The very first programming languages such as FORTRAN, LISP and COBOL were designed to map directly to the basic machine operations while providing a human intelligible description of what was going on and relieving the programmer of the burden of much of the incidental detail. The next step up in abstraction was ALGOL, which learnt from the lessons of FORTRAN. It was the first language to have a formal definition of its syntax and it was the first language to provide a block structure.

The predecessor of ALGOL 68 ALGOL 60 was widely used in the 1960s and as well as in the 1970s. However programming languages continued to develop. In 1966, the ALGOL committee of the International Federation for Information Processing including Niklaus Wirth, Gerhard Seegmueller, Aad van Wijngaarden, Tony Hoare, and Peter Naur started work on a revision of ALGOL, which turned into a radical redesign resulting in ALGOL 68. The first standard was published in 1968 as the name suggests, but a revised standard was published in 1973. It is the Revised Standard of 1973 that is now normally meant by "ALGOL 68". Translation was made into Russian, German, French, and into Bulgarian.

The language has imperative computational model. It is an exceptionally powerful general purpose programming language. Its design promotes clarity of thought and robust programming. It was revolutionary when it was introduced.

The support that the language provides for complex arithmetic, functions as data objects, and array slices makes it perfect for scientific and engineering purposes.

ALGOL 68 has been designed in such a way that most syntactic errors can be easily detected before they lead to breaks. Hence, opportunities for making such errors are greatly reduced.

In contrast to many languages, the semantics of Algol 68 is orthogonal to its syntax. We can change the modes (data types) of units and the program will still be meaningful.
This means that throughout the development cycle changes to one part of the source require fewer changes in other parts of the source.

One of the great strengths of Algol 68 is its mode system. The basic rule is that everything delivers a value and the set of possible values that a clause or unit can deliver is defined by a \textit{mode}. Mode is like a "type" in other languages, and there is a mode \texttt{void} to describe those cases where no tangible value is delivered. Algol 68 is very strict about its typing, and there is no mechanism to subvert the type security (as there is, by contrast, in C). This leads to more reliable and robust programs.

As well as a standard set of modes such as integers and reals, it provides the means to build infinitely more new modes allowing arbitrarily complicated data-structures to be simply represented. The pointers supported using a \texttt{REF} mode. The language has good support for rows. Rows are like arrays in other languages except that Algol 68 does not confuse the vector of values with the storage space it occupies. The bounds of a row are a property of the value, not the mode. That is they are in principle tracked at run-time, and one is never constrained to use an array of compile-time known size in Algol68, not even inside structures.

The notions of "function" and "subroutine" are unified with the notion of "procedure". A \textit{procedure} is a value in its own right, e.g., one can have arrays of procedures. The mode of a procedure specifies both the parameters (if any) and the mode of the result (or \texttt{VOID} if none).

The language has some usefull features such as array \texttt{slice} operation that delivers a subarray or single element or \texttt{skip} that delivers an undefined value of a required mode. Transfer of control is supported using the \texttt{GO} statement.

The language comes with a fairly good standard library of routines. However, there are few, if any, commercial libraries written in Algol 68.

Interfacing which is how easy it is to interface components, such as window managers or databases and cross-language linking are poor.

Perhaps the longest lasting contribution of Algol 68 is the influence it has had on the designers of the popular languages of today such as C++.

\begin{quote}Example of code:\end{quote}
"ask" takes a string which it prints as a question and
gets a "yes" or "no" answer, prompting the user if anything else
is typed and delivering TRUE if "yes" was typed.
Note the use of LC to force the response to lower-case.

PROC ask = (STRING question) BOOL:
BEGIN
    print ((question, " ?", newline));

    STRING s;

    WHILE
        read ((s, newline)); s:= LC s;
        s /= "yes" AND s /= "no"
    DO
        print ("Please answer ""yes"" or ""no"" !", newline)
    OD;

    s = "yes"
END;