THE ELLIOTT ALGOL SYSTEM.

(For the original Elliott Algol 60 documentation visit: https://tinyurl.com/Elliott-900-Algol60. Note that not all the features documented are provided in the web-based emulation, in particular there is no support for working with other than the standard built-in library. If you wish to explore these facilities download my Windows emulator from here: https://github.com/andrewjherbert/Elliott-900-simulator.

What kind of ALGOL is Elliott ALGOL? Well it comes with the Elliott I/O system, 18-bit integers and reals with 8 significant decimal places. It was written by CAP and Elliotts in 1966/67 as an IFIP subset ALGOL for an 8K machine and was derived loosely from KDF9 Whetstone ALGOL. Norman Spink at Elliotts subsequently turned it into a Load-and-Go system for a 16K machine.

On an Elliott 903, translation speed was 100 characters per second and programs ran at 60 statements per second on the Whetstone benchmark, one third as fast as its KDF9 parent.

The main restrictions imposed by Elliott ALGOL compared to the formal definition of ALGOL are listed later in this manual.

Character set.

Depending on the version, Elliott ALGOL can use the 920, 903 or 900 Elliott telecodes. Each character set includes upper and lower case letters, but they are regarded as the same. On output all letters are in upper case.

In addition to letters and digits, the characters # $ % & ' ( ) * + - . / : ; < = > ? @ [\ ] ^ _ are permitted in strings.

In Elliott telecode strings are quoted using ‘...’, in ASCII this is equivalent to '...@, and not '...`. The emulator also permits {....} for string quoting.

The representation of ALGOL symbols is as shown below and this allows spaces to occur in names (the stropping convention is double quotes around keywords in 900 and 903 telecode, or ~ in 920 telecode).

begin end	"BEGIN" "END" or "begin" "end"
<=	"LE"

>=	"GE"
/=	"NE"
and	"AND"
or	"OR"
not	"NOT"
equiv	"EQUIV"
impl	"IMPL"
string quotes	' and @, or { and }
subscript ten	?
*+-/()[]<=>:;.,	stand for themselves
exponentiation	^
div	"DIV"

Restrictions

903 ALGOL implements an IFIP subset of ALGOL 60.

1. Everything must be declared before it is used.

2. Recursion is not permitted.

3. Expressions are not allowed as all-by-name actual parameters.

4. Switch lists may only contain labels, not designational expressions.

5. Unsigned integers may not be used as labels.

6. The controlled variable in a for clause may not be subscripted.

7. own is not allowed.

8. All formal parameters of a procedure must be specified.

9. call of a type procedure can only occur in an expression.

10. Only the first six characters are significant in an identifier.

11. The identifiers checkr, checki, checkb and checks are reserved (see section on checking functions).

12. At most 14 parameters are allowed in a procedure call.

13. The formal parameters of an actual procedure parameter must all by "call-by-name".

14. if A+B then is indistinguishable from if A+B > 0 then

15. Comments can only contain characters that are legal in strings, plus the characters permitted as string quotes.

Miscellaneous

The largest real is about 9^18.

Standard library procedures.

The standard library for all version of Elliott ALGOL contains arctan, cos, instring, lowbound, outstring, range, sin, sqrt.

instring and outstring provide means to read and store text from a data tape. Such text has to be enclosed in string quotes.
instring(A,M) has the effect of searching for an opening string quote and then reading the text that follows up to the closing string quote that brackets the first. This text is stored in the locations A[M], A[M+1], ... three characters to each location. (A must be an integer array and M an integer variable. Before the procedure is used M must be assigned a value (normally) the lower bound of A. Following the call, M will be equal to the index of the next available element of M. Thus instring can be executed repeatedly in a way that the input strings do not overwrite each other.

Strings that have been read and stored by instring may be punched out by means of the procedure outstring(A,M). Initially M must have the value it had before the for the first instring procedure. Each time a string is punched out, M is advanced to be the start of the next string in A. Inner strings are interpreted as in print statements.

lowbound(A,M) gives the lower bound of the M-th bound pair of the array A. range(A,M) gives the range.

The following procedures are built-in to the ALGOL system: abs, aligned, digits, entier, exp, free point, ln, prefix, punch, reader, same line, scaled, sign, stop, wait.

aligned, digits, free point, prefix, punch, reader, same line and scaled are Elliott I/O procedures. If these procedures are called with a <u>print</u> statement their effect is local to that statement; if used outside of a <u>print</u> statement their effect applies to all subsequently executed <u>print</u> statements (unless overridden by other calls within those <u>print</u> statements or when updated by a subsequent call outside of a <u>print</u> statement).

prefix(S) causes every subsequent output item the be preceded with by string S; same line forces subsequent items to be printed on the same line (the default is a new line for each item).

Floating point numbers can be printed in free point, aligned or scaled form. Free point(N) prints numbers in N spaces with the decimal point where it belongs. The scaled(N) format always places the decimal point after the first digit and uses an exponent part to indicate the scale of the number. The aligned(M,N) format prints with M decimals before the point and N after. Leading zeros are replaced by spaces.

digits(N) prints integers in N spaces. Leading zeros are replaced by spaces.

reader(1) selects input from paper tape, reader(3) from teletype. punch(1) selects output to paper tape, punch(3) to teletype.

stop: the program halts.

COMPILING AND RUNNING ALGOL PROGRAMS.

The Elliott Algol system came in several versions. The most basic was a two-pass system for the smallest 8K store machines. First a translator program would be read in, then the user's source program, producing an intermediate binary tape. Then a run-time interpreter tape would be read in and fed the intermediate binary paper tape for execution, along with the user’s data.

For larger 16K machines the translator and interpreter were loaded as a single combined program. The user then had simply to load their program source code tape, enter the translator and if no translator errors were reported enter the interpreter and run the translated program. This mode of operation was convenient for running a “cafeteria” style service for student programming in colleges and is what the emulator provided here supports. (The loading of paper tapes and operator actions to enter the translator and interpreter are handled automatically by the emulator).

Diagnostic output from the interpreter and translator was always directed to the machine’s teletype. By default, program data input and output were to or from paper tape, but the programmer could override this to read data from the teletype and/or direct output to the teletype. (The emulator presently does not support teletype input).

NOTE

Elliott Algol provided additional capabilities to produce diagnostic output from the translator to help with locating run-time errors and a facility called “checking mode” to enable run-time tracing. These are presently not supported by the emulator.

CHECKING FUNCTIONS.

Elliott ALGOL provides checking functions to enable intermediate results of a calculation to be printed out if the program is translated in "checking mode". These functions are:

```
     checkr    (for real argument)
     checki    (for integer argument)
     checkb    (for Boolean argument)
```

The statement

```
     A := B/checkr(C);
```

causes the value of C to be output (preceded by a change to a new line and an asterisk), prior to the instruction A:=B/C being completed.

The argument of a checking function can be a variable or an arithmetic expression. Checking functions can be nested as in:

```
     A:=B+checkr(M[checki(j+3)])
```

which would print first the value of (j+3) and then the value of M[j+3] on the next line.

Additionally, there is a checking procedure

```
     checks(‘...’)
```

which can be used to trace the progress of a calculation.

If translation is not in checking mode, checking functions are all ignored.

REPORT MODE.

When run in report mode the translator produces a table of addresses for every procedure, function and label in the translated program. In addition to showing the size of the translated program, this information is useful for tracking down the source of run-time errors.

Here is an example output from compiling in report mode:

```
QUICKS
P EXCHAN    ADR   70
E           ADR   82
P WICHMA    ADR   84
E           ADR  187
P RANDOM    ADR  189
E           ADR  237
P PARTIT    ADR  239
L UP        ADR  265
L DOWN      ADR  288
L CHANGE    ADR  312
E           ADR  348
E           ADR  379
E           ADR  410
E           ADR  410
P QUICKS    ADR  412
```

```
P LOCAL     ADR  417
E           ADR  452
E           ADR  452
E           ADR  458
E           ADR  555
E           ADR  556
PROGRAM  585
SCALARS   17
```

This shows that the program titled QUICKS has various procedures called EXCHAN, WICHMA, RANDOM. PARTIT, QUICKS and LOCAL indicated by their names being preceded by the letter P. Labels are preceded by the letter L and "END"s are given by the Es, together with the overall length and the space occupied by all the scalars.