

String ライブラリー

佐原伸日本フィット株式会社
情報技術研究所

TEL : 03-3623-4683

shin.sahara@jfits.co.jp

平成 16 年 2 月 2 日

概 要

文字列 (seq of char) に関わる関数を提供するモジュールである。

0.1 String

文字列 (seq of char) に関わる関数を提供する。列型で定義された機能以外の機能を定義する。

class *String* is subclass of *Sequence*

0.1.1 変換関数群

数字文字列を整数に変換する

functions

public static

```
1.0 AsInteger : char* → ℤ
.1 AsInteger (s)  $\triangleq$ 
.2 String'AsIntegerAux (s) (0);
```

private static

```
2.0 AsIntegerAux : char* → ℤ → ℤ
.1 AsIntegerAux (s) (sum)  $\triangleq$ 
.2 if s = []
.3 then sum
.4 else AsIntegerAux (tl s) (10 × sum + Character'AsDigit (hd s));
```

0.1.2 判定関数群

数字文字列か判定する。

public static

```
3.0 IsDigits : char* → ℬ
.1 IsDigits (s)  $\triangleq$ 
.2 if s = []
.3 then true
.4 else Character'IsDigit (hd s) ∧ String'IsDigits (tl s);
```

空白かどうか判定する

public static

```
4.0 IsSpace : [char*] → ℬ
.1 IsSpace (s)  $\triangleq$ 
.2 if s = []
.3 then true
.4 else (hd s = '2' ∨ hd s = '\t') ∧ String'IsSpace (tl s);
```

文字列の辞書順序での大小を判定する。

public static

```

5.0   $LT : \text{char}^* \rightarrow \text{char}^* \rightarrow \mathbb{B}$ 
.1    $LT(s1)(s2) \triangleq$ 
.2     cases mk- ( $s1, s2$ ) :
.3       mk- ( $[], []$ )  $\rightarrow$  false,
.4       mk- ( $[], -$ )  $\rightarrow$  true,
.5       mk- ( $- \curvearrowright -, []$ )  $\rightarrow$  false,
.6       mk- ( $[x1] \curvearrowright xs1, [x2] \curvearrowright xs2$ )  $\rightarrow$ 
.7         if  $Character'LT(x1)(x2)$ 
.8         then true
.9         elseif  $Character'LT(x2)(x1)$ 
.10        then false
.11        else  $String'LT(xs1)(xs2)$ 
.12   end;

```

public static

```

6.0   $LE : \text{char}^* \rightarrow \text{char}^* \rightarrow \mathbb{B}$ 
.1    $LE(s1)(s2) \triangleq$ 
.2      $String'LT(s1)(s2) \vee s1 = s2$ ;

```

public static

```

7.0   $GT : \text{char}^* \rightarrow \text{char}^* \rightarrow \mathbb{B}$ 
.1    $GT(s1)(s2) \triangleq$ 
.2      $String'LT(s2)(s1)$ ;

```

public static

```

8.0   $GE : \text{char}^* \rightarrow \text{char}^* \rightarrow \mathbb{B}$ 
.1    $GE(s1)(s2) \triangleq$ 
.2      $\neg String'LT(s1)(s2)$ ;

```

$Index$ は、指定された文字 c が文字列 s の何番目にあるかを返す。最初の要素の位置を返す。

public static

```

9.0   $Index : \text{char} \rightarrow \text{char}^* \rightarrow \mathbb{Z}$ 
.1    $Index(c)(s) \triangleq$ 
.2      $Sequence'Index[\text{char}](c)(s)$ ;

```

$IndexAll$ は、指定された文字 c が文字列 s の何番目にあるかを持つ自然数集合を返す。

public static

```

10.0  $IndexAll : \text{char} \rightarrow \text{char}^* \rightarrow \mathbb{N}_1\text{-set}$ 
.1    $IndexAll(c)(s) \triangleq$ 
.2      $Sequence'IndexAll[\text{char}](c)(s)$ ;

```

文字列 s が、部分文字列として t を含むかを返す。

public static

```

11.0 IsSubStr : char* → char* → ℬ
.1 IsSubStr (t)(s)  $\triangleq$ 
.2   if t = ""
.3   then true
.4   else let indexSet = IndexAll (t (1)) (s) in
.5     ∃ i ∈ indexSet .
.6       SubStr (i) (len t) (s) = t;

```

0.1.3 文字列操作関数群

部分文字列を得る。

public static

```

12.0 SubStr : ℕ → ℕ → char+ → char*
.1 SubStr (i)(numOfChars)(s)  $\triangleq$ 
.2   Sequence 'SubSeq[char] (i) (numOfChars) (s);

```

部分文字列を得る。ただし、文字列長が指定された文字数より小さいとき、指定された詰め文字を補充する。

public static

```

13.0 SubStrFill : ℕ → ℕ → char → char* → char*
.1 SubStrFill (i)(numOfChars)(packChar)(s)  $\triangleq$ 
.2   let lastPos = i + numOfChars - 1,
.3   appendLen = lastPos - len s in
.4   if appendLen ≤ 0
.5   then SubStr (i) (numOfChars) (s)
.6   else SubStr (i) (numOfChars) (s)

```

MkContChar (*appendLen*) (*packChar*);

public static

```

14.0 MkContChar : ℕ1 → char → char*
.1 MkContChar (appendLen)(packChar)  $\triangleq$ 
.2   let r = λ x : char* · x ↗ [packChar] in
.3   (r ↑ appendLen) ("")

```

end *String*

Test Suite : vdm.tc

Class : String

Name	#Calls	Coverage
String'GE	3	✓
String'GT	4	✓
String'LE	6	✓
String'LT	51	✓

Name	#Calls	Coverage
String'Index	146	83%
String'SubStr	19	87%
String'IsSpace	10	✓
String'IndexAll	17	83%
String'IsDigits	12	✓
String'IsSubStr	6	✓
String'AsInteger	2	✓
String'MkContChar	3	✓
String'SubStrFill	7	✓
String'AsIntegerAux	12	✓
Total Coverage		98%

0.2 StringT

文字列のテストを行う。

class *StringT*

functions

public static

15.0 *run* : () → \mathbb{B}^*

.1 *run* () \triangleq

.2 let *testcases* =

.3 [

.4 *t1*, *t2*, *t3*, *t4*, *t5*, *t6*] in

.5 *Sequence*‘*Fmap*[*TestDriver*‘*TestCase**, \mathbb{B}] (*TestDriver*‘*run*) (*testcases*)

StringT01:.

values

16.0 *t1* = [

.1 mk-*TestDriver*‘*TestCase*

.2 (

.3 "*StringT01* : \tCompare-strings",

.4 let *LT* = *String*‘*LT*,

.5 *LE* = *String*‘*LE*,

.6 *GT* = *String*‘*GT*,

.7 *GE* = *String*‘*GE* in

.8 *LT* ("123") ("123") = false ∧

.9 *GT* ("123") ("123") = false ∧

.10 *LE* ("123") ("123") ∧

.11 *LE* ("123") ("1234") ∧

.12 *GE* ("123") ("1234") = false ∧

.13 ¬ *LE* ("1234") ("123") ∧

.14 *LE* ("") ("") ∧

.15 *Sequence*‘*Fmap*[char*, \mathbb{B}] (*LT* ("123")) ([*"123"*, *"1234"*, *" "*, *"223"*]) =

[false, true, false, true] ∧

.16 *Sequence*‘*Fmap*[char*, \mathbb{B}] (*LE* ("123")) ([*"1234"*, *" "*]) =

[true, false] ∧

.17 *Sequence*‘*Fmap*[char*, \mathbb{B}] (*GT* ("123")) ([*"123"*, *" "*, *"23"*]) =

[false, true, false] ∧

.18 *Sequence*‘*Fmap*[char*, \mathbb{B}] (*GE* ("123")) ([*"1234"*, *" "*]) =

[false, true]);

StringT02:

```

17.0  t2 = [
.1      mk-TestDriver`TestCase
.2      (
.3      "StringT02 : \tCompare-strings",
.4      let s1234 = "1234",
.5          s = new String()
.6      s1234 = "1234" ∧
.7      s.IsSpace("") = true ∧
.8      s.IsSpace("22") = true ∧
.9      s.IsSpace("2\t22") = true ∧
.10     s.IsSpace([]) = true)];

```

StringT03:。

```

18.0  t3 = [
.1      mk-TestDriver`TestCase
.2      (
.3      "StringT03 : \tGet-substrings",
.4      let s = new String()
.5          SubStr = String`SubStr in
.6      s.SubStr(6)(6)("Shin2Sahara") = "Sahara" ∧
.7      s.SubStr(6)(8)("Shin2Sahara") = "Sahara" ∧
.8      s.SubStr(6)(3)("Shin2Sahara") = "Sah" ∧
.9      s.SubStr(1)(0)("Shin2Sahara") = "" ∧
.10     s.SubStrFill(1)(3)(' * ')("sahara") = "sah" ∧
.11     s.SubStrFill(1)(6)(' * ')("sahara") = "sahara" ∧
.12     s.SubStrFill(1)(10)(' * ')("sahara") = "sahara * * *
*" ∧
.13     s.SubStrFill(3)(4)(' * ')("sahara") = "hara" ∧
.14     s.SubStrFill(3)(10)(' * ')("sahara") = "hara * * * *
*" ∧
.15     s.SubStrFill(1)(0)(' * ')("sahara") = "" ∧
.16     s.SubStrFill(1)(6)(' * ')("") = " * * * * * " ∧
.17     String`SubStr(6)(6)("Shin2Sahara") = "Sahara" ∧
.18     SubStr(6)(8)("Shin2Sahara") = "Sahara" ∧
.19     Sequence`Fmap[char*, char*](SubStr(6)(8))(["1234567890", "12345671"]) =
["67890", "671"]);

```

StringT04:。

```

19.0  t4 = [
.1      mk-TestDriver`TestCase
.2      (
.3      "StringT04 : \tCheck-digit-string",
.4      String`IsDigits("1234567890") = true ∧
.5      String`IsDigits("abc") = false ∧
.6      String`AsInteger("1234567890") = 1234567890 ∧
.7      String`AsInteger("") = 0)];

```

StringT05:、文字列に最初に出現する位置を検査する。

```

20.0  t5 = [
.1      mk-TestDriver `TestCase
.2      (
.3          "StringT05 : \tTest-Index-and-IndexAll",
.4          String'Index ('1') ("1234567890") = 1 ∧
.5          String'Index ('0') ("1234567890") = 10 ∧
.6          String'Index ('a') ("1234567890") = 0 ∧
.7          String'IndexAll ('1') ("1234567890") = {1} ∧
.8          String'IndexAll ('0') ("1234567890") = {10} ∧
.9          String'IndexAll ('a') ("1234567890") = {} ∧
.10         String'IndexAll ('1') ("1231567190") = {1, 4, 8} ∧
.11         String'IndexAll ('1') ("1231567191") = {1, 4, 8, 10} ∧
.12         String'Index ('1') ("1234567890") = 1 ∧
.13         String'Index ('0') ("1234567890") = 10 ∧
.14         String'Index ('a') ("1234567890") = 0 ∧
.15         String'IndexAll ('1') ("1234567890") = {1} ∧
.16         String'IndexAll ('0') ("1234567890") = {10} ∧
.17         String'IndexAll ('a') ("1234567890") = {} ∧
.18         String'IndexAll ('1') ("1231567190") = {1, 4, 8} ∧
.19         String'IndexAll ('1') ("1231567191") = {1, 4, 8, 10} ∧
.20         Sequence'Fmap[Char*, Z] (String'Index ('1')) (["1234567890", "2345671"]) =
[1, 7] ∧
.21         Sequence'Fmap[Char*, Z-set] (String'IndexAll ('1')) (["1231567190", "1231567191"]) =
[{1, 4, 8}, {1, 4, 8, 10}]);

```

StringT06:、ある文字列に含まれるかを検査する。

```

21.0  t6 = [
.1      mk-TestDriver `TestCase
.2      (
.3          "StringT06 : \tIs-substring",
.4          let IsSubStr = String'IsSubStr in
.5          String'IsSubStr ("abc") ("1234567890") = false ∧
.6          IsSubStr ("4F50539F4F38") ("4F50539F4F38") =
true ∧
.7          IsSubStr ("4F50") ("4F50539F4F38") = true ∧
.8          IsSubStr ("4F38") ("4F50539F4F38") = true ∧
.9          IsSubStr ("539F") ("4F50539F4F38") = true ∧
.10         IsSubStr ("") ("4F50539F4F38") = true)]

end StringT

```