage · TText · TPaintBox · TSelection · TFrame · TViewport3D
├─ TCustomWebBrowser → TWebBrowser
├─ TLayout (FMX.Layouts) → TScaledLayout · TGridLayout · TFlowLayout · TGridPanelLayout
├─ TStyledControl // StyleLookup, StylesData: aspecto por estilos (TStyleBook)
│   └─ TPresentedControl // Model + PresentationProxy; ControlType = Styled | Platform
│       └─ TPresentedTextControl → TLabel · TCheckBox · TRadioButton · TGroupBox
│           └─ TCustomEdit → TEdit · TClearingEdit · TCustomEditBox → TNumberBox · TSpinBox
│               └─ TCustomComboEdit → TComboEdit · TComboTrackBar
│                   └─ TDateEdit · TTimeEdit · TMaskEdit (13)
│       └─ TCustomPresentedScrollBar → TCustomMemo → TMemo
│           └─ TCustomListBox → TListBox · TComboBox
│               └─ TCustomGrid → TGrid · TStringGrid
│       └─ TPanel · TCalloutPanel · TSwitch · TCustomTrack → TTrackBar · TScrollBar
│           └─ TProgressBar · TMultiView
├─ TTextControl → TTabItem · TMenuItem · TListBoxItem · TTreeViewItem · TStatusBar
├─ TCustomScrollBar → TScrollBar · TVertScrollBar · THorzScrollBar · TFramedScrollBar
├─ TCustomComboBox → TComboBox · TCustomTabControl → TTabControl
├─ TCustomTreeView → TTreeView · TAdapterListView → ... → TListView (FMX.ListView)
├─ TToolBar · TExpander · TCalendar · TArcDial · TColorPanel · TImageControl
```

Jerarquía simplificada (se omiten clases intermedias). Todo control FMX puede contener hijos (Children), tiene Opacity, RotationAngle y coordenadas Single; no hay HWND por control: cada formulario es una única ventana nativa y FMX pinta sus controles (salvo ControlType = Platform).

Unidades FMX más usadas

Unidad	Contenido
FMX.Forms FMX.Types FMX.Controls FMX.Controls.Presentation	TForm, TApplication · TFmxObject, TTimer, TAlignLayout, TBounds · TControl, TStyledControl · TPresentedControl
FMX.StdCtrls FMX.Edit FMX.Memo FMX.ListBox FMX.ListView FMX.Grid FMX.TabControl	TButton, TLabel, TCheckBox, TPanel, TTrackBar · TEdit · TMemo · TListBox, TComboBox · TListView · TGrid, TStringGrid · TTabControl
FMX.Layouts FMX.Objects FMX.Graphics FMX.Ani FMX.Effects FMX.Styles	TLayout, TScrollBar · formas, TImage, TText · TBitmap, TCanvas, TBrush · animaciones · efectos · TStyleBook
FMX.Dialogs FMX.DialogService FMX.Menus FMX.ActnList FMX.ImgList FMX.MultiView	diálogos (ShowMessage) · diálogos asíncronos · menús · acciones · listas de imágenes · TMultiView
FMX.Platform FMX.Platform.Win/Android/iOS/Mac FMX.Helpers.Android FMX.Skia	servicios de plataforma (TPlatformServices), interop nativa · render Skia (12+)
FMX.VirtualKeyboard FMX.Media FMX.Sensors FMX.WebBrowser FMX.Maps FMX.PushNotification	teclado virtual · audio/video/cámara · sensores · navegador · mapas · push
System.Permissions System.Sensors System.Notification FMX.Advertising	permisos Android/iOS · sensores · notificaciones locales

VCL vs FMX: equivalencias y diferencias

Concepto	VCL	FMX
Posición / tamaño	Left Top Width Height (Integer, px)	Position.X Position.Y Width Height (Single, escalables)
Alineación	Align: TAlign = alClient...	Align: TAlignLayout = Client, Top, Contents, Scale, Fit...
Texto	Caption / Text	Text (casi siempre); TextSettings para fuente/color
Color	Color: TColor (clRed, BGR)	Fill.Color: TAlphaColor (TAlphaColors.Red, ARGB); fondos via estilo
Aspecto	tema de Windows / Vcl.Styles	estilos (TStyleBook, StyleLookup); ControlType = Platform usa control nativo
Jerarquía visual	Parent: TWinControl (requiere HWND)	Parent: TFmxObject: cualquier control puede contener otros
Dibujo	Canvas sobre HDC; OnPaint	Canvas.BeginScene ... EndScene; OnPaint(Sender, Canvas, ARect); vectorial, antialias
Imágenes	TBitmap GDI, TPicture	TBitmap propio (sin Handle), TImage.Bitmap, MultiResBitmap
Efectos / animación	no (código propio)	TShadowEffect ...; TFloatAnimation; AnimateFloat('Opacity', 0, 0.3)
Diálogos	ShowMessage, MessageDlg (modales)	ShowMessage; en móvil usar TDialogService.MessageDialog(...) procedure(const R) ... (asíncrono)
Modales	ShowModal: TModalResult	ShowModal(procedure(R: TModalResult) ...) en Android/iOS; ShowModal síncrono en escritorio
Eventos ratón/táctil	OnMouseDown(Button, Shift, X, Y: Integer)	OnMouseDown(..., X, Y: Single); OnTap; OnGesture; Touch
Plataformas	solo Windows (32/64/Arm64EC)	Windows, macOS, iOS, Android, Linux (FMXLinux)

Versiones de Delphi (CompilerVersion)

Versión	Año	Hitos	Ver
Delphi 1 / 2 / 7	1995 / 1996 / 2002	Win16 · Win32 · VCL clásica	8 / 9 / 15
2009 / 2010 / XE	2008-2010	Unicode (string = UnicodeString), genéricos, anónimos · RTTI extendido	20 / 21 / 22
XE2 ... XE8	2011-2015	Win64, FireMonkey, macOS · iOS (ARC) · Android · [1,2,3] arrays	23 ... 29
10 Seattle · 10.1 Berlin · 10.2 Tokyo	2015-2017	[weak] / [unsafe], UTF8String en móvil · Linux	30 · 31 · 32
10.3 Rio · 10.4 Sydney	2018 · 2020	variables inline, inferencia · records gestionados, ARC eliminado , LSP	33 · 34
11 Alexandria	2021	literales binarios %, separador _, IDE High-DPI, macOS ARM	35
12 Athens (12.1-12.3)	2023-2025	strings multilínea ' ' ', literales > 255, Skia, IDE 64 bits (12.3)	36
13 Florence (13.1 · 13.2)	2025-2026	if ternario, NameOf, is not/not in, noreturn, \$PUSHOPT · Windows on Arm (13.1) · LLVM 20 Linux (13.2)	37

Atajos del IDE más útiles (RAD Studio)

Atajo	Acción
Ctrl+Shift+C	completar clase: genera cuerpos de métodos / getters-setters de propiedades
Ctrl+Shift+↑ / ↓	saltar entre declaración e implementación
Ctrl+Clic · Alt+←/→	ir a la declaración · volver
Ctrl+Espacio · Ctrl+Shift+Espacio	completar código · parámetros
Ctrl+J	plantillas de código (begin...end, try...finally, class...)
Ctrl+Shift+G	insertar un GUID nuevo (interfaces)
F9 · F8 · F7 · F4 · F5	ejecutar · paso a paso · entrar · hasta cursor · punto de ruptura
Ctrl+F9 · Shift+F9	compilar · construir (build) el proyecto
F12 · F11 · Alt+F11	código ↔ formulario · Object Inspector · añadir unidad al uses
Ctrl+Shift+E · Ctrl+Shift+V · Ctrl+Shift+M	refactorizar: renombrar · declarar variable · extraer método
Ctrl+/ · Ctrl+Shift+I / U	comentar/descomentar · indentar / desindentar
Ctrl+Shift+F · Alt+G	buscar en ficheros · ir a línea

Ficheros de un proyecto Delphi

Extensión	Qué es
.dpr · .dproj · .groupproj	código fuente del proyecto (program/library) · configuración MSBuild (plataformas, opciones) · grupo
.pas · .inc	unidad de código · fichero incluido con {\$I}
.dfm · .fmx	formulario / data module VCL · FMX (texto o binario; se enlaza con {\$R *.dfm})
.dcu · .dcp · .bpl · .dpk	unidad compilada (por plataforma y configuración) · símbolos de paquete · paquete en ejecución (DLL) · fuente de paquete
.res · .rc · .drc	recursos compilados (icono, versión, {\$R x.res}) · fuente de recursos · resourcestrings generadas
.exe · .dll · .so · .dylib · .apk/ .aab · .ipa	binarios Windows · Linux/Android · macOS/iOS · paquetes Android · iOS
.local · .identcache · __history/ · __recovery/	estado del IDE y copias: no versionar
.tds · .map · .rsm	símbolos de depuración (Win32) · mapa del enlazador · símbolos remotos (Win64/móvil)

Buenas prácticas resumidas

- Cada Create sin Owner tiene su Free en un try...finally; campos → FreeAndNil en el destructor; destructor Destroy; override; + inherited.
- Usa const en parámetros string/record/interface; var/out solo si hace falta. Prefiere TArray<T>, TList<T> y TDictionary<K,V>.
- Comprueba Assigned(F0nEvento) antes de disparar un evento. Nunca toques la UI desde un hilo secundario (Synchronize/Queue).
- No mezcles referencias de objeto e interfaz al mismo TInterfacedObject. Rompe ciclos con [weak].
- Round es bancario; Extended cambia de tamaño por plataforma; LongInt no es portable → usa Integer/Int64/NativeInt.
- Activa {\$R+} {\$Q+} en Debug y ReportMemoryLeaksOnShutdown. Evita with, variables globales y cast duros TClase(Obj) sin is.
- strict private para encapsular de verdad; sealed/final cuando no quieras herencia. Documenta la propiedad (quién libera) en cada API.
- Rutinas cortas, una unidad por responsabilidad, interface con lo mínimo público; uses de implementation para romper ciclos.

Fuentes

docwiki.embarcadero.com (RAD Studio Florence): Fundamental Syntactic Elements · Simple Types · String Types · 64-bit Windows Data Types · Compiler Attributes · Constraints in Generics · Expressions · Automatic Reference Counting (Rio) · FireMonkey Component Library · API Vcl.Forms / FMX.Controls.Presentation / FMX.Memo. blogs.embarcadero.com: RAD Studio 13 Florence, Delphi 12 string literals, Summary 2025. blog.marcocantu.com: weak/unsafe (2016), ternary (2025). Fecha de elaboración: octubre 2026.