TECHNICAL REPORT 1672

Published 1977-02-15

ISO Technical Reports are subject to review within three years of publication, with the aim of achieving the agreements necessary for the publication of an International Standard.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION • МЕЖДУНАРОДНАЯ ОРГАНИЗАЦИЯ ПО СТАНДАРТИЗАЦИИ • ORGANISATION INTERNATIONALE DE NORMALISATION

Hardware representation of ALGOL basic symbols in the ISO 7-bit coded character set for information processing interchange

Représentation machine des symboles de base d'ALGOL dans le jeu de caractères codés ISO à 7 éléments pour l'échange d'information entre matériels de traitement de l'information

This Technical Report was drawn up by Technical Committee ISO/TC 97, *Computers and information processing.* It results from successive revisions of draft ISO Recommendation No. 1672, upon which voting by the ISO Member Bodies took place in 1969. Failure to reach the general agreement necessary for publication of the document as an International Standard led to a decision by the members of ISO/TC 97 to publish it as a Technical Report. The difficulties preventing agreement on an International Standard are discussed in detail in annex D.

In October 1976, this document was submitted to the ISO Council, which approved its publication as a Technical Report.

Possibility of a future International Standard

IFIP Working Group 2.1, which has responsibility for ALGOL, is considering a possible revision of the ALGOL 60 language. It is suggested that any further attempt to produce an International Standard on this subject should await news from IFIP of whether or not such a revision is to be made.

To be withdrawn

0 INTRODUCTION

0.1 The ALGOL 60 programming language (ISO/R 1538) contains 116 "basic symbols". It has always been recognised that the character sets available on computing equipment could not be expected to coincide with these symbols, and consequently that hardware representations were needed that would enable the ALGOL language to be used in practice.

0.2 In 1967, ISO Recommendation R 646, *6- and 7-bit coded character sets for information processing interchange,* was published, giving the possibility of an internationally agreed hardware representation in terms of the character sets described therein.

0.3 However, ISO/R 646-1967 was superseded by ISO 646-1973, which standardized only the 7-bit character set, relegating the 6-bit set to an appendix for information only. Since the representation of ALGOL in the 6-bit code had been the cause of some of the difficulties of the project, the proposal was recast in terms of the 7-bit code only (while retaining the equivalents of the 6-bit representations as alternatives where these caused no difficulties).

UDC 681.3.04 **Ref. No. ISO/TR 1672-1977 (E)**

Descriptors : data processing, information interchange, coded character sets, ISO seven-bit code, ALGOL, symbols.

Printed in Switzerland

Price based on 9 pages

1 SCOPE AND FIELD OF APPLICATION

This Technical Report suggests a relationship between the ALGOL symbols shown in ISO/R 1538-1972 (sub-clauses 2.1, 2.2 and 2.3) and the ISO 7-bit coded character set which is the subject of ISO 646-1973. The international reference version of ISO 646 is assumed, and the characters @ [ ] { and } are used.

2 REFERENCES

ISO 646, *7-bit coded character set for information processing interchange.*

ISO/R 1538, *Programming language ALGOL.*

3 REPRESENTATION

3.1 In the following table, the left-hand column shows the reference form of the ALGOL basic symbols. The right-hand column shows the equivalent symbols of the 7-bit coded character set. Some basic symbols are represented by a single character; others by a sequence of characters.

3.2 For certain ALGOL basic symbols, alternative representations are given. Any of the given representations should be recognized as valid, even if a document is not self-consistent but sometimes uses one representation and sometimes another, except that the characters [ ] and { } may be disallowed in contexts where they are inconsistent with national usage.

TABLE – Basic symbol representations

Reference form	Hardware representation	col/row
⟨letter⟩	corresponding upper case or lower case letter	
⟨digit⟩	corresponding digit	
true	'TRUE' 'true'	
false	'FALSE' 'false'	
+	+	2/11
−	−	2/13
×	★	2/10
/	/	2/15
÷	% '/'	2/5 2/7, 2/15, 2/7
↑	^ ★ ★	5/14 2/10, 2/10
<	< 'LT' 'lt'	3/12
⩽	<= 'LE' 'le'	3/12, 3/13
=	= 'EQ' 'eq'	3/13
⩾	>= 'GE' 'ge'	3/14, 3/13
>	> 'GT' 'gt'	3/14
≠	<> 'NE' 'ne'	3/12, 3/14
≡	'EQV' 'eqv'	
⊃	'IMPL' 'impl'	
∨	'OR' 'or'	
∧	'AND' 'and'	
¬	'NOT' 'not'	
go to	'GOTO' 'goto'	
if	'IF' 'if'	
then	'THEN' 'then'	
else	'ELSE' 'else'	
for	'FOR' 'for'	
do	'DO' 'do'	

Reference form	Hardware representation	col/row
,	,	2/12
.	.	2/14
10	@	4/0
:	:	3/10
;	;	3/11
:=	:=	3/10, 3/13
␣		2/0
step	'STEP' 'step'	
until	'UNTIL' 'until'	
while	'WHILE' 'while'	
comment	'COMMENT' 'comment'	
(	(	2/8
)	)	2/9
[	(/ '(/'	5/11 2/8, 2/15
]	/) '/)'	5/13 2/15, 2/9
`	{ '('	7/11 2/7, 2/8, 2/7
'	} ')'	7/13 2/7, 2/9, 2/7
begin	'BEGIN' 'begin'	
end	'END' 'end'	
own	'OWN' 'own'	
Boolean	'BOOLEAN' 'boolean' 'Boolean'	
integer	'INTEGER' 'integer'	
array	'ARRAY' 'array'	
real	'REAL' 'real'	
switch	'SWITCH' 'switch'	
procedure	'PROCEDURE' 'procedure'	
string	'STRING' 'string'	
label	'LABEL' 'label'	
value	'VALUE' 'value'	

ANNEX A

SUGGESTED RULES FOR TYPOGRAPHICAL FEATURES

According to ISO/R 1538 (sub-clause 2.3), "Typographical features such as blank space or change to a new line have no significance in the reference language". In this hardware representation, the suggested equivalent rules are as follows :

a) horizontal tab (0/9) is interpreted as a space (2/0);

b) vertical tab (0/11) or form feed (0/12) is interpreted as a line feed (0/10);

c) delete (7/15) is everywhere ignored, whether within a basic symbol or between basic symbols, except as in (i) below;

d) space (2/0) is everywhere ignored, except within a string where it represents the space symbol (␣), and except as in (i) below;

e) the effect of using backspace (0/8) is undefined;

f) the effect of using carriage return (0/13) without line feed (0/10) is undefined;

g) line feed (0/10) is ignored anywhere between basic symbols, but is not permitted within a basic symbol;

h) other characters in the 0 and 1 columns may be meaningful, so far as their control functions are concerned, but are ignored so far as their effect on ALGOL text is concerned, except as in (i) below;

i) the constituent characters of a string quote (where the multiple character versions are used) must not be separated by any other character.

ANNEX B

PROPOSAL FOR CHARACTER SUBSTRINGS

B.1 There is a difficulty in ALGOL, in that ISO/R 1538 (sub-clause 2.6.1) defines

⟨proper string⟩ ::= ⟨any sequence of basic symbols not containing 'or'⟩ | ⟨empty⟩

So that, apparently, only ALGOL basic symbols are allowed within a string. Yet for many practical purposes it is important to be able to transmit any characters available, including not only visible non-ALGOL characters, but also the invisible control characters.

B.2 The difficulty is increased in the input-output procedures of Parts II A and II B of ISO/R 1538 which deal in ALGOL basic symbols within a string, and additionally need to count the basic symbols. The string ':=' (see ISO/R 1538, 3.2.2) is then ambiguous – does it contain one basic symbol or two ?

B.3 To overcome this difficulty it is proposed that a string should be allowed to include, in addition to basic symbols, character substrings defined as follows :

⟨character list⟩ ::= ⟨col⟩/⟨row⟩ | ⟨character list⟩, ⟨col⟩/⟨row⟩

⟨col⟩ ::= ⟨unsigned integer⟩

⟨row⟩ ::= ⟨unsigned integer⟩

⟨character substring⟩ ::= "⟨character list⟩"

If col < 8 and row < 16 then the symbol indicated is that to be found in the designated column and row of ISO 646. Other characters may be indicated, by prior agreement within a given context, by col or row values exceeding the above limits.

B.4 In this proposal, characters within a string are to be considered as compound if possible, working from left to right. Line feed may not occur within a compound character (see annex A, rule (g)), but may occur and is ignored between compound characters or within a character substring.

B.5 The proper string of {:=} would then be, unambiguously, one basic symbol, not two. A string containing a colon followed by an equals sign would be expressed either as {"3/10,3/13"}, or as $\left\{\begin{matrix}:\\=\end{matrix}\right\}$ (where a new line immediately follows the colon, without intervening spaces).

Within a character substring, each ⟨col⟩/⟨row⟩ counts as if it were one basic symbol for purposes of assigning integers to the basic symbols of a string. The character " and the commas within a character list do not count at all. Thus the string

{B, "3/15,0/12" 'TRUE'}

contains five symbols numbered as follows :

1 B

2 ,

3 ?

4 form feed

5 **true**

Characters derived from a character list are never to be regarded as compound and never to be regarded as having any meaning; thus {"2/7,2/15,2/7"} means "/" which must not be taken to mean '÷'. {"7/13"} means '}' and the character does not close the string.

B.6 EXAMPLES

a) {***} means '↑x', not 'x↑' or 'xxx'

b) {<>=} means '≠=' not '<>'

ANNEX C

REPRESENTATION OF BASIC SYMBOL WORDS

Some implementations of ALGOL 60 do not use any special indication for basic symbol words, but use representations in which they are indistinguishable from identifiers. This usage may be adopted, by prior agreement within a given context, provided that :

a) no basic symbol word may be used as an identifier;

b) the characters horizontal tab (0/9), space (2/0), vertical tab (0/11), form feed (0/12), line feed (0/10) may not be used within an identifier or within any basic symbol (in particular, GOTO may not be written GO TO in this representation);

c) if a basic symbol word is preceded, or followed, by a letter, a digit or a basic symbol word, then one of the characters listed in (b) above must be used as a separator.

ANNEX D

DIFFICULTIES WHICH HAVE PREVENTED AGREEMENT ON AN INTERNATIONAL STANDARD

The difficulties arising in this project have largely been historical. If ALGOL 60 were now being devised for the first time, and the ISO 7-bit character set were available for a hardware representation, there would probably be little difficulty in achieving a compromise that could be accepted by various viewpoints; but in fact many different implementations were available, each with its own hardware representation, well before the ISO character sets appeared.

Thus, attitudes have become entrenched, many of those involved feeling, quite sincerely, that the particular tradition in which they have come to know ALGOL is the natural way of doing things and that the other traditions are a little perverse.

These difficulties of tradition would be hard to resolve even if nearly all hardware equipment used the ISO character set. The situation is further bedevilled, however, by the existence of large quantities of equipment using other character sets. Here, again, people seem to be deeply entrenched behind the character sets that they know and use.

If an International Standard refuses to compromise with these other character sets, is it likely to achieve nothing, by failing to notice reality? If it does compromise with them, what becomes of international standardization? Would there be any point in producing an international standard character code if the Standard were to be regarded as of little account as soon as it was challenged ?

Furthermore, if one takes note of one alternative code, what argument can there be for refusing to take note of others? If one changes the plans to take into account the code that happens to be favoured by any particular manufacturer, is this not unfair to a manufacturer who has taken the trouble to implement the ISO code ?

Of ALGOL's 116 basic symbols, there are 44 that present no difficulty whatever. These are the 26 upper case letters A to Z, the digits 0 to 9 and the symbols ( ) , . = + − /. These map directly from one representation to the other, and are universally available.

A further 29 map directly to the ISO code, but are less universal. These consist of the 26 lower case letters a to z and the symbols

: ; :=

the last of which maps directly by using two characters for the one symbol. If only one case of letters is available, the disadvantage is slight — indeed the ALGOL subsets specifically do not expect more than 26 letters to be available altogether. The lack of the other three symbols would make ALGOL impossible. This present Technical Report does not suggest any alternatives, but it might well be advantageous to allow

.. ., .=

thus permitting standard ALGOL representation on a small subset of the ISO code.

Another 24 symbols consist of "stropped" English words, the stropping consisting of any device that will make these words clearly distinguishable from identifiers consisting of the same letters.

In the Reference Language, stropping consists of underlining the words in manuscript or typescript, but bold face is normally used instead in printed text. For computer input, underlining has sometimes been used, and can give an appearance that is pleasantly close to the Reference Language, but on much equipment it is not available.

It is available in the ISO code but only by using back-spacing, which is a notoriously unreliable feature of some equipment. It also has the disadvantage of being very "long winded" — the symbol procedure takes 27 characters (9 letters, 9 backspaces and 9 underlines) — and calling for line reconstruction if one is to be able to tell solely from the print-out what program is represented.

Underlining was therefore rejected in favour of another form of stropping that has been widely used, namely surrounding the word by apostrophes. This has an additional small advantage in that the beginning and end of each stropped word is clearly marked, whereas with underlining stropped words may be run together, leaving a compiler to sort them out.

There seems to be fairly general agreement that apostrophes form an acceptable method of stropping, if stropping is to be used at all, but some people feel strongly that this extra punctuation is unnecessary, and that basic symbol words should be distinguished from identifiers merely by context. This cannot be done without introducing additional rules, which are set out in annex C as a permitted variation, but some regard this representation as contrary to the spirit of ALGOL and would wish to see it abandoned.

Of the remaining 17 symbols, 4 have caused little argument :

$\times$ to be represented by $\star$ is almost universal in computer usage;

$\leqslant$ and $\geqslant$ would, perhaps, be best represented by $\underline{<}$ and $\underline{>}$, but having decided not to use underlining, the alternative representation of <= and >= is widely accepted;

$\supset$ to be represented by 'IMPL' has caused no comment. The symbol is, in any case, so little used that it is unlikely to cause strong feelings.

The other 13 symbols are more contentious :

$\div$ to be represented by %. ALGOL definitely requires two different division symbols. It is not possible to use / for both real division and integer division as is done by FORTRAN. % has the advantage both of the oblique nature of a division sign, and of the approximate appearance of a rotated version of $\div$, but some arguments have been made for / / instead.

$\uparrow$ to be represented by ^ or $\star\star$. The $\star\star$ notation is used by both FORTRAN and PL/I and there is general agreement that it should be available as an alternative. ^ is more contentious. ISO/R 646 gave this character as upwards arrow or circumflex accent, so it seems the obvious choice for the ALGOL upwards arrow. ISO 646, however, gives it as upwards arrow head or circumflex accent, so the choice is not quite so direct. Much of the opposition to this symbol arises from the fact that the widely-used EBCDIC code employs this position for the $\neg$ character, which is also an ALGOL symbol. To users who have the ISO/R 646 $\uparrow$ symbol, it seems absurd not to be able to use it. To users of EBCDIC it seems even more absurd to use $\neg$ to mean $\uparrow$.

$\neq$ to be represented by <>. There is a certain logic in this choice, since if >= is to mean "greater or equal" and <= is to mean "less or equal" then <> should mean "less or greater", which is logically "not equal". However, this is not something that has been widely implemented, and EBCDIC users would prefer $\neg$= as in PL/I, if $\neg$ were available.

$\equiv$ to be represented by 'EQV'. In earlier drafts 'EQUIV' was suggested and this caused little comment, but it was realised that this could cause confusion with the *equiv* function of sub-clause 1.2.3.2 of Part II B of ISO/R 1538. A vote, on which of a number of possible actions to adopt, eliminated all suggestions except 'EV' and 'EQV', which tied for first choice – 'EQV' has been chosen as more widely used at present.

$\wedge$ $\vee$ to be represented by 'AND' and 'OR'. It is agreed that these representations should be available; the question is whether symbolic alternatives should be available in addition to these stropped words. An obvious choice for $\wedge$ would be &, which is used by PL/I, but it would seem illogical to have a symbol for *and* but not for *or*. The PL/I symbol is vertical bar, but this has the disadvantages of (i) being too difficult, in manuscript, to distinguish from 1, I and l, and (ii) coinciding with one of the metalinguistic connectives of ALGOL's formal syntax. At the time of discussion, the members of Sub-committee 5 of ISO/TC 97 were unaware that the reverse solidus of ISO 646 had been introduced to allow $\wedge$ and $\vee$ to be represented by /\ and \/. This is worthy of consideration, but would appear somewhat strange in countries whose national systems had adopted alternative symbols for position 5/12.

$\neg$ to be represented by 'NOT'. Nothing but $\neg$ will satisfy the EBCDIC adherents, but this corresponds to ^ in ISO code. A possible solution would be to allow ^ to mean either $\uparrow$ or $\neg$, the meaning in each case being determined by context. There would be no ambiguity except in strings.

$_{10}$ to be represented by @. In an earlier draft, the representation E was given, as in FORTRAN. This would have been ambiguous in ALGOL since $_{10}5$ is a valid number, but E5 is a valid identifier. Stropping was suggested to make 'E', but since this symbol is part of a number, and likely to be used in data as well as in program, it was considered essential that it should be represented by a single character. At the Berlin meeting of the preparatory working group, when it was thought essential that the 6-bit character set should be included, it was decided that the apostrophe was the only single character that would do. It was shown that it could be given this use, in addition to its use for stropping, without ambiguity except within a string. It was unpleasant, however, and the consequences within a string were tortuous. At the Washington meeting of Sub-committee 5, the abandoning of 6-bit representation led to its replacement by @ with much relief.

[ ] to be represented by [ ] or (/ /). The use of [ ] is obvious where available. These positions may be overwritten, however, in national standards. ( ) will not do, as ALGOL needs to distinguish between subscripts and other bracketing. (/ /) has been quite widely implemented, but some people would prefer (: :).

‘ ’ to be represented by { } . These positions also may be overwritten in national standards, in which case the alternatives '(' ')' must be used instead. ALGOL needs to be able to distinguish between opening and closing string quotes, and suggestions such as " " will not do.

␣ to be represented by a space. This is probably the most contentious point of all.

Arguments in favour of using an actual space are :

a) it is the sensible, natural, thing to do. What could be more obvious ?

b) if you insist on a visible symbol, programmers tend to forget to include it. Nobody will ever forget to include an actual space where needed;

c) what we need is a useful hardware representation. The purism of the reference language is not required.

d) to introduce the notion that, in hardware representation, a space is significant, is only a slight irregularity. It does not change ALGOL's layout independence outside strings, and strings are such a small part of ALGOL anyway;

e) in the annex C usage, spaces have some recognition as separators in any case;

f) programmers who have used actual spaces like them, and would be sorry to have to change.

Arguments in favour of using a visible space symbol are :

a) it is difficult to find a satisfactory way of including an actual space, and of distinguishing it from a "null", in a list of characters available, or in a document of equivalents (such as the present Technical Report);

b) one of the advantages of ALGOL is that it is layout independent, and any piece of paper will serve to record a program. No special coding forms are needed. This should be true of hardware representation just as much as of reference language;

c) programmers who use actual spaces in hardware representation come to regard them as valid in reference language too, and tend to be upset at the idea that their reference language recording of a program is invalid;

d) to be sure of the output that will be obtained, it is necessary to indicate to a punch operator how many spaces are to be included and, in a program listing, to count how many have been included. These things are much easier to do with a visible space symbol than without;

e) since ALGOL is not tied to any particular width of line, the counting of the number of spaces in a string may depend upon determining the number that appear at the end of a line, between the last non-space symbol and the end of record. This is impossible to do from a listing. It should always be possible to tell whether a program is correctly recorded or not by examining the listing without having to go back to the source;

f) the ISO character set includes horizontal tab (0/9), which according to annex A, rule (a), is to be interpreted in ALGOL as a space. If spaces have no significance, even in strings, this does no harm, but if spaces are significant, then the possibility of a horizontal tab means that counting spaces, from a print-out, is not merely difficult but impossible;

g) when programs are printed in book form (as distinct from the output from computer equipment) characters usually have different widths, and printers have their own spacing conventions. One only has to look at printed FORTRAN to see some of the dreadful things that result from this fact. In desperation, authors sometimes have to adopt a visible space symbol (such as "b", since FORTRAN has only upper case letters) as the only way of making their meaning clear;

h) programmers who have used visible space symbols like them, and would be sorry to have to change.

In an earlier draft, the symbol _ (5/15) had been adopted to indicate ␣, but a postal vote has now led to its replacement by an actual space (2/0).