8. Procedures and parameter transmission

8.1 Syntax

```
<procedure heading> ::= <procedure identifier>
                        <formal parameter part>;
                        <mode part><specification part>
<mode part> ::= <value part><name part>|
                <name part><value part>
<name part> ::= name <identifier list>;|
                <empty>
<specifier> ::= <type >|
                 array|
                <type> array|
                 label|
                 switch|
                 procedure|
                <type> procedure
<parameter delimiter> ::= ,
```

For actual parameter see section 7.1.1.

8.2 Semantics

With respect to procedures, SIMULA 67 deviates from ALGOL 60 on the following points:

1) Specification is required for each formal parameter.

2) The ALGOL specifier "string" is replaced by "text".

3) A "name part" is introduced as an optional part of a procedure heading to identify parameters called by name. Call by name is not the default parameter transmission mode.

4) Call by name is redefined in the case that the type of actual parameter does not coincide with that of the formal specification.

5) Exact type correspondence is required for array parameters not called by value.

There are three modes of parameter transmission: "call by value", "call by reference", and "call by name".

The default transmission mode is call by value for value type parameters and call by reference for all other kinds of parameters.

The available transmission modes are shown in fig. 8.1 for the different kinds of parameters to procedures. The upper left subtable defines transmission modes available for parameters of class declarations.

Parameter	Transmission modes		
	by value	by reference	by name
value type	D	I	O
object reference	I	D	O
text	O	D	O
value type array	O	D	O
reference type array	I	D	O
procedure	I	D	O
type procedure	I	D	O
label	I	D	O
switch	I	D	O

D: default mode O: optional mode I: illegal

fig. 8.1 Transmission modes

8.2.1 Call by value

A formal parameter called by value designates initially a local copy of the value (or array) obtained by evaluating the corresponding actual parameter. The evaluation takes place at the time of procedure entry or object generation.

The call by value of value type and value type array parameters is as in ALGOL 60.

A text parameter called by value is a local variable initialized in two steps, informally described by the statements:

```
FP :- blanks (AP.length); FP := AP;
```

where FP is the formal parameter and AP is the value of the actual parameter. Any text value is a legal actual parameter in this case.

Value specification is redundant for a parameter of value type.

There is no call by value option for object reference parameters and reference type array parameters.

8.2.2 Call by reference

A formal parameter called by reference designates initially a local copy of the reference obtained by evaluating the corresponding actual parameter. The evaluation takes place at the time of procedure entry or object generation.

A reference type formal parameter is a local variable initialized by a reference assignment

```
FP :- AP
```

where FP is the formal parameter and AP is the reference obtained by evaluating the actual parameter. The reference assignment is subject to the rules of section 6.1.2.2. Since in this case the formal parameter is a reference type variable, its contents may be changed by reference assignments within the procedure body, or within or without (by remote accessing) a class body. A string is not a legal actual parameter for a text parameter called by reference.

Although array-, procedure-, label-, and switch identifiers do not designate references to values, there is a strong analogy between references in the strict sense and references to entities such as arrays, procedures (i.e. procedure declarations), program points and switches. Therefore a call by reference mechanism is defined in these cases.

An array-, procedure-, label-, or switch parameter called by reference cannot be changed from within the procedure or class body; it will thus reference the same entity throughout its scope. However, the contents of an array called by reference may well be changed through appropriate assignments to its elements.

For an array parameter called by reference, the type associated with the actual parameter must coincide with that of the formal specification. For a procedure parameter called by reference, the type associated with the actual parameter must coincide with or be subordinate to that of the formal specification.

8.2.3 Call by name

Call by name is an optional transmission mode available for parameters to procedures. It represents a textual replacement as in ALGOL 60.

However, for an expression within a procedure body which is

1) a formal parameter called by name,

2) a subscripted variable whose array identifier is a formal parameter called by name, or

3) a function designator whose procedure identifier is a formal parameter called by name,

the following rules apply:

1) Its type is that prescribed by the corresponding formal specification.

2) If the type of the actual parameter does not coincide with that of the formal specification, then an evaluation of the expression is followed by an assignment of the value or reference obtained to a fictitious variable of the latter type. This assignment is subject to the rules of section 6.1.2. The value or reference obtained by the evaluation is the contents of the fictitious variable.

Section 6.1.2 defines the meaning of an assignment to a variable which is a formal parameter called by name, or is a subscripted variable whose array identifier is a formal parameter called by name, if the type of the actual parameter does not coincide with that of the formal specification.

Assignment to a procedure identifier which is a formal parameter is illegal, regardless of its transmission mode.

Notice that each dynamic occurrence of a formal parameter called by name, regardless of its kind, may invoke the execution of a non-trivial expression, e.g. if its actual parameter is a remote identifier.