Naur
2.
3
4A

Meeting of the European Representatives to the ALGOL Conference

Mainz, December 14 - 16, 1959

0. General Remarks

Where information is described as optional, it must be correct if given, since translators may make use of it. In particular, if optionally an identifier appears type information concerning it must be complete.

An identifier represents one and onlyone entity.

1. Declarations governing a statement

Any declaration Δ is to hold throughout and only throughout the statement Σ or compound statement

begin Σ ; Σ ;...Σ end which immediately follows it.

Form $\underbrace{\Sigma \sim L :}_{\text{may be empty}}$ $\underbrace{\text{local } (I,I,\ldots I);}_{\text{may be empty}}$ $\underbrace{\Delta ; \Delta \ldots \Delta}_{\text{may be empty}}$ $\underbrace{\text{begin } \Sigma ; \Sigma ; \ldots \Sigma \text{ end}}_{\text{may be } \Sigma}$

For the new declaration local (I,I,...I) see below.

This means that the declarations immediately in front of the statement are no longer valid after exit from the statement, either by completion of the last statement contained in it or by a go to leading out of it.

Moreover, the values of the variables governed by the declarations cannot be expected to be available after exit from the statement, even after a second entry.

Thus, the identifiers governed by the declarations in front of the statement may be used outside it completely independently and the quantities corresponding outside are unaffected by passing through it.

The quantities governed by the above declarations are local for the statement (if no other declaration applies to an identifier, the declaration local alone is used). The significance of all other identifiers extends outside the statement (but they might be local on a higher level).

Labels and switches are local in this sense without needing a declaration to that effect.

All declarations concerning one identifier have to be given by juxtaposition of the declarators.

Example: integer array x[1:n]; integer function f(x,i):=sign(x[i])

2. Procedure declaration

A procedure can be formed from any statement Σ (presumably a compound statement) by giving the procedure name, the list of identifiers representing the formal parameters and information concerning them and optionally the list of global identifiers, accompanied optionally by information concerning them.

Form

procedure I(R)=:(R)⊠ global (I,I,...I) ⊠, Σ

(global (I,I,...I) ⊠ may be empty)

where ⊠, ⊠ (if not empty) comprise the information referred to above.

If the statement Σ has quantities which are not local and not included in the formal parameter list, the corresponding identifiers can be listed in the list of global identifiers for the benefit of the user (and possibly the translator). In any case, they are global, which means their significance is prescribed from outside the procedure, i.e. in any compound statement in which the procedure is called (and in this compound statement they may denote local quantities).

The above holds also for functions in the form of procedures. Details are given below.

(Multiple procedure declarations are no longer necessary, since common parts of different procedures may be reached by those procedures by means of global identifiers.)

Form: 1 (procedure)

procedure $I(P_1)=:(P_0);D_1;D_2;D_3;\ldots D_n;$

$G_1;\ G_2;\Sigma$

Form: 2 (function in the form of a procedure)

type function $I(P_1);D_1;D_2;\ldots D_n;$

$G_1;\ G_2;\Sigma$

Here, I is the identifier of the procedure. P_i represents an ordered list of identifiers representing the formal input parameters, while P_o is the ordered list of identifiers representing the formal output parameters, which include any exits (formal labels or switches) required by the procedure. Either or both of the parts (P_i) and $=:(P_o)$ may be absent.

The D_1 to D_n are groups of symbols, in form similar to declarations, giving complete information concerning the input and output parameters. ? These are to be constructed in the following manner. The group D_i must give information for each formal variable which is not simply an ordinary variable. If necessary the declarations D_1 will use dummy identifiers. The possible types of information will be of the form:

type (I,I,I,...)

array (I,I,...I[d:d], I,I,...I[d,d:d,d].....)

type array label (I,I,...I)

switch (I:=(d,d,...d), I:=(d,d,...d))

function (I(d,d,...d), I(d,d,...d)....)

type function procedure (I(d,d,d)=:(d,d,...d), I(d,d,...d)=:(d,d,...d)...)

where the d's are dummy identifiers. Again D_2 must provide information for all d's appearing within D_1 which are not ordinary variables, written in the same form as the information for D_2, and so on.

The G_1 is an optional declaration of the form

global (I,I,...I)

The G_2 is an optional group of symbols in form similar to declarations giving information in the above manner, concerning the global identifiers.

The entry to the procedure is the first statement following begin. At least one operational end of the procedure must be indicated by a return statement. In the compound statement for Form 2 (function) a value must be assigned within the statement by an assignment "I:=F", where I is the identifier naming the function.

3. Procedure call

In a procedure call, the nature of the quantities entered as input or output variables or expressions must correspond exactly with the nature of the parameters in the procedure declaration taken in the same order.

By "nature" here we mean the types and for subscripted variables, functions and procedures also the number, order and nature of their subscripts and parameters.

By "correspond" here in the case of types we mean that the type of the actual variable or expression is compatible with the type of the corresponding parameter. The arithmetic treatment will in any case be in accordance with the type declaration in the procedure declaration.

Any expressions involved are to be evaluated when the procedure call is encountered (cf. similar treatment of for-statements).

4. Type declaration and compatibility

1) Expressions which have according to the rules available to the translator (and stated in the ALGOL-60 report), a type T 1, may be used in replacements (assignment statements, for-statements, procedure statements) where the quantity to be replaced has been declared explicitly or implicitly to be of a type T 2 which includes T 1.

2) The rules should assign types to permitted combinations of variables of different as well as of equal types.

3) Even where an expression has on mathematical grounds a type T 1 included in and different from the type T 2 which is given by the rules it must be treated as having the type T 2.

4) It is anticipated that in complicated cases of arithmetic, e.g. in multiple precision work, the above-mentioned rules should be supported by extra information, e.g. from a declaration governing the result y in a replacement y:=E.

5) Standard functions shall be provided that have the property that their value is by definition of a given type. An example of such a transfer-function is the function entier (χ) which takes values which are by definition declared to be integer.

5. Miscellaneous remarks

Don't use the delimiters for exponentiation.

Only integer valued expressions are permitted as subscripts and as subscript bounds.