Text Editor ADT

14585438 Lawrence Thorpe

NAME

Text editor - an ADT representing an editor with text and left and right cursors for selecting text

SETS

N the set of Natural Numbers $\{\forall n.n \in \mathbb{Z} \wedge n \geq 0\}$
C the set of Characters $\{a..z \cup A..Z \cup 0..9\}$
T the set of Text $\{\ [C]\ \}$
E the set of Text Editors $\{(T \times N \times N)\}$

SYNTAX

create:	$\perp$	$\rightarrow E$
destroy:	E	$\rightarrow \perp$
init:	E	$\rightarrow E$
getText:	E	$\rightarrow T$
setText:	$E \times T$	$\rightarrow E$
insertCharacter	$E \times C$	$\rightarrow E$
getTextLength:	E	$\rightarrow N$
moveCursorLeft:	E	$\rightarrow E$
moveCursorRight:	E	$\rightarrow E$
jumpCursorWordStart:	E	$\rightarrow E$
jumpCursorWordEnd:	E	$\rightarrow E$
selectCursorLeft:	E	$\rightarrow E$
selectCursorRight:	E	$\rightarrow E$
selectCursorToWordStart:	E	$\rightarrow E$
selectCursorToWordEnd:	E	$\rightarrow E$
selectCursorToStart:	E	$\rightarrow E$
selectCursorToEnd:	E	$\rightarrow E$
jumpCursorToStart:	E	$\rightarrow E$
jumpCursorToEnd:	E	$\rightarrow E$
copySelection:	E	$\rightarrow (E \times T)$
pasteText:	$E \times T$	$\rightarrow E$
deleteSelection:	E	$\rightarrow E$
cutSelection:	E	$\rightarrow (E \times T)$
deleteCursorLeft:	E	$\rightarrow E$
deleteCursorRight:	E	$\rightarrow E$
setTextFromFile:	$E \times T$	$\rightarrow E$
saveTextToFile:	$E \times T$	$\rightarrow E$

SEMANTICS

$\forall t.t \in T, \forall c_L.c_L \in N, \forall c_R.c_R \in N, \forall x.x \in T, \forall y.y \in C$

pre-create() :: true
post-create(r) :: r = ([], 0, 0)

```
pre-destroy(e) :: true
post-destroy(e; r) :: r = $\perp$

pre-init(e) :: true
post-init((_, _, _); r) :: r = ([], 0, 0)

pre-getText(e) :: true
post-getText((t, _, _); r) :: r = t

pre-setText(e, t) :: $length(t) \leq 1024$
post-setText((_, _, _), x; r) :: r = (x, $length(x)$, $length(x)$)

pre-insertCharacter((t, _, _), c) :: $length(t) < 1024$
post-insertCharacter((t, $c_L$, $c_R$), y; r) :: r = (t[0..$c_L$] ^^ y ^^ t[$c_R$..$length(t)$], $c_L + 1$, $c_L + 1$)

pre-getTextLength(e) :: true
post-getTextLength((t, _, _)) :: r = TextLength(t)
    Where
    TextLength ([]) = 0
    TextLength (t.ts) = 1 + TextLength(ts)

pre-moveCursorLeft((_, $c_L$, _)) :: $c_L > 0$
post-moveCursorLeft((t, $c_L$, _); r) :: r = (t, $c_L - 1$, $c_L - 1$)

pre-moveCursorRight((t, _, $c_R$)) :: $c_R < length(t)$
post-moveCursorRight((t, _, $c_R$); r) :: r = (t, $c_R + 1$, $c_R + 1$)

pre-jumpCursorWordStart(e) :: true
post-jumpCursorWordStart((t, $c_L$, $c_R$); r) :: r = LeftUntilSpace(t, $c_L$, $c_R$)
    Where
    LeftUntilSpace ([], _, _) = ([], 0, 0)
    LeftUntilSpace (t, $c_L$, _) =
    If $c_L = 0 \lor$ (head . reverse) t[0..$c_L$] = ‘ ’
        Then (t, $c_L$, $c_R$)
    Else LeftUntilSpace (moveCursorLeft (t, $c_L$, $c_R$))

pre-jumpCursorWordEnd(e) :: true
post-jumpCursorWordEnd((t, $c_L$, $c_R$); r) :: r = RightUntilSpace(t, $c_L$, $c_R$)
    Where
    RightUntilSpace ([], _, _) = ([], 0, 0)
    RightUntilSpace (t, _, $c_R$) =
    If $c_R = length(t) \lor$ head t[$c_R$..$length(t)$] = ‘ ’
        Then (t, $c_L$, $c_R$)
    Else RightUntilSpace (moveCursorRight (t, $c_L$, $c_R$))

pre-selectCursorLeft((_, $c_L$, _)) :: $c_L > 0$
post-selectCursorLeft((t, $c_L$, $c_R$); r) :: r = (t, $c_L - 1$, $c_R$)

pre-selectCursorRight((t, _, $c_R$)) :: $c_R < length(t)$
post-selectCursorRight((t, $c_L$, $c_R$); r) :: r = (t, $c_L$, $c_R + 1$)

pre-selectCursorToWordStart(e) :: true
post-selectCursorToWordStart((t, $c_L$, $c_R$); r) :: r = SelectLeftUntilSpace(t, $c_L$, $c_R$)
    Where
    SelectLeftUntilSpace ([], _, _) = ([], 0, 0)
    SelectLeftUntilSpace (t, $c_L$, _) =
    If $c_L = 0 \lor$ (head . reverse) t[0..$c_L$] = ‘ ’
        Then (t, $c_L$, $c_R$)
    *Else* SelectLeftUntilSpace (t, $c_L - 1$, $c_R$)

pre-selectCursorToWordEnd(e) :: true
post-selectCursorToWordEnd((t, $c_L$, $c_R$); r) :: r = SelectRightUntilSpace(t, $c_L$, $c_R$)
    *Where*
    SelectRightUntilSpace ([], _, _) = ([], 0, 0)
    SelectRightUntilSpace (t, $c_L$, _) =
    *If* $c_R = length(t) \lor$ head t[$c_R..length(t)$] = ‘ ’
        *Then* (t, $c_L$, $c_R$)
    *Else* SelectRightUntilSpace (t, $c_R + 1$, $c_R + 1$)

pre-selectCursorToStart(e) :: true
post-selectCursorToStart((t, _, $c_R$); r) :: r = (t, 0, $c_R$)

pre-selectCursorToEnd(e) :: true
post-selectCursorToEnd((t, $c_L$, _); r) :: r = (t, $c_L$, $length(t)$)

pre-jumpCursorToStart(e) :: true
post-jumpCursorToStart((t, _, _); r) :: r = (t, 0, 0)

pre-jumpCursorToEnd(e) :: true
post-jumpCursorToEnd((t, _, _); r) :: r = (t, $length(t)$, $length(t)$)

pre-copySelection(e) :: true
post-copySelection((t, $c_L$, $c_R$); r) :: r = ((t, $c_L$, $c_R$), t[$c_L..c_R$])

pre-pasteText(e, t) :: true
post-pasteText((t, $c_L$, $c_R$), x; r) :: r = (t[0..$c_L$] ^^ x ^^ t[$c_R..length(t)$], $c_L + length(x)$, $c_L + length(x)$)

pre-deleteSelection(e) :: true
post-deleteSelection((t, $c_L$, $c_R$); r) :: r = (t[0..$c_L$] ^^ t[$c_R..length(t)$], $c_L$, $c_L$)

pre-cutSelection(e) :: true
post-cutSelection((t, $c_L$, $c_R$); r) :: r = (deleteSelection(t, $c_L$, $c_R$), snd copySelection(t, $c_L$, $c_R$))

pre-deleteCursorLeft((_, $c_L$, _)) :: $c_L > 0$
post-deleteCursorLeft((t, $c_L$, _); r) :: r = (t[0..($c_L - 1$)] ^^ t[$c_L..length(t)$], $c_L - 1$, $c_L - 1$)

pre-deleteCursorRight((t, _, $c_R$)) :: $c_R < length(t)$
post-deleteCursorRight((t, _, $c_R$); r) :: r = (t[0..$c_R$] ^^ t[($c_R + 1$)..$length(t)$], $c_R$, $c_R$)

pre-setTextFromFile(e, t) :: *File exists*
post-setTextFromFile((t, $c_L$, $c_R$), x; r) :: r = ReadFile((t, $c_L$, $c_R$), x)
    *Where*
    ReadFile((_, _, _), x) = *Read contents of file* x *to* t
        *Then* (t, $length(t)$, $length(t)$)

pre-saveTextToFile(e) :: true
post-saveTextToFile((t, $c_L$, $c_R$), x; r) :: r = WriteFile((t, $c_L$, $c_R$), x)
    *Where*
    WriteFile((t, $c_L$, $c_R$), x) = *Write contents of* t *to file* x
        *Then* (t, $c_L$, $c_R$)
```

AUTHOR

Lawrence Thorpe (THO14585438), University of Lincoln, UK, 2018