Parameters Module

Template Module

param

Uses

N/A

Syntax

Exported Constants

Constant name	Type	Value
E	real	7.17×10^{7}
TD	real	3.0
M	real	7.0
MK	real	2.86×10^{-53}
LSF	real	1.0

Exported Types

Param = ?

Exported Access Programs

Routine name	In	Out	Exceptions
new Param		Param	

Semantics

State Variables

Variable name	Type
a	real
b	real
t	real
w	real
tnt	real
$pbtol$	real
$asprat$	real
sd	real
h	real
gtf	real
ldf	real
$wtnt$	real
$sdvect$	sequence [3] of real
gt	String

State Invariant

None

Assumptions

None

Access Routine Semantics

new Param:

- transition:

$$
\begin{aligned}
a &:= 0.0 \\
b &:= 0.0 \\
t &:= 0.0 \\
w &:= 0.0 \\
tnt &:= 0.0 \\
pbtol &:= 0.0
\end{aligned}
$$

$asprat$:= 0.0

sd := 0.0

h := 0.0

gtf := 0.0

ldf := 0.0

$wtnt$:= 0.0

$sdvect$:=< 0.0 , 0.0 , 0.0 >

b := “”

- output:

  out := *self*

- exception:

  none

Input Format Module

Module

inputFormat

Uses

param

Syntax

Exported Access Programs

Routine name	In	Out	Exceptions
get_input	string, Param		

Semantics

Environment Variables

$filesys$: FileSystem Read

Assumptions

None

Access Routine Semantics

get_input($filename, params$):

- transition:

$$
\begin{aligned}
filesys &:= filename \\
params.a &:= filesys.readline \\
params.b &:= filesys.readline \\
params.t &:= filesys.readline \\
params.gt &:= filesys.readline \\
params.w &:= filesys.readline \\
params.tnt &:= filesys.readline
\end{aligned}
$$

$$
\begin{aligned}
params.sdvect[0] &:= filesys.readline \\
params.sdvect[1] &:= filesys.readline \\
params.sdvect[2] &:= filesys.readline \\
params.pbtol &:= filesys.readline
\end{aligned}
$$

- output:

  None

- exception:

  None

Input Constraints Module

Module

derivedValues

Uses

param

Syntax

Exported Access Programs

Routine name	In	Out	Exceptions
derived_params	Param	Param	

Semantics

Assumptions

None

Access Routine Semantics

derived_params(*params*):

- transition:

$$
\begin{aligned}
params.asprat &:= \frac{params.a}{params.b} \\
params.sd &:= \sqrt{params.sdvect[0]^2 + params.sdvect[1]^2 + params.sdvect[2]^2} \\
params.ldf &:= \frac{params.td}{60.0} \frac{params.m}{16.0} \\
params.wtnt &:= params.w \times params.tnt
\end{aligned}
$$

$$
\begin{aligned}
params.h \quad := (\ & params.t = 2.50 \implies 2.16 \\
| \ & params.t = 2.70 \implies 2.59 \\
| \ & params.t = 3.0 \implies 2.92 \\
| \ & params.t = 4.0 \implies 3.78 \\
| \ & params.t = 5.0 \implies 4.57 \\
| \ & params.t = 6.0 \implies 5.56 \\
| \ & params.t = 8.0 \implies 7.42 \\
| \ & params.t = 10.0 \implies 9.02 \\
| \ & params.t = 12.0 \implies 11.91 \\
| \ & params.t = 16.0 \implies 15.09 \\
| \ & params.t = 19.0 \implies 18.26 \\
| \ & params.t = 22.0 \implies 21.44 \\
| \ & [otherwise] \implies 0.0\ )
\end{aligned}
$$

$$
\begin{aligned}
params.gtf \quad := (\ & params.gt = \text{“AN”} \vee params.gt = \text{“an”} \implies 1.0 \\
| \ & params.gt = \text{“HS”} \vee params.gt = \text{“hs”} \implies 2.0 \\
| \ & params.gt = \text{“FT”’} \vee params.gt = \text{“ft”} \implies 3.0 \\
| \ & [otherwise] \implies 0.0\ )
\end{aligned}
$$

- output:

$$out := params$$

- exception:

None

Input Verification Module

Module

checkConstraints

Uses

param

Syntax

Exported Access Programs

Routine name	In	Out	Exceptions
check_constraints	Param		INPUTERROR

Semantics

Assumptions

None

Access Routine Semantics

check_constraints(*params*):

- transition:

  None

- output:

  None

- exception:

$$
\begin{aligned}
exc := \ & (\ params.a \leq 0.0 \vee params.b \leq 0.0 \implies \text{INPUTERROR} \\
& \mid \neg(1.0 \leq params.asprat \leq 5.0) \implies \text{INPUTERROR} \\
& \mid params.t \notin \{2.50, 2.70, 3.0, 4.0, 5.0, 6.0, 8.0, 10.0, 12.0, 16.0, 19.0, 22.0\} \\
& \quad \implies \text{INPUTERROR} \\
& \mid params.gt \notin \{\text{“AN”}, \text{“an”}, \text{“HS”}, \text{“hs”}, \text{“FT”}, \text{“ft”}\} \\
& \quad \implies \text{INPUTERROR} \\
& \mid params.tnt \leq 0.0 \implies \text{INPUTERROR} \\
& \mid \neg(4.5 \leq params.wtnt \leq 910.0) \implies \text{INPUTERROR} \\
& \mid \neg(6.0 \leq params.sd \leq 130.0) \implies \text{INPUTERROR} \ )
\end{aligned}
$$

Table Input Module

Module

readTable

Uses

None

Syntax

Exported Access Programs

Routine name	In	Out	Exceptions
read_table	string	sequence [2, ?, ?] of real	FILEERROR

Semantics

State Variables

contents : sequence [?, ?] of string

Environment Variables

$filesys$: FileSystem Read

Assumptions

None

Access Routine Semantics

read_table($filename$):

- transition:

  $contents := map\ splitOn(',')\ filesys.readall(filename)$

- output:

  $out := map\ (\lambda x \rightarrow x[1 : : 2])\ contents\ ||\ map\ (\lambda x \rightarrow x[2 : : 2])\ contents$

- exception:

$$exc := \neg filesys.exists(filename) \implies \text{FILEERROR}$$