# Make it as simple as possible, but not simpler: The Programming Language Oberon

Niklaus Wirth

Oberon Day

CERN, 10.3.2004

# **Part 1**: Historical perspective

The ambivalent relationship between physicists and computer scientists:
- Physicists caused the construction of large computers, invented the Web, thereby pushing technology.
- Physicists clang to inadequate tools for too long, thereby hindering progress.

Fortran: no data structures, rigid format, no recursion
Computers: geared to Fortran
Teaching programming at ETHZ in 1968: Fortran, assembler, (Algol)

# Design of Pascal, 1968-70

- Algol-like phrase structure (syntax)
- Generality of expressions
- Conditions and the type Boolean
- Procedures and functions, recursion
- The concept of locality (block structure)
- Data structures: Array, Record, Set, File, Pointer

Pascal designed with three primary goals:
- A tool for decent teaching
- A tool for designing system software (compilers, operating systems)
- Compactness and efficiency

These goals required a systematic language structure, a concentration on what is essential, avoidance of unnecessary bells and whistles

# Design of Modula-2 1977-1979

- Pascal as basis
- Additional standard data types
- Modules, interfaces, information hiding, separate compilation
- Elements for parallel programming

Program development in large teams, software engineering
Reasoning about programs with assertions and loop invariants

```
PROCEDURE Reci(x: REAL): REAL;   (*0 < x < 2*)
        VAR y, c: REAL;
BEGIN y := 1.0; c := 1.0 – x;
        WHILE c > e DO
                (* y×x = 1-c,  0 < |c| < 1 *)
                y := y × (1.0+c); c := c×c
        END
        (* (1-e)/x <= y < 1/x *)   RETURN y
END Reci
```

```
PROCEDURE Sqrt(x: REAL): REAL;   (*0 < x < 2*)
        VAR y, c: REAL;
BEGIN y := x; c := 1.0 – x;
        WHILE c > e DO
                (*y2 = x×(1-c),  c >= 0*)
                y := y × (1.0 + 0.5×c);
                c := c × c × (0.75 + 0.25×c)
        END ;
        RETURN y    (*x×(1-e) <= y2 < x*)
END Sqrt
```

# Design of Oberon 1986-1988

- Modula-2 as basis
- Discarding several inessential features
- Adding type extensibility (inheritance)
- Simplify syntax

Swimming against the current (PL/1, Ada, C++) :
Reduce rather than increase complexity

# Complexity of syntax of programming languages

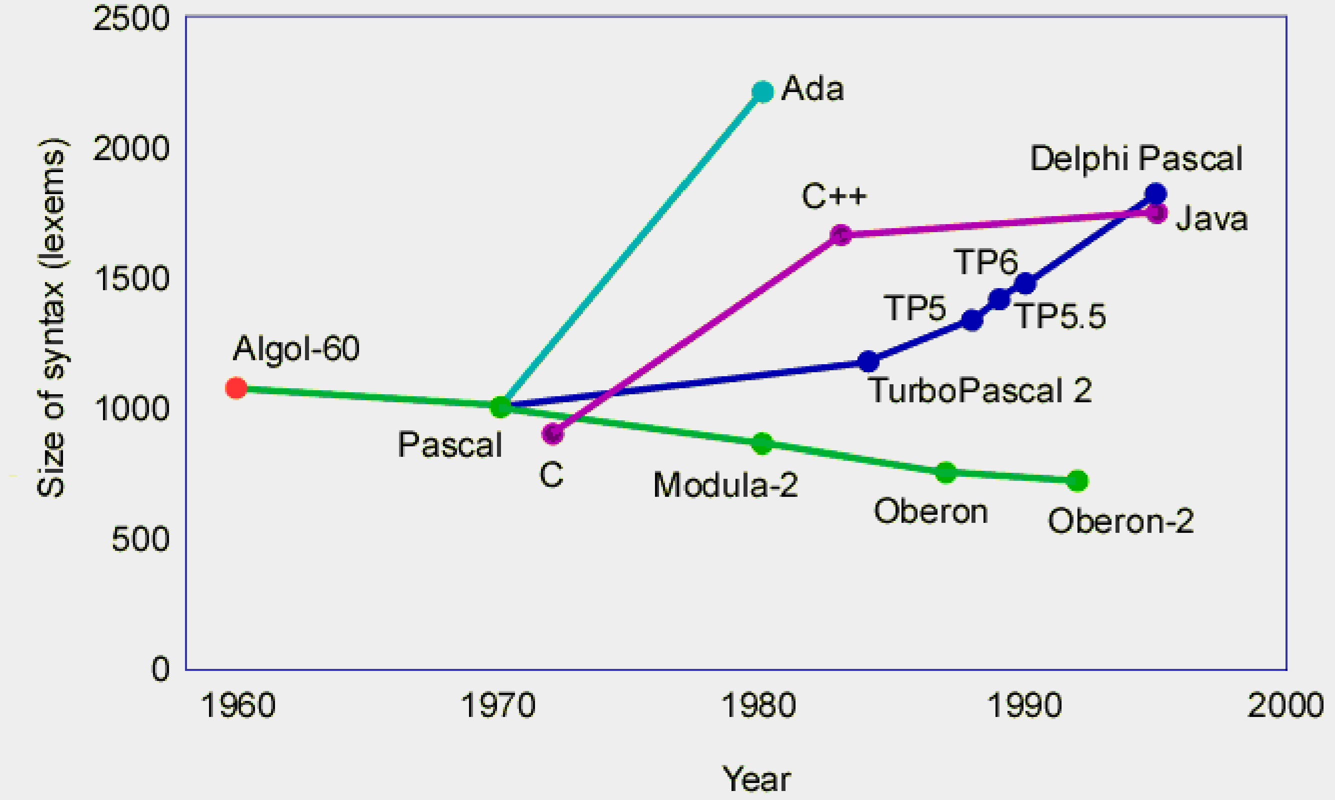


Figure 1. Complexity of syntax of programming languages

S.Z.Sverdlov (University of Vologda, Russia)

# **Part 2**: The benefits of simplicity, or the curse of complexity

- Economy of design
- Simpler to define and document
- Easier to learn and understand
- Less difficult to implement, more efficient compilation
- Fewer misunderstandings, more efficient programs
- Disciplined programming, fewer mistakes

# Conclusions and questions

- All together increases efficiency of program development, program maintenance, and program execution.
- The more complex the task, the more perspicuous and reliable must be the tools
- If problem is complex, do not add further, home-made complexity through tools
- Can a simple language be powerful?
- Can flexibility be achieved without sacrificing efficiency (or vice versa)?

# **Part 3**: Oberon for embedded systems

- System engineers want to have close control over program and code
- No hidden mechanisms tolerated
- Oberon compiler generates “straight” code
- Predictable behavior, no surprises
- Ideal for light-weight systems with or without underlying “operating system”
- Modules with separate compilation

- Oberon allows to program device drivers through its “low-level features”, which are encapsulated within specific modules.
- Directs access to device interface registers.
- No overhead through crossing of module boundaries.
- Watertight type checking, also across module boundaries, at compile time! Very fast loading and linking.
- Fast, dynamic loading of modules upon demand at run-time.
- Compiler was designed/ported for Strong-ARM within a month.

# Simplicity and Complexity

Самый верный признак истины -- это простота и ясность. Ложь всегда бывает сложна, вычурна и многословна.

Лев Николаевич Толстой

The most reliable sign of truth is simplicity and clarity. Lie is invariably complicated, gaudy and verbose.