Algol

The *DTSS (GE-235)* Version

Background

Dartmouth DTSS Algol is based on the Algol-60 Report, edited by Peter Naur, and published in the *Communications of the ACM*, January 1960, (hence the name, Algol-60.) A group of students (Steve Garland, Tony Knapp, Bob Hargraves, and Jorge Llacer) built a compiler for Algol on the LGP-30 around 1961 or so. We called this "Algol-30" as it was not a complete implementation of Algol-60 and was for the LGP-30 computer. Later two of the students (Steve Garland and Tony Knapp) built a load-and-go subset of Algol-60 which we called SCALP, for "Self-Contained ALgol Processor."

After DTSS came into being in 1964, Steve Garland returned from graduate study during that summer to build an Algol-60 system for the first DTSS. Kevin O'Gorman and Sarr Blumsen followed Steve as maintainers and extenders of the Algol system.

Brief Description

Algol is no longer widely used. However, it should not be unfamiliar to anyone who has programmed in Pascal. Indeed, Algol was a direct antecedent of Pascal.

Dartmouth Algol required line numbers, as that was the only way at the time that one could edit programs. (This was long before WYSIWYG displays and text editors.) But unlike BASIC, these line numbers did not serve as GOTO targets.

Like BASIC, spaces were ignored in Algol. But unlike BASIC, line-endings were also ignored. That is, a single statement could extend over several typed lines. And there could be several statements on a single line.

While DTSS allowed only uppercase letters, this description follows the practice in the Algol-60 Report of using primarily lowercase letters.

Variables

Variables can have any number of letters or digits, up to 30 or so, as long as the first character is a letter. This follows the practice of most modern computer languages.

All variables must be declared. Three types are allowed: **real**, **integer**, and **Boolean**. This anticipated similar types in Pascal and similar languages. Subscripted variables also bear the type **array**. Thus, an integer array might be declared as

```
integer array game[0:10, -1:n];
```

Note that the range of the second subscript in this example may not be known until the time of execution of the program.

Statements

The most fundamental construct in Algol is the statement. Several statements can be grouped using a **begin** - **end** pair, the result considered as a single statement. For example,

if you have these three statements:

statement1
statement2
statement3

then the combination

begin statement1; statement2; statement3 **end**

is considered to be a single statement. Notice that the semicolon serves to separate, or delimit, statements from one another.

Constructs

The assignment statement (LET statement in BASIC) looks like this:

sum := x + y;

The ":=" is supposed to suggest the left-pointing arrow "<--" to indicate the direction of the assignment.

Other common constructs include the **for** statement, and the if-then or **conditional** statement. The following are typical, but by no means exhaustive.

for <variable> := <initial value> **step** <step value> **until** <final value> **do** <statement>;

if <cond1> **then** <s1> **else if** <cond2> **then** <s2> **else** <s3>;

where cond1 and cond2 are conditionals, and s1, s2, and s3 are statements.

Arithmetic Expressions

As with BASIC, Algol allows the four standard arithmetic operations and raising-to-the-power, the symbol of which is "^".

Normal precedence rules are used: Exponentiation, multiply and divide, add and subtract. Left-association is used for multiple operations within a group. That is

(a - b - c) is understood to be the same as ((a - b) - c)

All real arithmetic was done in floating point. In the GE-225 and GE-235, this meant a precision of about 30 bits (roughly ten digits) with an base 2 exponent range of -256 to +255.

Functions

Nine numeric functions were defined in the Algol-60 Report.

abs(E)	The absolute value of the expression E.
sign(E)	The sign of E (0 if E = 0).
sqrt(E)	The square root of E..
sin(E)	The sine of E.
cos(E)	The cosine of E.
arctan(E)	The arctangent of E.

ln(E)	The natural logarithm of E.
exp(E)	The exponential of E.
entier(E)	The integer part of E.

The expression E can be **real** or **integer**, except for **entier**, where it is assumed to be **real**. All results are of type real except for **sign** and **entier**, where the results are of type **integer**. Arguments for the **sin** and **cos** functions are assumed to be in radians, as is the result of the **arctan** function.

In addition, Dartmouth Algol includes the **RANDOM** function, whose value is a "random" number in the range 0 <= **RANDOM** < 1.

INPUT and PRINT Functions

Input in Dartmouth Algol is provided by the **READATA** procedure. Thus

READATA (DATABLOCK, X, Y, Z);

will read three values into X, Y, and Z from the collection of data block whose definition might be

DATA DATABLOCK := 1.2, 3.5, 8.7, -4.5;

Input from the Teletype can be done with

READATA (TELETYPE, MYGUESS);

Printing is done with the **PRINT** procedure. Thus

PRINT ("ANSWER IS", X, Y);

acts just as you would expect, printing the string "ANSWER IS" into zone one, the value of X into zone 2, and so on.

The Algol-60 Report simply ignores input and output. The procedures **READATA** and **PRINT**, and the keyword **DATA** were additions made to Dartmouth Algol.

Procedures and Defined Functions

The user may define any number of procedures. In the original BASIC the only subroutine was the GOSUB and RETURN construct, and the only new functions had to be named FNA to FNZ. In Algol, the concept **procedure** embraces both subroutines and functions. That is, **functions** are simply **procedures** that are typed (i.e., **real**, **integer**, or **Boolean**.) This idea was carried over almost without change into Pascal.

As in BASIC, the value of a **function procedure** is provided by an assignment to a "variable" which is the name of the function procedure.

Example Programs

These two programs are equivalent to the two BASIC programs included in the brief outline of BASIC.

```
100 BEGIN COMMENT PLOT A NORMAL DISTRIBUTION CURVE;
110
```

```
120 REAL PROCEDURE NORMAL(X); REAL X;
122    BEGIN NORMAL := EXP(-(X^2/2))/SQRT(2*3.14159265) END;
130
140 PROCEDURE PLOTIT (T); INTEGER T;
142    BEGIN INTEGER I, J, K;
143       J := ENTIER(T/3); K := T - 3*J;
145       FOR I := 1 STEP 1 UNTIL J DO PRINT ("", "   ", "");
150       IF K = 0 THEN PRINT ("", "*")
155          ELSE IF K = 1 THEN PRINT ("", " *")
160          ELSE PRINT ("", "  *");
170    END PLOTIT PROCEDURE;
180
190 REAL X, Y; INTEGER T;
200
210 FOR X := -2 STEP .1 UNTIL 2 DO
220    BEGIN Y := NORMAL(X); T := ENTIER(100*Y);
225       PLOTIT(T);
230    END LOOP;
235
240 END PROGRAM;
```

The Algol program is a bit more complicated than the BASIC program as all string prints produce a multiple of three characters..

```
100 BEGIN INTEGER X, G, N, A;
110    PRINT ("GUESS THE NUMBER BETWEEN 1 AND 100.");
120 START: X := ENTIER(RANDOM*100 + 1); N := 0;
130 LOOP: READATA (TELETYPE, G); N := N+1;
140       IF G < X THEN PRINT ("TOO SMALL, GUESS AGAIN")
150       ELSE IF G > X THEN PRINT ("TOO LARGE, GUESS AGAIN")
160       ELSE GOTO DONE;
170       GOTO LOOP;
180 DONE: PRINT ("YOU GUESSED IT, IN ", "", N, "", " TRIES");
185       PRINT ("ANOTHER GAME (YES = 1, NO = O)", "");
190       READATA (TELETYPE, A);
200       IF A = 1 THEN GOTO START
210 END
```

Commands

The commands of the operating system apply to Algol as well as to BASIC.

HELLO	Start a new session, enter your user number
NEW	Start a new program
OLD	Retrieve a program from storage
SAVE	Save the current program to storage
REPLACE	Save the current program to storage, overwriting older version
RENAME	Rename the current program
CAT	List the names of your saved programs (short for CATALOG)
LIST	List the current program
RUN	Run the current program
STOP	Stop the current run of the program (in case an infinite loop)
UNSAVE	Unsave the current program program

SYSTEM	Name the system -- limited to either BASIC (default) or ALGOL
BYE	End the session
GOODBYE	Same as BYE

All commands may be abbreviated to the first three letters.

The NEW, OLD, and RENAME commands may be followed by a program name. If not, the operating system will ask you for the name of the program. The SYSTEM command may be followed by either BASIC or ALGOL. If not, the operating system will ask you for a system name. (Like the commands, the system names may be abbreviated to three letters.)

In addition, the SPEED command allows you to specify the teletype speed, for a more realistic simulation. Thus, SPEED 10 will slow things down to about 10 characters per second.

See the separate document *Commands*.

Thomas E. Kurtz
2004 May 28