Highly Structured Language

Descendant of Algol 60 and Algol 68, Algol W

Used primarily for Educational Purposes in the 1970’s and 80’s.

Block Structured

Procedures and Functions

Records

Pointers

Recursion

Not Suited for Object Oriented Programming

Strongly Typed; Early Binding

Highly Orthogonal

The goal was to design a language that could be compiled in one pass, and recursive descent parsing algorithm was to develop the initial compiler. … This compiler executed on a Control Data computer and developed the now-famous “P-Code” interpreter.

Pascal compiles by first translating Pascal source program into a hypothetical machine language for a machine that was based upon a stack architecture. A relatively efficient P-code interpreter would execute the P-code. This structure made Pascal programs relatively easy to port to other computer systems. Today Pascal P-code compilers still exist, but most compilers are written to produce native machine for efficiency reasons.

Since the original compilers were written in Pascal, getting started was a problem. The process called “bootstrapping” was used to compile the compiler by hand into its P-code. This was tedious but not difficult. Once the Pascal compiler was running optimizations in the generated code could be made to improve the process. Moving to a new machine was not difficult since the Pascal compiler on the old machine was
available to develop P-code. The final step of redoing the P-code interpreter was not difficult.

Pascal was developed in 1970. An American standard was developed in 1983 (IEEE 770 / ANSI X3.97 [Ledyard, 1984] and ISO standard soon followed (ISO 7185).

**Brief Overview**

Pascal’s run-time structure is very much like C; it also has the ability to declare internal local procedures and create a full nested name hierarchy of object. Pascal programs are formed by having a single main program block which contains definitions of the subprograms used. Each Block has a characteristic structure: a header giving the specification of parameters and results, followed by constant definitions, type definitions, local variable declarations, other (nested) subprogram definitions, and the statements that make up the executable part of the program, in that order.

Pascal’s large set of primitive and structured data types includes: integers, reals, characters, enumerations, Booleans, arrays, records, sequential files, and limited form of sets. New types can be enumerated by the programmer using a type statement although there is no encapsulation or grouping for operating on new data objects of the new type. A pointer type allows the programmer to create new data objects of any type during program execution.

Subprograms are functions (for single values) or procedures if they act by modifying their parameters or global variables. All parameters must be typed and there must be complete agreement of types between formal and actual parameters.

Structured control statements include compound statements, conditional and case statements, and three forms of iteration statement (For Do, While Do, Repeat Until). A goto statement exists but its use is highly discouraged. Subprograms are invoked with the usual call-return structure, with recursion.

Pascal is a highly structured language. Scope rules are standard in this regard and consistent with nested program formats characteristic of block structures. Parameters may be transmitted either by value or by reference. Subprograms may be passed as parameters.

During execution of a Pascal program a central stack is used for subprogram activation records, with a heap storage area for data objects created directly for use with pointer variables, and a state area for subprogram code segments and run-time support routines. Few run-time support routines are required beyond the standard procedures for input-output to sequential files and procedures for storage management.