3. Types and variables

3.1 Syntax

```
<type declaration> ::= <type><type list>
<array declaration> ::= array <array list>|
                        <type> array <array list>
<type> ::= <value type>|
           <reference type>
<value type> ::= integer|
                 real|
                 Boolean|
                 character
<reference type> ::= <object reference>|
                     text
<object reference> ::=  ref (<qualification>)
<qualification> ::= <class identifier>
```

3.2 Semantics

The syntax for type declaration represents a deviation from ALGOL 60, in that own is not a part of SIMULA 67.

A "value" is a piece of information interpreted at run time to represent itself. Examples of values are: an instance of a real number, an object, or a piece of text. A "reference" is a piece of information which identifies a value, called the "referenced" value. The distinction between a reference and the referenced value is determined by context.

The reference concept corresponds to the intuitive notion of a "name" or a "pointer". It also reflects the addressing capability of computers: in certain simple cases a reference could be implemented as the memory address of a stored value.

For computer efficiency the reference concept is not introduced in its full generality. In particular, there is no reference concept associated with any value type.

A variable local to a block instance is a memory device whose "contents" is either a value or a reference, according to the type of the variable. A value type variable has a value which is the contents of the variable. A reference type variable is said to have a value which is the one referenced by the contents of the variable. The contents of a variable may be changed by an appropriate assignment operation, see section 6.1.

3.2.1 Object references

Associated with an object there is a unique "object reference" which identifies the object. And for any class C there is an associated reference type <u>ref</u> (C). A quantity of that type is said to be qualified by the class C. Its value is either an object, or the special value <u>none</u> which represents "no object". The qualification restricts the range of values to objects of classes included in the qualifying class. The range of values includes the value <u>none</u> regardless of the qualification.

3.2.2 Characters

A <u>character</u> value is an instance of an "internal character". For any given implementation there is a one-one mapping between a subset of internal characters and external ("printable") characters. The character sets (internal and external) are implementation defined.

3.2.2.1 Collating sequence

The set of internal characters is ordered according to an implementation defined collating sequence. The collating sequence defines a one-one mapping between internal characters and integers expressed by the function procedures:

integer procedure rank(c); character c;
whose value is in the range [0,N-1], where N is the number of internal characters, and

character procedure char(n); integer n;
The parameter value must be in the range [0,N-1], otherwise a run time error is caused.

Example:

Most character codes are such that the digits (0-9) are character values which are consecutive and in ascending order with respect to the collating sequence. Under this assumption, the expressions

"rank(c) - rank('0')" and "char(rank('0')+i)"

provide implementation independent conversion between digits and their arithmetic values.

3.2.2.2 Character subsets

Two character subsets are defined by the standard non-local procedures:

Boolean procedure digit(c); character c;
which is true if c is a digit, and

Boolean procedure letter(c); character c;
which is true if c is a letter.

3.2.3 Text

A text value is an ordered sequence, possibly empty, of internal characters. The number of characters is called the "length" of the text. A non-empty text value is either a "text object", or it is part of a longer character sequence which is a text object.

A text reference identifies a text value. Certain properties of a text reference are represented by procedures accessible through remote accessing (the dot notation). The text concept is further described in section 10.

3.2.4 Initialization

Any declared variable is initialized at the time of entry into the block to which the variable is local. The initial contents depends on the type of the variable.

real	0.0
integer	0
Boolean	false
character	implementation defined
object reference	none
text	notext (see section 10)

3.2.5 Subordinate types

An object reference is said to be "subordinate" to a second object reference if the qualification of the former is a subclass of the class which qualifies the latter.

A proper procedure is said to be of "type universal". Any type is subordinate to the universal type. (Cf. sections 2.2.3, 8.2.2 and 8.2.3.)