10. The type "text"

Cf. sections 3.2.3, 4.4.2, 5.2 and 5.4.

10.1 Text attributes

The following procedures are attributes of any text reference. They may be accessed by remote identifiers of the form

<simple text expression>.<procedure identifier>

integer procedure length	(cf. 10.2)
text procedure main	(cf. 10.2)
integer procedure pos	(cf. 10.3)
procedure setpos	(cf. 10.3)
Boolean procedure more	(cf. 10.3)
character procedure getchar	(cf. 10.3)
procedure putchar	(cf. 10.3)
text procedure sub	(cf. 10.7)
text procedure strip	(cf. 10.7)
integer procedure getint	(cf. 10.9)
real procedure getreal	(cf. 10.9)
integer procedure getfrac	(cf. 10.9)
procedure putint	(cf. 10.10)
procedure putfix	(cf. 10.10)
procedure putreal	(cf. 10.10)
procedure putfrac	(cf. 10.10)

In the following section "X" denotes a text reference unless otherwise is specified.

10.2 "length" and "main"

integer procedure length;

The value of "X.length" is the number of characters of the text value referenced by X (cf. section 3.2.3). "notext.length" is equal to zero.

text procedure main;

"X.main" is a reference to the text object which is, or contains, the text value referenced by X (cf. section 3.2.3).

"notext.main" is identical to "notext".

The following relations are true for any text reference X.

X.main.length ≥ X.length
X.main.main == X.main

10.3 Character access

The characters of a text are accessible one at a time. Any text reference contains a "position indicator", which identifies the currently accessible character, if any, of the referenced text object. The position indicator of a given text reference X is an integer in the range [1,X.length+1].

The position indicator of notext is equal to 1. A text reference obtained by calling any system defined text procedures(i.e. main, sub and strip) has its position indicator equal to 1.

The position indicator of a given text reference may be altered by the procedures "setpos", "getchar", and "putchar" of the text reference. Also any of procedures defined in sections 10.9 and 10.10 may alter the position indicator of the text reference which contains the procedure.

Position indicators are ignored and left unaltered by text reference relations, text value relations and text value assignments.

The following procedures are facilities available for character accessing. They are oriented towards sequential access.

integer procedure pos;

The value of "X.pos" is the current value of the position indicator of the text reference X.

procedure setpos(i); integer i;

The effect of "X.setpos(i)" is to assign the integer i to the position indicator of X, if i is in the range [1,X.length+1]. Otherwise the value X.length+1 is assigned.

Boolean procedure more;

The value of "X.more" is true if the position indicator is in the range [1,X.length]. Otherwise the value is false.

character procedure getchar;

The value of "X.getchar" is a copy of the currently accessible character of X, provided that the current value of X.more is true. Otherwise the evaluation constitutes a run time error. In the former case the position indicator of X is increased by one after the copying operation.

procedure putchar(c); character c;

The effect of "X.putchar(c)" is to replace the currently accessible character of X by a copy of the character c provided that the current value of X.more is true.

Otherwise the execution constitutes a run time error. In the former case the position indicator of X is increased by 1 after the replacement operation.

Example:

```
procedure compress(T); text T;
begin text U;  character c;
      T.setpos(1);  U :- T;
      for c := c while U.more do
      begin c := U.getchar;
            if c ≠ '_' then T.putchar(c)
      end;
      for c := c while T.more do T.putchar('_')
end compress;
```

The procedure will rearrange the characters of the text value referenced by its parameter. The non-blank characters are collected in the leftmost part of the text and the remainder, if any, is filled with the blank characters. Since the parameter is called by reference, its position indicator is not altered. The character constant '_' represents a blank character value.

10.4 Text generation

The following basic procedures are available for text object generation. The procedures are non-local.

```
text procedure blanks(n); integer n;
```

The reference value is a new text object of length n, filled with blank characters. If n=0, the reference value is notext. For n<0, a run-time error will occur.

text procedure copy (T); value T; text T;

The referenced value is a new text object, which is a copy of the text value which is (or is referenced by) the actual parameter.

Example:

The statement "T :- copy ("ABC")", where T is a text variable, is equivalent to the compound statement

begin T :- blanks(3); T := "ABC" end

10.5 Text reference assignment

Syntax, see section 6.1.

A text reference assignment causes a text reference to be assigned as the new contents of the left part. The text reference is a copy of the one which is obtained by evaluating the right part (see section 6.2), and includes a copy of its position indicator.

If X is a text variable and Y is a text reference, then after the execution of the reference assignment "X :- Y", the relations "X == Y" and "X.pos = Y.pos" both have the value true.

10.6 Text value assignment

Syntax, see section 6.1.

Let the left part of a text value assignment be a text of length Ll, and let the right part be of length Lr. If the right part is itself a text value assignment, Lr is defined as the length of its constituent left part.

The effect of the text value assignment depends on the relationship between Ll and Lr.

Ll = Lr : The character contents of the right part text are copied to the left part text.

Ll > Lr : The character contents of the right part text are copied to the first Lr characters of the left part text. The remaining Ll-Lr characters of the left part text are filled with blanks.

Ll < Lr : The statement constitutes a run time error.

The effect of a text value assignment is implementation defined if the left part and right part refer to overlapping texts.

The position indicators of the left and the right parts are ignored and remain unchanged.

If X and Y are non-overlapping texts of the same length then after the execution of the value assignment "X := Y", the relation "X = Y" is true.

10.7 Subtexts

Two procedures are available for referencing subtexts.

text procedure sub(i,n); integer i,n;

Let i and n be integers such that $i \geq 1$, $n \geq 0$, and $i + n \leq X.length+1$. Then the expression "X.sub(i,n)" refers to that part of the text value X whose first

character is character number i of X, and which contains n consecutive characters. The position indicator of the text reference is equal to 1, and defines a local character numbering within the subtext. The position indicator of X is ignored and not altered. In the exceptional case n = 0, the reference obtained is notext. If i and n do not satisfy the above conditions, a run time error is caused.

If legal, the Boolean expressions

X.sub(i,n).sub(j,m) == X.sub(i+j-1,m),

and n ≠ 0⊃X.main == X.sub(i,n).main

both have the value true.

text procedure strip;

The expression "X.strip" is equivalent to "X.sub(1,n)", where n is the smallest integer such that the remaining characters of X, if any, are blanks.

Let X and Y be text references. Then after the value assignment "X := Y", if legal, the relation

X.strip = Y.strip

has the value true.

10.8 Numeric text values

10.8.1 Syntax

<EMPTY> ::=
<DIGIT> ::= 0|1|2|3|4|5|6|7|8|9
<DIGITS> ::= <DIGIT>|<DIGITS><DIGIT>
<BLANKS> ::= <EMPTY>|<BLANKS><BLANK>

```
<SIGN> ::= <EMPTY>|+|-
<SIGN PART> ::= <BLANKS><SIGN><BLANKS>
<INTEGER ITEM >::= <SIGN PART><DIGITS>
<FRACTION >::= .<DIGITS>
<DECIMAL ITEM> ::= <INTEGER ITEM>|
                   <SIGN PART><FRACTION>|
                   <INTEGER ITEM><FRACTION>
<EXPONENT> ::= 10<INTEGER ITEM>
<REAL ITEM >::= <DECIMAL ITEM>|
                <SIGN PART><EXPONENT>|
                <DECIMAL ITEM><EXPONENT>
<GROUPS> ::= <DIGITS>|
             <GROUPS><BLANKS><DIGITS>
<GROUPED ITEM> ::= <SIGN PART><GROUPS>|
                   <SIGN PART>.<GROUPS>|
                   <SIGN PART><GROUPS>.<GROUPS>
<NUMERIC ITEM> ::= <REAL ITEM>|
                   <GROUPED ITEM>
```

10.8.2 Semantics

The syntax applies to sequences of characters, i.e. to text values. <BLANK> stands for a blank character.

A numeric item is a character sequence which is a production of <NUMERIC ITEM>. "Editing" and "de-editing" procedures are available for the conversion between arithmetic values and text values which are numeric items, and vice versa.

10.9 "De-editing" procedures

A de-editing procedure of a given text reference X operates in the following way:

1) The longest numeric term, if any, of a given form is located, which is contained in X and contains the first character of X. (Notice that leading blanks are accepted as part of any numeric item.)

2) If no such numeric item is found, a run time error is caused.

3) Otherwise the numeric item is interpreted as a number.

4) If that number is outside a relevant implementation defined range, a runtime error is caused.

5) Otherwise an arithmetic value is computed, which is equal to or approximates that number.

6) The position indicator of X is made one greater than the position of the last character of the numeric item.

The following de-editing procedures are available.

integer procedure getint;

The procedure locates an INTEGER ITEM. The function value is equal to the corresponding integer.

real procedure getreal;

The procedure locates a REAL ITEM. The function value is equal to or approximates the corresponding number. If the number is an integer within an implementation defined range, the conversion is exact.

integer procedure getfrac;

The procedure locates a GROUPED ITEM. In its interpretation of the GROUPED ITEM the procedure

will ignore any BLANKS and a possible decimal point. The function value is equal to the resulting integer.

10.10 Editing procedures

Editing procedures of a given text reference X serve to convert arithmetic values to numeric items. After an editing operation, the numeric item obtained, if any, is right adjusted in the text X and preceded by as many blanks as necessary to fill the text. The final value of the position indicator of X is equal X.length+1.

A positive number is edited without a sign, a negative number is edited with a minus sign immediately preceding the most significant character. Leading non-significant zeros are suppressed, except possibly in an EXPONENT.

If X is identical to notext, a runtime error is caused. Otherwise if the text value is too short to contain the resulting numeric item, an "edit overflow" is caused. Then an implementation defined character sequence is edited into the text. In addition, an appropriate warning will be given after the completion of a program execution if an edit overflow has occurred.

```
procedure putint(i); integer i;
```

The value of the parameter is converted to an INTEGER ITEM which designates an integer equal to that value.

```
procedure putfix(r,n); real r; integer n;
```

The resulting numeric item is an INTEGER ITEM if n = 0 or a DECIMAL ITEM with a FRACTION of n digits if n > 0. It designates a number equal to the value of r

or an approximation to the value of r, correctly rounded to n decimal places. If n < 0, a run time error is caused.

procedure putreal(r,n); real r; integer n;

The resulting numeric item is a REAL ITEM containing an EXPONENT with a fixed implementation defined number of characters. The EXPONENT is preceded by a SIGN PART if n = 0, or by an INTEGER ITEM with one digit if n = 1, or if n > 1, by a DECIMAL ITEM with an INTEGER ITEM of 1 digit only, and a fraction of n-1 digits. If n < 0 a run time error is caused.

In putfix and putreal, the numeric item designates that number of the specified form which differs by the smallest possible amount from the value of r or from the approximation to the value of r.

procedure putfrac(i,n); integer i,n;

The resulting numeric item is a GROUPED ITEM with no decimal point if n <= 0, and with a decimal point followed by total of n digits if n > 0. Each digit group consists of 3 digits, except possibly the first one, and possibly the last one following a decimal point. The numeric item is an exact representation of the number $i.10^{-n}$.

The editing and de-editing procedures are oriented towards "fixed field" text manipulation.

Example:

```
text Tr, type, amount, price, payment;
integer pay, total;
Tr :- blanks (80); type :- Tr.sub (1,10);
amount :- Tr.sub(20,5); price :- Tr.sub (30,6);
payment :- Tr.sub (60,10);
.................
if type.strip = "order" then
begin pay := amount.getint × price.getfrac;
      total := total + pay;
      payment.putfrac (pay,2)
end
```