50 Years of Pascal

Niklaus Wirth

The programming language Pascal has won world-wide recognition. In celebration of its 50th birthday I shall remark briefly today about its origin, spread, and further development.

Background

In the early 1960's, the languages Fortran (John Backus, IBM) for scientific, and Cobol (Jean Sammet, IBM and DoD) for commercial applications dominated. Programs were written on paper, then punched on cards, and one waited for a day for the results. Programming languages were recognized as essential aids and accelerators of the hard process of programming.

In 1960, an international committee published the language Algol 60. It was the first time that a language was defined by concisely formulated constructs and by a precise, formal syntax. Already two years later it was recognized that a few corrections and improvements were needed. Mainly, however, the range of applications should be widened, because Algol 60 was intended for scientific calculations (numerical mathematics) only. Under the auspices of IFIP a Working Group (WG 2.1) was established to tackle this project.

The group consisted of about 40 members with almost the same number of opinions and views about what a successor of Algol should look like. There ensued many discussions, and on occasions the debates ended even violently. Early in 1964 I became a member, and soon was requested to prepare a concrete proposal. Two factions had developed in the committee. One of them aimed at a second, after Algol 60, milestone, a language with radically new, untested concepts and pervasive flexibility. The other faction remained more modest and focused on realistic improvements of known concepts. After all, time was pressing: PL/1 of IBM was about to appear. However, my proposal, although technically realistic, succumbed to the small majority that favored a milestone.

Simply postulating a language and defining it on paper would not suffice. A solid compiler also had to be built, which usually was a highly complex program. In this respect, large industrial firms had an advantage over our Working Group, which had to rely on enthusiasts at universities. I left the Group in 1966 and devoted myself together with a few doctoral students at Stanford University to the construction of a compiler for my proposal. The result was the language Algol W, which after 1967 came into use at many locations on large IBM computers. It became quite successful. The milestone Algol 68 did appear and then sank quickly into obscurity under its own weight, although a few of its concepts did survive into subsequent languages.

But in my opinion Algol W was not perfectly satisfactory. It still contained too many compromises, having emerged from a committee. After my return to Switzerland, I designed a language after my own preferences: Pascal. Together with a few assistants, we wrote a user manual and constructed a compiler. In the course of it, we made a dire experience. We intended to describe the compiler in Pascal itself, then translate it manually to Fortran, and finally compile the former with the latter. This resulted in a great failure, because we found it impossible to translate a program written in a structured language into an unstructured language. After this unfortunate, expensive lesson, a second try succeeded, where in place of Fortran the local language Scallop (M. Engeli) was used.

Pascal

Like its precursor [*predecessor* means *died earlier*] Algol 60, Pascal featured a precise definition and a few, perspicuous basic elements. Its structure, the syntax, was formally defined in EBNF. Statements described assignments of values to variables, and conditional and repeated execution. Moreover, there were procedures. A significant extension were data types and structures: Arrays, records, files (sequences), and pointers. Its elementary data types were integers and real numbers, Boolean values and enumerations (of constants). Procedures featured two kinds of parameters, value- and variable-parameters. Procedures could be used recursively. Most essential was the pervasive concepts of data types: Every constant, variable, or function was of a fixed, static type. Thereby programs obtained much redundancy which a compiler had to use for checking type consistency. This contributed to the detection of error, and this *before* the program’s execution.

Pascal was easy to teach, and it covered a wide spectrum of applications, which was a significant advantage over Algol, Fortran, and Cobol. The Pascal System was efficient, compact, and easy to use. The language was strongly influenced by the new discipline of structured programming, advocated primarily by E.W. Dijkstra to fight the threatening software crisis (1968).

Already in 1970 Pascal was published and for the first time used in large courses at ETH Zurich on a grand scale. We had even defined a subset Pascal-S and built a smaller compiler, in order to save computing time and memory space on our large CDC computer, and to reduce the turn-around time for students. Back then, computing time and memory space were still scarce.

Pascal's spread and distribution

Soon Pascal became noticed at several universities, and interest rose for its use in classes. We received requests for possible help in implementing compilers for other large computers. It was my idea to postulate a hypothetical computer, which would be simple to realize on various other main frames, and for which we would build a Pascal compiler at ETH. The hypothetical computer would be quickly implementable with relatively little effort using readily available tools (assemblers). Thus emerged the architecture Pascal-P (P for portable), and this technique proved to be extremely successful. The first clients came from Belfast (Prof. Hoare). Two assistants brought two heavy cartons of punched cards to Zurich. At the border, they were inspected with scrutiny, for there was the suspicion that the holes might contain secrets subject to custom fees. - All this occurred without international project organizations, without bureaucracy and research budgets. It would be impossible today.

An interesting consequence of these developments was the emergence of user groups, mostly of young enthusiasts who wanted to promote and distribute Pascal. Their core resided under Andy Mickel in Minneapolis, where they regularly published a *Pascal Newsletter.* This movement contributed significantly to the rapid spread of Pascal.

Several years went by until in 1975 the first micro-computers appeared on the market. These are small computers with a processor integrated on a single chip and with 8-bit data paths, affordable by private persons. It was recognized that Pascal was suitable for these processors, due to its compact compiler which would fit into the small memory (64K). A group under Ken Bowles at the University of San Diego, and Philippe Kahn at Borland Inc. in Santa Cruz surrounded our compiler with a simple operating system, a text editor, and routines for error discovery and diagnostics. They sold this package for $50 on floppy disks (Turbo Pascal). Thereby Pascal spread immediately, particularly in schools, and it became the entry point for many to programming and computer science. Our Pascal Manual became a best seller.

This spreading did not remain restricted to America and Europe. Russia and China welcomed Pascal with enthusiasm. This I became aware of only later, during my first travels to China (1982) and Russia (1990), when I was presented with a copy of our Manual written in (for me) illegible characters and symbols.

Pascal's successors

But time did not stand still. Rapidly computers became faster, and therefore demands on applications grew, as well as those on programmers. No longer were programs developed by single persons. Now they were built by teams. Constructs had to be offered by languages that supported teamwork. Single persons were to design parts of systems called modules, and to do this relatively independently. Modules would later be linked and loaded automatically. Already Fortran had offered this facility, but now a linker would have to verify the consistency of data types also across module boundaries. This was not a simple matter!

Modules with type consistency checking across boundaries were indeed the primary extension of Pascal's first successor Modula-2 (for modular language, 1979). It evolved from Pascal, but also from Mesa, a language developed at Xerox PARC for system-programming, which itself originated from Pascal. Mesa, however, had grown too wildly and needed "taming". Modula-2 also included elements for system-programming, which admitted constructs that depended on specific properties of a computer, as they were necessary for interfaces to peripheral devices or networks. This entailed sacrificing the essence of higher languages, namely machine-independent programming. Fortunately, however, such parts could now be localized in specific "low-level" modules, and thereby be properly isolated.

Apart from this, Modula contained constructs for programming concurrent processes (or quasi-parallel threads). “Parallel programming” was the dominant theme of the 1970s. Overall, Modula-2 grew rather complex and became too complicated for my taste, and for teaching programming. An improvement and simplification appeared desirable.

From such deliberations emerged the language Oberon, again after a sabbatical at Xerox PARC. No longer were main frame computers in use, but powerful workstations with high-resolution displays and interactive usage. For this purpose, the language and interactive operating system Cedar had been developed at PARC. Once again, a drastic simplification and consolidation seemed desirable. So, an operating system, a compiler, and a text editor were programmed at ETH for Oberon. This was achieved by only two programmers, Wirth and Gutknecht, in their spare time over 6 months. Oberon was published in 1988. The language was influenced by the new discipline of object-oriented programming. However, no new features were introduced except type extension. Thereby for the first time a language was created that was not more complex, but rather simpler, yet even more powerful than its ancestor. A highly desirable goal had finally been reached.

Even today Oberon is successfully in use in many places. A breakthrough like Pascal’s, however, did not occur. Complex, commercial systems are to widely used and entrenched. But it can be claimed that many of those languages, like Java (Sun Microsystems) and C# (Microsoft) have been strongly influenced by Oberon or Pascal.

Around 1995 electronic components dynamically reprogrammable at the gate level appeared on the market. These field programmable gate arrays (FPGA) can be configured into almost any digital circuit. The difference between hardware and software became increasingly diffuse. In 1996 (and 2017) I developed the language Lola (logic language) with similar elements and the same structure as Oberon for describing digital circuits. Such hardware description languages (HDL) replace circuit diagrams as used in earlier times by formal texts. This facilitates the common design of hardware and software, which has become increasingly important in practice.