Predefined SIMULA Procedures

1. Basic procedures

REM

```
    integer procedure rem(a, b); integer a, b;
    rem  :=   The remainder of dividing a by b
```

ABS

```
    <type of arg> procedure abs(arg); < arithmetical type > arg;
        abs := if arg < 0 then -arg else arg;
```

SIGN

```
    integer procedure sign(a); <arithmetical type> a;
        sign := if a < 0 then -1 else if a = 0 then 0 else 1;
```

ENTIER

```
    integer procedure entier(a); real a;
        entier := < the largest integer less than or equal to a >
```

MOD

```
    integer procedure mod(a, b); integer a, b;
        mod := < a modulo b >;
```

MIN

```
    <type> procedure min(a, b); <type> a, b;
        min := < the smaller of a and b >;
```

MAX

```
    <type> procedure max(a, b); <type> a, b;
        max := < the larger of a and b >;
```

Legal parameter types for MAX and MIN are **`text`**, **`character`**, **`real`**`-type` and **`integer`**`-type`.

UPPERBOUND

```
    integer procedure upperbound(arr, dim);
       <type> array arr;
       integer dim;
    upperbound := < the upper bound for the array in dimension dim >;
```

LOWERBOUND

```
    integer procedure lowerbound(arr, dim);
       <type> array arr;
       integer dim;
    lowerbound:= < the lower bound for the array in dimension dim >;
```

2. Time and date

DATETIME

```
    text procedure datetime;
    datetime :- copy( text with the format "YYYY-MM-DD HH:MM:SS.ssss");
```

CPUTIME

```
    long real procedure cputime;
    cputime  :=  the number of processor seconds used by the program
```

CLOCKTIME

```
long real procedure clocktime;
clocktime := < the number of seconds since midnight >
```

3. Text procedures

COPY

```
text procedure copy(t); text t;
copy :- a copy of the text object referenced by t
```

BLANKS

```
text procedure blanks(n); integer nl-
blanks :-  a text object containing n blanks
```

CHAR

```
character procedure char(n); integer n;
char := < the character with Rank equal to n
          in the locally used character sequence >;
```

ISOCHAR

```
character procedure isochar(n)- integer n;
isochar := < the character with Rank equal to n
          in the ISO character sequence >;
```

RANK

```
integer procedure rank(c); character c;
rank := < the Rank of character c in the locally used
          character sequence >;
```

ISORANK

```
integer procedure isorank(c); character c;
isorank := < the Rank of character c in the ISO
             character sequence >;
```

DIGIT

```
boolean procedure digit(c); character c;
digit := < true if c is a decimal digit; false otherwise >;
```

LETTER

```
boolean procedure letter(c); character c;
letter := < true if c is a letter; false otherwise >;
```

UPCASE

```
text procedure upcase(t); text t;
upcase :- < t after converting every letter in t to upper case >;
```

LOWCASE

```
    text procedure lowcase(t); text t;
    lowcase :- < t after converting every letter in t to lower case >;
```

4. Implementation-dependent procedures

MAXINT

```
    integer procedure maxint;
    maxint := < the largest possible integer > ;
```

MININT

```
    integer procedure minint;
    minint := < the smallest possible integer > ;
```

MAXREAL

```
    real procedure maxreal;
    maxreal := < the largest possible real >;
```

MINREAl

```
    real procedure minreal;
    minreal := < the smallest possible real >;
```

MAXRANK

```
    integer procedure maxrank;
    maxrank := <number of characters in the locally used character
                sequence> ;
```

ADDEPSILON

```
    <type of E> procedure addepsilon(E); <real-type> E;
    addepsilon := E + < the smallest possible difference between two
                        reals >;
```

AddEpsilon provides the *NEXT* real value (if it exists) according to the local implementation of real numbers.

SUBEPSILON

```
    <type of E> procedure subepsilon(E); <real-type> E;
    subepsilon := E - < the smallest possible difference between two
                       reals >;
```

SubEpsilon provides the *PREVIOUS* real value (if it exists) according to the local implementation of real numbers.

MAXLONGREAL

```
    long real procedure maxlongreal;
    maxlongreal := < the largest possible double precision real >;
```

MINLONGREAL

```
    long real procedure minlongreal;
    minlongreal := < the smallest possible double precision real >;
```

SIMULAID

```
    text simulaid;
```

```
The value of this text bas the following format:

<simid>!!! <siteid>!!!<OS>!!!<CPU>!!!<user>!!!<job>!!!<acc>!!!<prog>

where
   <simid> =    Identification of the Simula system
   <siteid>=    Identification of the installation
   <OS>    =    Identification of the operating system
   <CPU>   =    Identification of the computer
   <user>  =    Identification of the user
   <job>   =    Identification of the job
   <acc>   =    Identification of the account
   <prog>  =    Identification of the program
```

Note that the name of this procedure stands for SIMULA ID; and not SIMUL AID.

5. Debugging

SOURCELINE

```
integer procedure sourceline;
sourceline := < the number of the program line in which the
                invocation of this procedure occurs >  ;
```

ERROR

```
procedure error(t); text t;
    displays the text t and stops the program
```

TERMINATE_PROGRAM

```
procedure terminate_program;
    < Closes SYSIN and SYSOUT and then stops the program > ;
```

6. Mathematical fonctions

LN

```
real procedure ln(r); real r;
ln := < the natural logarithm of r >
```

LOG10

```
real procedure log10(r); real r;
log10 := < the base_10 logarithm of r
```

EXP

```
real procedure exp(r); real r;
exp := < e raised to the power of r >
```

SQRT

```
real procedure sqrt(r); real r;
sqrt := < the square root of r >;
```

Note: In all the trigonometric functions which follow angles are measured in **radians** and angular results are between `-pi/2 and pi/2` .

SIN

```
real procedure sin(r); real r;
sin := < the sine of r >;
```

COS

```
    real procedure cos(r); real r;
    cos := < the cosine of r >;
```

TAN

```
    real procedure tan(r); real r;
    tan :=  < the tangent of r >;
```

COTAN

```
    real procedure cotan(r); real r;
    cotan := < the cotangent of r
```

ARCSIN

```
    real procedure arcsin(r); real r;
    arcsin := < the arcsine of r );
```

ARCCOS

```
    real procedure arccos(r); real r;
    arccos : = the arccosine of r
```

ARCTAN

```
    real procedure arctan(r); real r;
    arctan := the arctangent of r
```

SINH

```
    real procedure sinh(r); real r;
    sinh := the hyperbolic sine of r
```

TANH

```
    real procedure tanh(r); real r;
    tanh := < the hyperbolic tangent of r > ;
```

---

7. Random numbers

DRAW

```
    boolean procedure draw(p, seed);
    name seed; real p; integer seed;
    draw := true with probability p, false with probability (1 - p)
```

RANDINT

```
    integer procedure randint(low, high, seed);
        name seed; integer low, high, seed;
```

Randint returns one of the integers in [low, low + 1,... high] with equal probability.

UNIFORM

```
    long real procedure uniform(low, high, seed);
    name seed; long real low, high; integer seed;
    uniform := a value chosen uniformly on the interval [low .. high]
```

NORMAL

```
    long real procedure normal(mean, stdv, seed);
        name seed; long real mean, stdv; integer seed;
```

Normal returns a value chosen so that, if the fonction is used a largenumber of times with the same parameters, the values will bave an average value **mean** with a standard deviation of **stdv**.

Some additional and rather specialized predefined functions that may be used to draw numbers randomly are not described here.