DELF · 26 OCT

The big ideas

What’s in front of us beyond the next to-do: ideas, not designs yet

Each slide names the note where the idea is recorded · nothing here is built

PART ONE

The Present

What’s done so far, as a proof of concept

THE PRESENT · DELFC, DELF TO DELPHI

What works today, compiled on dcc32 and dcc64

Syntax

λ lambdas, arrow types, tuples and destructuring, named record literals, UFCS, free generic functions.

Sums and match

Exhaustive, nested patterns, strings, narrowing with if … is, a trapdoor for bad tags.

Optional

Present or Absent, with Map, AndThen and OrDefault, and an Exit checked against it.

Inference

Type arguments from inside and outside, _ for the rest; dcc32 infers neither.

Conditionals

Every {$IFDEF} branch translated or checked; includes followed.

Proof

66 tests, 43 dcc32 experiments, 5 examples; 15,273 corpus files in 9 minutes.

Plain Delphi passes through byte for byte; a renamed file builds as it did.

poc/README.md · SPEC · experiments/README.md · the Finding the start deck shows it in use

PART TWO

The Future

The big ideas in front of us: ideas, not designs yet

THE USER'S IDEA · 26 OCT

Objects as nullable things: live, maybe, or retired

```
TFoo             // a live object: never nil, valid until retired
Optional<TFoo>   // maybe one: the doors Optional already has
Free             // retires it, for good: the name can't be used again
```

Non-nil types

"To invent" (user, 19 Oct). So far only discussed as an obstacle.

Optional for classes

Could be a bare pointer, nil as Absent, once TFoo itself is never nil.

Free consumes

Using or freeing it again is a compile error. Decided in SPEC §7, not built.

concepts-map.md §1 · optional.md · SPEC §7 · an idea, not a design

THE USER'S IDEA · WHAT MAKES IT HARD

Where a reference can be nil, or worse

Half-built objects

A field is nil until the constructor sets it. If a constructor raises, Delphi may call Destroy on what was built: a prediction, to test.

Freed by someone else

A TComponent's owner frees it; FreeAndNil through a getter leaves the field dangling (🧪 experiment 16).

Plain Delphi's references

Anything from the RTL, the VCL or a library could be nil. At the boundary each would be Optional until checked.

Arrays and generics

A new TArray<TFoo> is all nil; a generic T with a class constraint says nothing about nil.

concepts-map.md §4 · evidence-community.md §3 · experiment 16 · rung 3

BIG IDEA · OWNERSHIP

No use after free, no double free, no leaks

```
var L := TStringList.Create;    // owned: no Free, no try/finally written
L.Add('x');
```

Decided

Owned locals freed at scope end; Free consumes; ownership moves on assignment.

The evidence

A const string read after it was freed (🧪 15c); FreeAndNil through a getter (🧪 16).

Not yet

Lifetimes, so no escape analysis; borrows only in parameters.

SPEC §7 · ownership.md · todo-lifecycle.md · rung 5 · test: Game of Life

BIG IDEA · ERRORS AS VALUES

A Result type, and composing the steps that fail

```
Optional<T>      // there, or not: built
TResult<T, E>    // a value, or why not: not yet
O.AndThen(F)      // bind: the next step only if this one worked
```

The name

Result is already Delphi’s: every function’s answer. The type needs another.

Composition

The Kleisli family: steps that may fail, chained without nesting.

The Try family

Delphi’s Boolean-and-out pattern, replaced by a value.

brief-values-try.md · kleisli-in-delf.md · todo-naming.md · SPEC §5, §12

BIG IDEA · CONSTRUCTION

Constructors that guarantee what a field holds

Word	What it would promise	Status
required	set by every constructor	decided
immutable	never changed after construction	decided
owned	freed by its object’s destructor	decided
once	set exactly once	noted
final	not inheritable? write-once? to pin down	open

The question underneath: what a destructor sees when a constructor raised half-way.

SPEC §3.3 · concepts-map.md §1, §4 · rung 3 · test: the user’s Turbo Pascal conversion

BIG IDEA · TYPES THAT SAY MORE

Constraints, newtypes, phantom types, interfaces

Constraints and traits

A T that can be compared, ordered or added: Eq, Ord, Num. Nothing decided.

Newtypes

A distinct type, not an alias: Delphi’s type X = type Y gives no safety (🧪 experiment 4).

Phantom types

Units, states and ids told apart at no cost. Experiment 9 is planned.

Non-refcounted interfaces

To invent (user, 19 Oct), so Delphi’s [unsafe] has no job in Delf.

todo-constraints.md · todo-interfaces.md · handover-1.md · concepts-map.md §1

BIG IDEA · STAGE 2, CONTROL FLOW

Coroutines, async and threads

Coroutines and yield

Generators written as loops; await is the same transform. The user's Dev Days talk is the starting point.

Async, compositional

Cold TAsync values: Map, AndThen, Race, Ensure. The hoist demo ran (🧪 experiment 17).

Threads

Open: without lifetimes there is no escape analysis.

Cancellation and its memory questions are parked: "beyond my level of competence" (user, 8 Oct).

todo-async.md · apps/hoist · todo-threading.md · parked-async-cancellation.md (🅿)

BIG IDEA · PASKELL

A stricter level: purity, laws, witnesses

Pure functions
Pure getters first: a property that can’t change anything.

Laws
Functor and Kleisli laws stated beside the code, and tested for you.

Compile-time witnesses
Delf’s verdict made enforceable by dcc32 itself (the user’s idea).

Proofs
FlatMap’s three tiers proved in Lean 4, no sorry.

paskell.md · proofs/Cousins.lean · flatmap-tests.md · todo-tables-are-functions.md §15

BIG IDEA · REACH

Real libraries, and tools that show the types

Calling real libraries

Today delfc reads a used unit's types, one level. Its routines are the next step.

Type hints

{:: T} helps inference, and stops the build if wrong; {:: ?} asks. An LSP could show the same.

Edit and debug as Delf

Breakpoints and errors on the .delf line, not the output's.

delfc in Delphi

The aspiration, with no urgency: Python is fine for now.

delf-calls-delf.md §5 · types-and-conversions.md · todo-ide.md · todo-diagnostics.md · poc/README.md

WHERE TO START · CLAUDE'S VIEW

One idea, not two: the life of an object reference

Born live, maybe absent, retired for good. Non-nil objects and ownership share their hard cases, and together they would let Delf say: no nil dereference, no use after free.

First step: a design note, as optional.md was. Nothing built until the design is the user's.

A recommendation, for the user to accept or not