Delphi modern language features

Pascal Conference
September 2025

Andrea Magni
andrea.magni@gmail.com

Andrea Magni

Monza, Italy - https://andreamagni.eu

Education: Computer Engineer
Artificial Intelligence and Robotics
Polytechnic of Milan

Job: Freelance / Trainer / Consultant

Platforms: Desktop, Server, Web, Mobile, IoT

Open Source: TFrameStand, TFormStand, FMXER
MARS-Curiosity REST library

Author: “Delphi GUI programming with FireMonkey”,
Packt Publishing, November 2020

embarcadero MVP Delphi

New in Delphi 13 (1)

- New syntax: ternary operator
  - **if** (cond) **then** Value1 **else** Value2
  - **if** (cond) **then** Stmt1 **else** Stmt2
- Example:

```
var LCount := if Assigned(LList) then LList.Count else 0;
// IfThen(Assigned(LList), LList.Count, 0)

ShowMessage(if LCount = 1 then ‘developer’ else ‘developers’)

ShowMessage(
  if LCategory = ‘One’ then CountItems(1) else CalcQuantity(LCategory)
);
```

New in Delphi 13 (2)

- New syntax: **is not**
- New function: **NameOf**
- Example:

```
if Sender is not TButton then
  ShowMessage(NameOf(Sender) + ' is not a TButton');
```

New in Delphi 13 (3)

- New syntax: **not in**
- Example:

```
type
  TOption = (One, Two, Three, None);
  TOptions = set of TOption;

var LOptions: TOptions := [One, Three];
var LSelected := Two;

if LSelected not in LOptions then
  ShowMessage('Not available');
```

Modern language features

- multiline string literals / decimal literals
- inline variable declarations (with type inference)
- anonymous methods
- records with methods and operators
- intrinsic types helpers
- class, record and enum helpers
- RTTI
- generics
- ternary operator and NameOf

Code quality principles (1)

- **KISS** (Keep It Simple, Stupid)
  - prefer the simplest construct that solves the problem
- **YAGNI** (You Aren't Gonna Need It)
  - don't add capability until it's actually needed
- **DRY** (Don't Repeat Yourself)
  - use features that reduce duplication without obscuring intent
- **Low Coupling** & **High Cohesion**
  - aim for low coupling, high cohesion; avoid features that tightly bind modules or mix concerns
- **Principle of Least Astonishment**
  - code should behave as a typical reader expects

Code quality principles (2)

- **Readability & Maintainability**
  - optimize for the reader; future maintainers outweigh short-term cleverness
- **Testability**
- **Performance vs. Premature Optimization**
- **Backward Compatibility & Portability**
- **Observability**
- **Consistency & Conventions**

Modern features

- Generics: typed collections
  - TList<T>, TDictionary<K,V>
- Anonymous and Generics
  - parallel programming library
    - TTask.Run(<TaskToExecute>)
    - TTask.Future<T>(<FunctionReturningT>)
  - System.Messaging (publish/subscribe)
- Capturing vs callback parameters tweaks
- Better scope definition: inline variables
- Type Helpers
- RTTI