ALGOL

[ALGORITHMIC LANGUAGE]

Submitted By:
Jaideep Jha
Roll No. 17

Nikhil Kumar
Roll No. 19
HISTORY
ALGOL (ALGOrithmic Language) is one of several high level languages designed specifically for programming scientific computations. It started out in the late 1950's, first formalized in a report titled ALGOL 58, and then progressed through reports ALGOL 60, and ALGOL 68. It was designed by an international committee to be a universal language. Their original conference, which took place in Zurich, was one of the first formal attempts to address the issue of software portability. ALGOL’s machine independence permitted the designers to be more creative, but it made implementation much more difficult. Although ALGOL never reached the level of commercial popularity of FORTRAN and COBOL, it is considered the most important language of its era in terms of its influence on later language development. ALGOL’s lexical and syntactic structures became so popular that virtually all languages designed since have been referred to as "ALGOL - like"; that is they have been hierarchical in structure with nesting of both environments and control structures.

ALGOL 68 was defined using a two-level grammar formalism invented by Adriaan van Wijngaarden to generate an infinite set of productions that will recognize a particular ALGOL 68 program; notably, they are able to express the kind of requirements that in many other programming language standards are labeled "semantics" and have to be expressed in ambiguity-prone natural language prose, and then implemented in compilers as ad hoc code attached to the formal language parser.

FEATURES
ALGOL was the first second-generation programming language. First consider the data structures, which are very close to first generation structures. In ALGOL 60 the block structure was introduced: the ability to create blocks of statements for the scope of variables and the extent of influence of control statements. Along with that, two different means of passing parameters to subprograms; call by value and call by name. Structured control statements: if - then - else and the uses of a general condition for iteration control were also features, as was the concept of recursion: the ability of a procedure to call itself.

ALGOL 60 as officially defined had no I/O facilities; implementations necessarily had to add some, but they varied from one implementation to another. In contrast,
ALGOL 68 offered an extensive library of *transput* (ALGOL 68 parlance for Input/Output) facilities.

ALGOL 60 allowed for two types of parameter passing: the common call-by-value, and the unique call-by-name, which has never again been adopted by any of its successor languages. Call-by-name had certain limitations in contrast to call-by-reference, making it an undesirable feature in language design. For example, it is impossible in ALGOL 60 to develop a procedure that will swap the values of two parameters if the actual parameters that are passed in are an integer variable and an array that is indexed by that same integer variable. However, call-by-name is still beloved of ALGOL implementers for the interesting "thunks" that are used to implement it.

One of the greatest impacts ALGOL 60 had was a result of its. A major contribution of this report was the introduction of BNF notation for defining the syntax of the language. Overall, ALGOL is considered to be perhaps the most orthogonal programming language, meaning it has a relatively small number of basic constructs and a set of rules for combining those constructs. Every construct has a type associated with it and there are no restrictions on those types. In addition, most constructs produce values. Several of ALGOL’s other characteristics are listed below:

**Dynamic Arrays** - one for which the subscript range is specified by variables so that the size of the array is set at the time storage is allocated.

**Reserved Words** - the symbols used for keywords are not allowed to be used as identifiers by the programmer.

**User defined data types** - allow the user to design data abstractions that fit particular problems very closely.

**AREAS OF APPLICATION**

ALGOL was used mostly by research computer scientists in the United States and in Europe. Its use in commercial applications was hindered by the absence of standard input/output facilities in its description and the lack of interest in the language by large computer vendors. ALGOL 60 did however become the standard for the publication of algorithms and had a profound effect on future language development.
SAMPLE CODE

Description
This program computes the mean (average) of the absolute value of an array. Block structures, a dynamic array, and iterative statements are featured in this program. The bold type print represents keywords.

Code

begin
integer N;
Read Int(N);

begin
real array Data[1:N];
real sum, avg;
integer i;
sum:=0;

for i:=1 step 1 until N do
begin real val;
Read Real(val);
Data[i]:=if val<0 then -val else val
end;

for i:=1 step 1 until N do
sum:=sum + Data[i];
avg:=sum/N;
Print Real(avg)
end
end

ALGOL 68 Specifications
ALGOL 68 was complex, and posed difficulties for both implementers and users. But it was also the first truly universal language. ALGOL 68 is designed to communicate algorithms, to execute them efficiently on a variety of different computers. It is a general purpose language, most famous for its "orthogonal design"; to quote from the language's defining document "The primitive concepts have been applied 'orthogonally' in order to maximize the expressive power of the language while trying to avoid deleterious superfluities." Perhaps the longest lasting contribution of ALGOL
68 is the influence it has had on the designers of the popular languages of today (e.g. C++).

**Features**

- Structural equivalence. Automatic type conversion, including de-referencing
- Flexible arrays
- Generalized loops (for-from-by-to-while-do-od), if-then-elif-fi
- Integer case statement with 'out' clause, skip statement, goto
- Blocks, procedures and user-defined operators
- Procedure parameters
- Concurrent execution (cobegin/coend) and semaphores
- Generators heap and loc for dynamic allocation
- No abstract data types, no separate compilation.

**Insecurities**

There is remarkably little insecurity in ALGOL. The language is mainly criticized, in this area, for its numerous and complex "coercions". These are value conversions from one type ("mode" in ALGOL 68) to another-dependent upon the context-that happen automatically. For example, a simple value, such as an integer, may be automatically converted into an array of integers of length one, masking a possible programmer error. Like C and C++ it is an expression-based language, i.e. most constructs return a value even in contexts where that value is immediately discarded. This "voiding" (as it is called) may be intentional or a blunder by the programmer. If a procedure P of one argument, i.e. P(x), is called with zero arguments the resulting value is the value of the procedure itself which is silently discarded and, of course, no error is reported because the language rules have not been violated.

**REFERENCES:**

1- Concepts of Programming Languages, Robert W. Sebesta
2- http://en.wikipedia.org/wiki/ALGOL_68
3- http://www.cap-lore.com/Software/Algol68.html