System.Text.RegularExpressions Library Reference

September 24, 2014

Contents

Description

Provides basic support for regular expressions.

Copyrights

Namespaces

Namespace	Description
System.Text.RegularExpressions	Contains classes and functions that provide support for regular expressions.

1 Usage

1.0.1 Referencing the Regular Expression Library

Right-click a project node in IDE | Project References... | Add System Extension Library Reference... | enable *System.Text.RegularExpressions* check box

or add following line to your project's .cmp file:

```
reference <ext/System.Text.RegularExpressions/System.Text.RegularExpressions.cml>;
```

2 Syntax

2.0.1 Regular Expression Syntax

⟨*regular-expression*⟩ ::= ⟨*alternative-expr*⟩

2.0.1.1 Alternatives

Alternatives are separated by vertical bar.

⟨*alternative-expr*⟩ ::= ⟨*sequence-expr*⟩ (`|` ⟨*sequence-expr*⟩)*

2.0.1.2 Sequences

Catenated postfix expressions form sequences.

⟨*sequence-expr*⟩ ::= ⟨*postfix-expr*⟩ (⟨*postfix-expr*⟩)*

2.0.1.3 Postfix Expressions

Asterisk after a primary expression means zero or more occurrences of that expression. Plus after a primary expression means one or more occurrences of that expression. Question mark after a primary expression means zero or one occurrences of that expression.

⟨*postfix-expr*⟩ ::= ⟨*primary-expr*⟩ (`*` | `+` | `?`)?

2.0.1.4 Primary Expressions

Primary expression can be a dot, a class escape, a character class, an alternative expression enclosed in parenthesis, an escape or any other character. Dot matches any character.

⟨*primary-expr*⟩ ::= `.` | ⟨*class-escape*⟩ | ⟨*char-class*⟩ | `(` ⟨*alternative-expr*⟩ `)` | ⟨*escape*⟩ | ⟨*other-char*⟩

2.0.1.5 Class Escapes

⟨*class-escape*⟩ ::= `\\` [dDsSwWlLuU]

\d matches a decimal digit.
\D matches anything but a decimal digit.
\s matches a space (space, tab, etc.).
\S matches anything but a space.

\w matches a letter, a digit or an underscore.
\W matches anything but a letter, a digit or an underscore.
\l matches a lowercase letter (non-standard).
\L matches anything but a lowercase letter (non-standard).
\u matches a uppercase letter (non-standard).
\U matches anything but a uppercase letter (non-standard).

2.0.1.6 Character Classes

A character class is denoted by enclosing the classes or ranges in brackets. If the first character after an opening bracket is a circumflex ˆ it matches anything but the classes and ranges that follow it.

⟨*char-class*⟩ ::= `[` `^`? (⟨*class*⟩ | ⟨*range*⟩)* `]`

⟨*class*⟩ ::= `[:alnum:]`
| `[:alpha:]`
| `[:blank:]`
| `[:cntrl:]`
| `[:digit:]`
| `[:graph:]`
| `[:lower:]`
| `[:print:]`
| `[:punct:]`
| `[:space:]`
| `[:upper:]`
| `[:word:]`
| `[:w:]`
| `[:xdigit:]`

Classes are described in table 2.1.

A character range is denoted by starting character or escape, a dash (-) and the ending character or escape. It matches any character within the range.

⟨*range*⟩ ::= ⟨*class-char*⟩ (`-` ⟨*class-char*⟩)?

⟨*class-char*⟩ ::= ⟨*escape*⟩ | 'any character except `\` or `]`'

2.0.1.7 Escapes

A backslash followed by letter 'x' or 'X' and a character code in hex forms a hexadecimal escape. A backslash followed by a character code expressed as three octal digits forms a octal escape. A backslash followed by letter 'd' or 'D' and a character code in decimal forms a decimal escape. A backslash followed by letter 'a', 'b', 'f', 'n', 'r', 't', 'v', or '0' forms a C-escape. C-escapes are described in table 2.2.

⟨*escape*⟩ ::= \ ([xX] ⟨*hex-digit*⟩+ | ⟨*octal-digit*⟩ ⟨*octal-digit*⟩ ⟨*octal-digit*⟩ | [dD] ⟨*decimal-digit*⟩+
| [abfnrtv0])

Table 2.1: Classes

Class	**Meaning**
`[:alnum:]`	Any alphanumeric character
`[:alpha:]`	Any alphabetic character
`[:blank:]`	Any whitespace character but not a line separator
`[:cntrl:]`	Any control character
`[:digit:]`	Any decimal digit
`[:graph:]`	Any graphical character
`[:lower:]`	Any lowercase character
`[:print:]`	Any printable character
`[:punct:]`	Any punctuation character
`[:space:]`	Any whitespace character
`[:upper:]`	Any uppercase character
`[:word:]`	Any word character (alphanumeric character or underscore)
`[:w:]`	Any word character (alphanumeric character or underscore)
`[:xdigit:]`	Any hexadecimal digit

Table 2.2: C-escapes

Escape	**Meaning**
\a	alert
\b	backspace
\f	formfeed
\n	newline
\r	carriage return
\t	horizontal tab
\v	vertical tab
\0	null

⟨*hex-digit*⟩ ::= [0-9a-fA-F]

⟨*octal-digit*⟩ ::= [0-7]

⟨*decimal-digit*⟩ ::= [0-9]

3 System.Text.RegularExpressions Name pace

Contains classes and functions that provide support for regular expressions.

3.1 Classes

Class	Description
CharClass	Represents set of characters.
Dfa	Represents a deterministic finite automaton (DFA).
DfaState	Represents a state of a deterministic finite automaton (DFA).
Nfa	Represents a nondeterministic finite automaton (NFA).
NfaState	Represents a state in a nondeterministic finite automaton (NFA).
NfaStateNumberSet	Represents a set of NFA state numbers.
PtrNfa	Represents a nondeterministic finite automaton whose transitions are represented by edges containing pointers.
PtrNfaEdge	Represents an edge of a nondeterministic finite automaton in which a transition is represented by a pointer to an NFA state.
PtrNfaState	Represents a state of an NFA in which edges contain pointers to states.
PtrNfaStateFactory	Represents a container of PtrNfaState objects implemented as a singleton.
RegEx	Represents a regular expression.

3.1.1 CharClass Class

Represents set of characters.

Syntax

```
public class CharClass;
```

3.1.1.1 Member Functions

Member Function	Description
CharClass()	Default constructor. Initializes an empty character class.
CharClass(CharClass&&)	Move constructor.
CharClass(const CharClass&)	Copy constructor.
CharClass(const String&, bool)	Constructor. Initializes a character class with specified range string and inverse flag.
CharClass(char)	Constructor. Initializes a character class with specified character.
CharClass(char, char)	Constructor. Initializes a character class with specified character range.
operator=(CharClass&&)	Move assignment.
operator=(const CharClass&)	Copy assignment.
~CharClass()	Destructor.
Bits() const	Returns a bit set representing the character ranges in this character class.
Clear()	Makes the character class empty.
Contains(char) const	Returns true, if the specified character is contained in this character class, false otherwise.
Inverse() const	A flag telling whether this character class is treated as inverse of the specified ranges.

Ranges() const	Returns a string representation of the ranges in this character class.

3.1.1.1.1 CharClass() Member Function

Default constructor. Initializes an empty character class.

Syntax

```
public nothrow CharClass();
```

3.1.1.1.2 CharClass(CharClass&&) Member Function

Move constructor.

Syntax

```
public nothrow CharClass(CharClass&& that);
```

Parameters

Name	Type	Description
that	CharClass&&	A character class to move.

3.1.1.1.3 CharClass(const CharClass&) Member Function

Copy constructor.

Syntax

```
public CharClass(const CharClass& that);
```

Parameters

Name	Type	Description
that	const CharClass&	A character class to copy.

3.1.1.1.4 CharClass(const String&, bool) Member Function

Constructor. Initializes a character class with specified range string and inverse flag.

Syntax

```
public CharClass(const String& ranges_, bool inverse_);
```

Parameters

Name	Type	Description

ranges_	const String&	A string that contains character ranges.
inverse_	bool	Whether to treat the character class as inverse of the specified ranges.

3.1.1.1.5 CharClass(char) Member Function

Constructor. Initializes a character class with specified character.

Syntax

```
public CharClass(char c);
```

Parameters

Name	Type	Description
c	char	The sole character to be included in character class.

3.1.1.1.6 CharClass(char, char) Member Function

Constructor. Initializes a character class with specified character range.

Syntax

```
public CharClass(char first, char last);
```

Parameters

Name	Type	Description
first	char	The first character to be included in character class.
last	char	The last character to be included in character class.

3.1.1.1.7 operator=(CharClass&&) Member Function

Move assignment.

Syntax

```
public nothrow void operator=(CharClass&& that);
```

Parameters

Name	Type	Description
that	CharClass&&	A character class to move.

3.1.1.1.8 operator=(const CharClass&) Member Function

Copy assignment.

Syntax

```
public void operator=(const CharClass& that);
```

Parameters

Name	Type	Description
that	const CharClass&	A character class to copy.

3.1.1.1.9 ∼CharClass() Member Function

Destructor.

Syntax

```
public nothrow ∼CharClass();
```

3.1.1.1.10 Bits() const Member Function

Returns a bit set representing the character ranges in this character class.

Syntax

```
public nothrow const BitSet& Bits() const;
```

Returns

const BitSet&

A bit set.

3.1.1.1.11 Clear() Member Function

Makes the character class empty.

Syntax

```
public nothrow void Clear();
```

3.1.1.1.12 Contains(char) const Member Function

Returns true, if the specified character is contained in this character class, false otherwise.

Syntax

```
public bool Contains(char c) const;
```

Parameters

Name	Type	Description

c	char	A character to test.

Returns

bool

Returns true, if the specified character is contained in this character class, false otherwise.

3.1.1.1.13 Inverse() const Member Function

A flag telling whether this character class is treated as inverse of the specified ranges.

Syntax

```
public nothrow bool Inverse() const;
```

Returns

bool

Returns true, if this character class is treated as inverse, false otherwise.

3.1.1.1.14 Ranges() const Member Function

Returns a string representation of the ranges in this character class.

Syntax

```
public String Ranges() const;
```

Returns

String

Returns a string representation of the ranges in this character class.

3.1.1.2 Nonmember Functions

Function	Description
operator<<(OutputStream&, const CharClass&)	Puts the string representation of the specified character class to an output stream.

3.1.1.2.1 operator<<(OutputStream&, const CharClass&) Function

Puts the string representation of the specified character class to an output stream.

Syntax

```
public OutputStream& operator<<(OutputStream& s, const CharClass& cls);
```

Parameters

Name	**Type**	**Description**
s	OutputStream&	An output stream.
cls	const CharClass&	A character class to put.

Returns

OutputStream&

Returns the output stream.

3.1.2 Dfa Class

Represents a deterministic finite automaton (DFA).

Syntax

```
public class Dfa;
```

3.1.2.1 Member Functions

Member Function	Description
Dfa()	Default constructor. Initializes a DFA that contains no states.
Dfa(const Dfa&)	Copy constructor.
Dfa(Dfa&&)	Move constructor.
operator=(Dfa&&)	Move assignment.
operator=(const Dfa&)	Copy assignment.
~Dfa()	Destructor.
AddState(const DfaState&)	Adds a DFA state to this DFA.
GetState(int) const	Returns a DFA state with the specified identifier.
Next(char, int)	Returns identifier of next DFA state when a transition with specified character and current state identifier is given.
NumStates() const	Returns the number of DFA states in this DFA.
SetAccepting(int)	Sets a DFA state with the specified identifier as accepting state.
SetNext(char, int, int)	Adds a transition with the specified character from the specified state identifier to the specified state identifier.

3.1.2.1.1 Dfa() Member Function

Default constructor. Initializes a DFA that contains no states.

Syntax

```
public Dfa();
```

3.1.2.1.2 Dfa(const Dfa&) Member Function

Copy constructor.

Syntax

```
public Dfa(const Dfa& that);
```

Parameters

Name	Type	Description
that	const Dfa&	A DFA to copy.

3.1.2.1.3 Dfa(Dfa&&) Member Function

Move constructor.

Syntax

```
public nothrow Dfa(Dfa&& that);
```

Parameters

Name	Type	Description
that	Dfa&&	A DFA to move.

3.1.2.1.4 operator=(Dfa&&) Member Function

Move assignment.

Syntax

```
public nothrow void operator=(Dfa&& that);
```

Parameters

Name	Type	Description
that	Dfa&&	A DFA to move.

3.1.2.1.5 operator=(const Dfa&) Member Function

Copy assignment.

Syntax

```
public void operator=(const Dfa& that);
```

Parameters

Name	Type	Description
that	const Dfa&	A DFA to copy from.

3.1.2.1.6 ∼Dfa() Member Function

Destructor.

Syntax

```
public nothrow ~Dfa();
```

3.1.2.1.7 AddState(const DfaState&) Member Function

Adds a DFA state to this DFA.

Syntax

```
public void AddState(const DfaState& state);
```

Parameters

Name	Type	Description
state	const DfaState&	A DFA state to add.

3.1.2.1.8 GetState(int) const Member Function

Returns a DFA state with the specified identifier.

Syntax

```
public nothrow const DfaState& GetState(int i) const;
```

Parameters

Name	Type	Description
i	int	Identifier of the DFA state to retrieve.

Returns

const DfaState&

Returns a DFA state with the specified identifier.

3.1.2.1.9 Next(char, int) Member Function

Returns identifier of next DFA state when a transition with specified character and current state identifier is given.

Syntax

```
public nothrow int Next(char c, int s);
```

Parameters

Name	Type	Description
c	char	A character for the transition.
s	int	Current state identifier.

Returns

int

Returns identifier of next DFA state.

3.1.2.1.10 NumStates() const Member Function

Returns the number of DFA states in this DFA.

Syntax

```
public nothrow int NumStates() const;
```

Returns

int

Returns the number of DFA states in this DFA.

3.1.2.1.11 SetAccepting(int) Member Function

Sets a DFA state with the specified identifier as accepting state.

Syntax

```
public nothrow void SetAccepting(int s);
```

Parameters

Name	Type	Description
s	int	An identifier of a DFA state.

3.1.2.1.12 SetNext(char, int, int) Member Function

Adds a transition with the specified character from the specified state identifier to the specified state identifier.

Syntax

```
public nothrow void SetNext(char c, int from, int to);
```

Parameters

Name	Type	Description
c	char	A character of the transition.

from	int	Identifier of the target state of the transition.
to	int	Identifier of the source state of the transition.

3.1.2.2 Nonmember Functions

Function	Description
operator<<(OutputStream&, const Dfa&)	Puts the string representation of the specified DFA to an output stream.

3.1.2.2.1 operator<<(OutputStream&, const Dfa&) Function

Puts the string representation of the specified DFA to an output stream.

Syntax

```
public OutputStream& operator<<(OutputStream& s, const Dfa& dfa);
```

Parameters

Name	Type	Description
s	OutputStream&	An output stream.
dfa	const Dfa&	A DFA to put.

Returns

OutputStream&

Returns the output stream.

3.1.3 DfaState Class

Represents a state of a deterministic finite automaton (DFA).

Syntax

```
public class DfaState;
```

3.1.3.1 Member Functions

Member Function	Description
DfaState()	Default constructor. Initializes an empty DFA state.
DfaState(DfaState&&)	Move constrctor.
DfaState(const DfaState&)	Copy constructor.
DfaState(int)	Constructor. Initializes a DFA state with the specified state identifier.
operator=(DfaState&&)	Move assignment.
operator=(const DfaState&)	Copy assignment.
~DfaState()	Destructor.
Accepting() const	Returns true if this state is an accepting state, false otherwise.
Id() const	Returns the identifier of this state.
Next(char) const	Returns the identifier of the next DFA state with the specified character, or -1 if there is no transition with the specified character.
SetAccepting()	Sets this DFA state as accepting state.
SetNext(char, int)	Adds a transition with the specified character to the specified state.

3.1.3.1.1 DfaState() Member Function

Default constructor. Initializes an empty DFA state.

Syntax

```
public DfaState();
```

3.1.3.1.2 DfaState(DfaState&&) Member Function

Move constrctor.

Syntax

```
public nothrow DfaState(DfaState&& that);
```

Parameters

Name	Type	Description
that	DfaState&&	A DFA state to move.

3.1.3.1.3 DfaState(const DfaState&) Member Function

Copy constructor.

Syntax

```
public DfaState(const DfaState& that);
```

Parameters

Name	Type	Description
that	const DfaState&	A DFA state to copy.

3.1.3.1.4 DfaState(int) Member Function

Constructor. Initializes a DFA state with the specified state identifier.

Syntax

```
public DfaState(int id_);
```

Parameters

Name	Type	Description
id_	int	A state identifier.

Remarks

The identifier is an index of the state in containing DFA.

3.1.3.1.5 operator=(DfaState&&) Member Function

Move assignment.

Syntax

```
public nothrow void operator=(DfaState&& that);
```

Parameters

Name	Type	Description
that	DfaState&&	A DFA state to move.

3.1.3.1.6 operator=(const DfaState&) Member Function

Copy assignment.

Syntax

```
public void operator=(const DfaState& that);
```

Parameters

Name	Type	Description
that	const DfaState&	A DFA state to copy.

3.1.3.1.7 ∼DfaState() Member Function

Destructor.

Syntax

```
public nothrow ~DfaState();
```

3.1.3.1.8 Accepting() const Member Function

Returns true if this state is an accepting state, false otherwise.

Syntax

```
public nothrow bool Accepting() const;
```

Returns

bool

Returns true if this state is an accepting state, false otherwise.

3.1.3.1.9 Id() const Member Function

Returns the identifier of this state.

Syntax

```
public nothrow int Id() const;
```

Returns

int

Returns the identifier of this state.

3.1.3.1.10 Next(char) const Member Function

Returns the identifier of the next DFA state with the specified character, or -1 if there is no transition with the specified character.

Syntax

```
public nothrow int Next(char c) const;
```

Parameters

Name	Type	Description
c	char	A character of the transition.

Returns

int

Returns the identifier of the next DFA state with the specified character, or -1 if there is no transition with the specified character.

3.1.3.1.11 SetAccepting() Member Function

Sets this DFA state as accepting state.

Syntax

```
public nothrow void SetAccepting();
```

3.1.3.1.12 SetNext(char, int) Member Function

Adds a transition with the specified character to the specified state.

Syntax

```
public nothrow void SetNext(char c, int n);
```

Parameters

Name	Type	Description
c	char	A character of the transition.
n	int	An identifier of the target state of the transition.

3.1.3.2 Nonmember Functions

Function	**Description**
operator<<(OutputStream&, const DfaState&)	Puts the string representation of the specified DFA state to an output stream.

3.1.3.2.1 operator<<(OutputStream&, const DfaState&) Function

Puts the string representation of the specified DFA state to an output stream.

Syntax

```
public OutputStream& operator<<(OutputStream& s, const DfaState& state);
```

Parameters

Name	**Type**	**Description**
s	OutputStream&	An output stream.
state	const DfaState&	A DFA state to put.

Returns

OutputStream&

Returns the output stream.

3.1.4 Nfa Class

Represents a nondeterministic finite automaton (NFA).

Syntax

```
public class Nfa;
```

3.1.4.1 Member Functions

Member Function	Description
Nfa()	Default constructor. Initializes an empty NFA.
Nfa(Nfa&&)	Move constructor.
Nfa(const Nfa&)	Copy constructor.
operator=(Nfa&&)	Move assignment.
operator=(const Nfa&)	Copy assignment.
~Nfa()	Destructor.
AddState(const NfaState&)	Adds an NFA state to this NFA.
GetState(int) const	Returns an NFA state with the specified identifier.
NumStates() const	Returns the number of states in this NFA.
States() const	Returns a list of NFA states in this NFA.

3.1.4.1.1 Nfa() Member Function

Default constructor. Initializes an empty NFA.

Syntax

```
public Nfa();
```

3.1.4.1.2 Nfa(Nfa&&) Member Function

Move constructor.

Syntax

```
public nothrow Nfa(Nfa&& that);
```

Parameters

Name	Type	Description
that	Nfa&&	An NFA to move.

3.1.4.1.3 Nfa(const Nfa&) Member Function

Copy constructor.

Syntax

```
public Nfa(const Nfa& that);
```

Parameters

Name	Type	Description
that	const Nfa&	An NFA to copy.

3.1.4.1.4 operator=(Nfa&&) Member Function

Move assignment.

Syntax

```
public nothrow void operator=(Nfa&& that);
```

Parameters

Name	Type	Description
that	Nfa&&	An NFA to move.

3.1.4.1.5 operator=(const Nfa&) Member Function

Copy assignment.

Syntax

```
public void operator=(const Nfa& that);
```

Parameters

Name	Type	Description
that	const Nfa&	An NFA to copy.

3.1.4.1.6 ~Nfa() Member Function

Destructor.

Syntax

```
public nothrow ~Nfa();
```

3.1.4.1.7 AddState(const NfaState&) Member Function

Adds an NFA state to this NFA.

Syntax

```
public void AddState(const NfaState& state);
```

Parameters

Name	Type	Description
state	const NfaState&	An NFA state to add.

3.1.4.1.8 GetState(int) const Member Function

Returns an NFA state with the specified identifier.

Syntax

```
public nothrow const NfaState& GetState(int i) const;
```

Parameters

Name	Type	Description
i	int	Identifier of the state to retrieve.

Returns

const NfaState&

Returns an NFA state with the specified identifier.

3.1.4.1.9 NumStates() const Member Function

Returns the number of states in this NFA.

Syntax

```
public nothrow int NumStates() const;
```

Returns

int

Returns the number of states in this NFA.

3.1.4.1.10 States() const Member Function

Returns a list of NFA states in this NFA.

Syntax

```
public nothrow const List<NfaState>& States() const;
```

Returns

const List<NfaState>&

Returns a list of NFA states in this NFA.

3.1.4.2 Nonmember Functions

Function	Description
operator<<(OutputStream&, const Nfa&)	Puts the string representation of the specified NFA to an output stream.

3.1.4.2.1 operator<<(OutputStream&, const Nfa&) Function

Puts the string representation of the specified NFA to an output stream.

Syntax

```
public OutputStream& operator<<(OutputStream& s, const Nfa& nfa);
```

Parameters

Name	Type	Description
s	OutputStream&	An output stream.
nfa	const Nfa&	An NFA to put.

Returns

OutputStream&

Returns the output stream.

3.1.5 NfaState Class

Represents a state in a nondeterministic finite automaton (NFA).

Syntax

```
public class NfaState;
```

3.1.5.1 Member Functions

Member Function	Description
NfaState()	Default constructor. Initializes an empty NFA state.
NfaState(const NfaState&)	Copy constructor.
NfaState(NfaState&&)	Move constructor.
NfaState(int)	Constructor. Initializes an NFA state with the specified identifier.
operator=(const NfaState&)	Copy assignment.
operator=(NfaState&&)	Move assignment.
∼NfaState()	Destructor.
Accepting()	Returns true, if this NFA state is an accepting state, false otherwise.
Epsilon() const	Returns a set of NFA state numbers to which there are epsilon transitions (transitions with an empty string) from this NFA state.
Id() const	Returns the identifier of this NFA state.
Next(char) const	Returns a set of NFA state numbers to which there are transitions with the given character from this NFA state.
SetAccepting()	Sets this NFA state as an accepting state.
SetNext(char, const NfaStateNumberSet&)	Adds transitions with the specified character to the specified states.

3.1.5.1.1 NfaState() Member Function

Default constructor. Initializes an empty NFA state.

Syntax

```
public NfaState();
```

3.1.5.1.2 NfaState(const NfaState&) Member Function

Copy constructor.

Syntax

```
public NfaState(const NfaState& that);
```

Parameters

Name	Type	Description
that	const NfaState&	An NFA state to copy.

3.1.5.1.3 NfaState(NfaState&&) Member Function

Move constructor.

Syntax

```
public nothrow NfaState(NfaState&& that);
```

Parameters

Name	Type	Description
that	NfaState&&	An NFA state to move.

3.1.5.1.4 NfaState(int) Member Function

Constructor. Initializes an NFA state with the specified identifier.

Syntax

```
public NfaState(int id_);
```

Parameters

Name	Type	Description
id_	int	A state identifier.

3.1.5.1.5 operator=(const NfaState&) Member Function

Copy assignment.

Syntax

```
public void operator=(const NfaState& that);
```

Parameters

Name	Type	Description
that	const NfaState&	An NFA state to copy.

3.1.5.1.6 operator=(NfaState&&) Member Function

Move assignment.

Syntax

```
public nothrow void operator=(NfaState&& that);
```

Parameters

Name	Type	Description
that	NfaState&&	An NFA state to move.

3.1.5.1.7 ~NfaState() Member Function

Destructor.

Syntax

```
public nothrow ~NfaState();
```

3.1.5.1.8 Accepting() Member Function

Returns true, if this NFA state is an accepting state, false otherwise.

Syntax

```
public inline nothrow bool Accepting();
```

Returns

bool

Returns true, if this NFA state is an accepting state, false otherwise.

3.1.5.1.9 Epsilon() const Member Function

Returns a set of NFA state numbers to which there are epsilon transitions (transitions with an empty string) from this NFA state.

Syntax

```
public nothrow const NfaStateNumberSet& Epsilon() const;
```

Returns

const NfaStateNumberSet&

Returns a set of NFA state numbers to which there are epsilon transitions (transitions with an empty string) from this NFA state.

3.1.5.1.10 Id() const Member Function

Returns the identifier of this NFA state.

Syntax

```
public inline nothrow int Id() const;
```

Returns

int

Returns the identifier of this NFA state.

3.1.5.1.11 Next(char) const Member Function

Returns a set of NFA state numbers to which there are transitions with the given character from this NFA state.

Syntax

```
public nothrow const NfaStateNumberSet& Next(char c) const;
```

Parameters

Name	Type	Description
c	char	A character of the transition.

Returns

const NfaStateNumberSet&

Returns a set of NFA state numbers to which there are transitions with the given character from this NFA state.

3.1.5.1.12 SetAccepting() Member Function

Sets this NFA state as an accepting state.

Syntax

```
public inline nothrow void SetAccepting();
```

3.1.5.1.13 SetNext(char, const NfaStateNumberSet&) Member Function

Adds transitions with the specified character to the specified states.

Syntax

```
public void SetNext(char c, const NfaStateNumberSet& s);
```

Parameters

Name	Type	Description
c	char	A character of the transition.
s	const NfaStateNumberSet&	A set of target state numbers.

3.1.5.2 Nonmember Functions

Function	Description
operator<<(OutputStream&, const NfaState&)	Puts the string representation of the specified NFA state to an output stream.

3.1.5.2.1 operator<<(OutputStream&, const NfaState&) Function

Puts the string representation of the specified NFA state to an output stream.

Syntax

```
public OutputStream& operator<<(OutputStream& s, const NfaState& state);
```

Parameters

Name	Type	Description
s	OutputStream&	An output stream.
state	const NfaState&	An NFA state to put.

Returns

OutputStream&

Returns the output stream.

3.1.6 NfaStateNumberSet Class

Represents a set of NFA state numbers.

Syntax

```
public class NfaStateNumberSet;
```

3.1.6.1 Member Functions

Member Function	Description
NfaStateNumberSet()	Default constructor. Initializes an empty set.
NfaStateNumberSet(const NfaStateNumberSet&)	Copy constructor.
NfaStateNumberSet(NfaStateNumberSet&&)	Move constructor.
NfaStateNumberSet(int)	Constructor. Initializes an NFA state number set with the specified state number.
operator=(const NfaStateNumberSet&)	Copy assignment.
operator=(NfaStateNumberSet&&)	Move assignment.
~NfaStateNumberSet()	Destructor.
Add(int)	Adds a state number to this NFA state number set.
CanonicalContains(int)	Returns true, if the specified state number is included in the set, false otherwise.
Contains(int)	Returns true, if the specified state number is included in the set, false otherwise.
IsEmpty() const	Returns true, if this NFA state number set is empty, false otherwise.
MakeCanonical()	Sorts state numbers in this set to ascending order.
Merge(const NfaStateNumberSet&)	Merges an NFA state number set to this state number set.

Rep() const	Returns the list of NFA state numbers in this NFA state number set.
ToString() const	Returns a string representation of this NFA state number set.

3.1.6.1.1 NfaStateNumberSet() Member Function

Default constructor. Initializes an empty set.

Syntax

```
public NfaStateNumberSet();
```

3.1.6.1.2 NfaStateNumberSet(const NfaStateNumberSet&) Member Function

Copy constructor.

Syntax

```
public NfaStateNumberSet(const NfaStateNumberSet& that);
```

Parameters

Name	Type	Description
that	const NfaStateNumberSet&	An NFA state number set to copy.

3.1.6.1.3 NfaStateNumberSet(NfaStateNumberSet&&) Member Function

Move constructor.

Syntax

```
public nothrow NfaStateNumberSet(NfaStateNumberSet&& that);
```

Parameters

Name	Type	Description
that	NfaStateNumberSet&&	An NFA state number set to move.

3.1.6.1.4 NfaStateNumberSet(int) Member Function

Constructor. Initializes an NFA state number set with the specified state number.

Syntax

```
public NfaStateNumberSet(int s);
```

Parameters

Name	Type	Description
s	int	An NFA state number.

3.1.6.1.5 operator=(const NfaStateNumberSet&) Member Function

Copy assignment.

Syntax

```
public void operator=(const NfaStateNumberSet& that);
```

Parameters

Name	Type	Description
that	const NfaStateNumberSet&	An NFA state number set to copy.

3.1.6.1.6 operator=(NfaStateNumberSet&&) Member Function

Move assignment.

Syntax

```
public nothrow void operator=(NfaStateNumberSet&& that);
```

Parameters

Name	Type	Description
that	NfaStateNumberSet&&	An NFA state number set to move.

3.1.6.1.7 ~NfaStateNumberSet() Member Function

Destructor.

Syntax

```
public nothrow ~NfaStateNumberSet();
```

3.1.6.1.8 Add(int) Member Function

Adds a state number to this NFA state number set.

Syntax

```
public void Add(int stateNumber);
```

Parameters

Name	Type	Description
stateNumber	int	A state number to add.

3.1.6.1.9 CanonicalContains(int) Member Function

Returns true, if the specified state number is included in the set, false otherwise.

Syntax

```
public nothrow bool CanonicalContains(int stateNumber);
```

Parameters

Name	Type	Description
stateNumber	int	A state number to test.

Returns

bool

Returns true, if the specified state number is included in the set, false otherwise.

Remarks

Assumes this NFA state number set is made canonical. In a canonical set state numbers are sorted in ascending order using the MakeCanonical() member function.

3.1.6.1.10 Contains(int) Member Function

Returns true, if the specified state number is included in the set, false otherwise.

Syntax

```
public nothrow bool Contains(int stateNumber);
```

Parameters

Name	Type	Description
stateNumber	int	A state number to test.

Returns

bool

Returns true, if the specified state number is included in the set, false otherwise.

3.1.6.1.11 IsEmpty() const Member Function

Returns true, if this NFA state number set is empty, false otherwise.

Syntax

```
public nothrow bool IsEmpty() const;
```

Returns

bool

Returns true, if this NFA state number set is empty, false otherwise.

3.1.6.1.12 MakeCanonical() Member Function

Sorts state numbers in this set to ascending order.

Syntax

```
public void MakeCanonical();
```

3.1.6.1.13 Merge(const NfaStateNumberSet&) Member Function

Merges an NFA state number set to this state number set.

Syntax

```
public void Merge(const NfaStateNumberSet& x);
```

Parameters

Name	Type	Description
x	const NfaStateNumberSet&	An NFA state number set to merge.

3.1.6.1.14 Rep() const Member Function

Returns the list of NFA state numbers in this NFA state number set.

Syntax

```
public nothrow const List<int>& Rep() const;
```

Returns

const List<int>&

Returns the list of NFA state numbers in this NFA state number set.

3.1.6.1.15 ToString() const Member Function

Returns a string representation of this NFA state number set.

Syntax

```
public String ToString() const;
```

Returns

String

Returns a string representation of this NFA state number set.

3.1.6.2 Nonmember Functions

Function	Description
operator==(const NfaStateNumberSet&, const NfaStateNumberSet&)	Compare two NFA state number sets for equality.
operator<<(OutputStream&, const NfaStateNumberSet&)	Puts the string representation of the specified NFA state number set to an output stream.

3.1.6.2.1 operator==(const NfaStateNumberSet&, const NfaStateNumberSet&) Function

Compare two NFA state number sets for equality.

Syntax

```
public nothrow bool operator==(const NfaStateNumberSet& left, const NfaStateNumberSet&
right);
```

Parameters

Name	Type	Description
left	const NfaStateNumberSet&	The first NFA state number set to compare.

right	const NfaStateNumberSet&	The second NFA state number set to compare.

Returns

bool

Returns true, if the first state number set contains the same state numbers as the second state number set, false otherwise.

3.1.6.2.2 operator<<(OutputStream&, const NfaStateNumberSet&) Function

Puts the string representation of the specified NFA state number set to an output stream.

Syntax

```
public OutputStream& operator<<(OutputStream& s, const NfaStateNumberSet& ns);
```

Parameters

Name	Type	Description
s	OutputStream&	An output stream.
ns	const NfaStateNumberSet&	An NFA state number set.

Returns

OutputStream&

Returns the output stream.

3.1.7 PtrNfa Class

Represents a nondeterministic finite automaton whose transitions are represented by edges containing pointers.

Syntax

```
public class PtrNfa;
```

3.1.7.1 Member Functions

Member Function	Description
PtrNfa()	Default constructor. Constructs an empty NFA.
PtrNfa(const PtrNfa&)	Copy constructor.
PtrNfa(PtrNfa&&)	Move constructor.
PtrNfa(PtrNfaState*, PtrNfaState*)	Constructor. Initializes an NFA with the specified start and end states.
operator=(const PtrNfa&)	Copy assignment.
operator=(PtrNfa&&)	Move assignment.
~PtrNfa()	Destructor.
AddState(PtrNfaState*)	Adds a state to this NFA.
End() const	Returns the end state of this NFA.
NumberStates()	Numbers the states in this NFA. The states are given ascending integer identifiers starting from zero.
SetEnd(PtrNfaState*)	Adds an end state to this NFA.
SetStart(PtrNfaState*)	Adds a start state to this NFA.
Start() const	Returns the start state of this NFA.
States() const	Returns a list of NFA states in this NFA.

3.1.7.1.1 PtrNfa() Member Function

Default constructor. Constructs an empty NFA.

Syntax

```
public PtrNfa();
```

3.1.7.1.2 PtrNfa(const PtrNfa&) Member Function

Copy constructor.

Syntax

```
public PtrNfa(const PtrNfa& that);
```

Parameters

Name	Type	Description
that	const PtrNfa&	An NFA to copy.

3.1.7.1.3 PtrNfa(PtrNfa&&) Member Function

Move constructor.

Syntax

```
public nothrow PtrNfa(PtrNfa&& that);
```

Parameters

Name	Type	Description
that	PtrNfa&&	An NFA to move.

3.1.7.1.4 PtrNfa(PtrNfaState*, PtrNfaState*) Member Function

Constructor. Initializes an NFA with the specified start and end states.

Syntax

```
public PtrNfa(PtrNfaState* start_, PtrNfaState* end_);
```

Parameters

Name	Type	Description
start_	PtrNfaState*	A start state.
end_	PtrNfaState*	An end state.

3.1.7.1.5 operator=(const PtrNfa&) Member Function

Copy assignment.

Syntax

```
public void operator=(const PtrNfa& that);
```

Parameters

Name	Type	Description
that	const PtrNfa&	An NFA to copy.

3.1.7.1.6 operator=(PtrNfa&&) Member Function

Move assignment.

Syntax

```
public nothrow void operator=(PtrNfa&& that);
```

Parameters

Name	Type	Description
that	PtrNfa&&	An NFA to move.

3.1.7.1.7 ~PtrNfa() Member Function

Destructor.

Syntax

```
public nothrow ~PtrNfa();
```

3.1.7.1.8 AddState(PtrNfaState*) Member Function

Adds a state to this NFA.

Syntax

```
public void AddState(PtrNfaState* state);
```

Parameters

Name	Type	Description
state	PtrNfaState*	A state to add.

3.1.7.1.9 End() const Member Function

Returns the end state of this NFA.

Syntax

```
public inline nothrow PtrNfaState* End() const;
```

Returns

PtrNfaState*

Returns the end state of this NFA.

3.1.7.1.10 NumberStates() Member Function

Numbers the states in this NFA. The states are given ascending integer identifiers starting from zero.

Syntax

```
public nothrow void NumberStates();
```

3.1.7.1.11 SetEnd(PtrNfaState*) Member Function

Adds an end state to this NFA.

Syntax

```
public inline void SetEnd(PtrNfaState* end_);
```

Parameters

Name	Type	Description
end_	PtrNfaState*	A state to add.

3.1.7.1.12 SetStart(PtrNfaState*) Member Function

Adds a start state to this NFA.

Syntax

```
public inline void SetStart(PtrNfaState* start_);
```

Parameters

Name	Type	Description
start_	PtrNfaState*	A state to add.

3.1.7.1.13 Start() const Member Function

Returns the start state of this NFA.

Syntax

```
public inline nothrow PtrNfaState* Start() const;
```

Returns

PtrNfaState*

Returns the start state of this NFA.

3.1.7.1.14 States() const Member Function

Returns a list of NFA states in this NFA.

Syntax

```
public nothrow const List<PtrNfaState*>& States() const;
```

Returns

const List<PtrNfaState*>&

Returns a list of NFA states in this NFA.

3.1.7.2 Nonmember Functions

Function	Description
operator<<(OutputStream&, const PtrNfa&)	Puts the string representation of the specified pointer NFA to an output stream.

3.1.7.2.1 operator<<(OutputStream&, const PtrNfa&) Function

Puts the string representation of the specified pointer NFA to an output stream.

Syntax

```
public OutputStream& operator<<(OutputStream& s, const PtrNfa& nfa);
```

Parameters

Name	Type	Description
s	OutputStream&	An output stream.
nfa	const PtrNfa&	A pointer NFA to put.

Returns

OutputStream&

Returns the output stream.

3.1.8 PtrNfaEdge Class

Represents an edge of a nondeterministic finite automaton in which a transition is represented by a pointer to an NFA state.

Syntax

```
public class PtrNfaEdge;
```

3.1.8.1 Member Functions

Member Function	Description
PtrNfaEdge()	Default constructor. Initializes an empty edge.
PtrNfaEdge(const PtrNfaEdge&)	Copy constructor.
PtrNfaEdge(PtrNfaEdge&&)	Move constructor.
PtrNfaEdge(const CharClass&, PtrNfaState*)	Constructor. Initializes an edge with the specified set of characters and the specified target state.
operator=(PtrNfaEdge&&)	Move assignment.
operator=(const PtrNfaEdge&)	Copy assignment.
~PtrNfaEdge()	Destructor.
Chars() const	Returns a set of characters associated with this edge.
Next() const	Returns the target state of this edge.

3.1.8.1.1 PtrNfaEdge() Member Function

Default constructor. Initializes an empty edge.

Syntax

```
public PtrNfaEdge();
```

3.1.8.1.2 PtrNfaEdge(const PtrNfaEdge&) Member Function

Copy constructor.

Syntax

```
public PtrNfaEdge(const PtrNfaEdge& that);
```

Parameters

Name	Type	Description
that	const PtrNfaEdge&	An edge to copy.

3.1.8.1.3 PtrNfaEdge(PtrNfaEdge&&) Member Function

Move constructor.

Syntax

```
public nothrow PtrNfaEdge(PtrNfaEdge&& that);
```

Parameters

Name	Type	Description
that	PtrNfaEdge&&	An edge to move.

3.1.8.1.4 PtrNfaEdge(const CharClass&, PtrNfaState*) Member Function

Constructor. Initializes an edge with the specified set of characters and the specified target state.

Syntax

```
public PtrNfaEdge(const CharClass& cls_, PtrNfaState* next_);
```

Parameters

Name	Type	Description
cls_	const CharClass&	A set of characters.
next_	PtrNfaState*	A target state.

3.1.8.1.5 operator=(PtrNfaEdge&&) Member Function

Move assignment.

Syntax

```
public nothrow void operator=(PtrNfaEdge&& that);
```

Parameters

Name	Type	Description
that	PtrNfaEdge&&	An edge to move.

3.1.8.1.6 operator=(const PtrNfaEdge&) Member Function

Copy assignment.

Syntax

```
public void operator=(const PtrNfaEdge& that);
```

Parameters

Name	Type	Description
that	const PtrNfaEdge&	An edge to copy,

3.1.8.1.7 ~PtrNfaEdge() Member Function

Destructor.

Syntax

```
public nothrow ~PtrNfaEdge();
```

3.1.8.1.8 Chars() const Member Function

Returns a set of characters associated with this edge.

Syntax

```
public inline nothrow const CharClass& Chars() const;
```

Returns

const CharClass&

Returns a set of characters associated with this edge.

3.1.8.1.9 Next() const Member Function

Returns the target state of this edge.

Syntax

```
public inline nothrow PtrNfaState* Next() const;
```

Returns

PtrNfaState*

Returns the target state of this edge.

3.1.8.2 Nonmember Functions

Function	Description
operator<<(OutputStream&, const PtrNfaEdge&)	Puts the string representation of the specified pointer NFA edge to an output stream.

3.1.8.2.1 operator<<(OutputStream&, const PtrNfaEdge&) Function

Puts the string representation of the specified pointer NFA edge to an output stream.

Syntax

```
public OutputStream& operator<<(OutputStream& s, const PtrNfaEdge& edge);
```

Parameters

Name	Type	Description
s	OutputStream&	An output stream.
edge	const PtrNfaEdge&	A pointer NFA to put.

Returns

OutputStream&

Returns the output stream.

3.1.9 PtrNfaState Class

Represents a state of an NFA in which edges contain pointers to states.

Syntax

```
public class PtrNfaState;
```

3.1.9.1 Member Functions

Member Function	Description
PtrNfaState()	Default constructor. Initializes an empty state.
PtrNfaState(PtrNfaState&&)	Move constructor.
operator=(PtrNfaState&&)	A move assignment.
~PtrNfaState()	Destructor.
Accepting() const	Returns true, if this state is an accepting state, false otherwise.
AddEdge(const PtrNfaEdge&)	Adds an edge to this NFA state.
AddEdge(PtrNfaEdge&&)	Adds an edge by moving it into this NFA state.
Clear()	Makes this NFA state empty.
Edges() const	Returns a list of edges this NFA state contains.
Id() const	Returns the identifier of this NFA state.
Next(char)	Returns a list of NFA states into which there are transitions with the specified character from this NFA state.
ResetAccepting()	Sets this NFA state to nonaccepting.
SetAccepting()	Sets this NFA state to an accepting state.

SetEdges(const List<PtrNfaEdge>&)	Replaces the contained list of edges with the specified list of edges.
SetId(int)	Sets the identifier of this NFA state.

3.1.9.1.1 PtrNfaState() Member Function

Default constructor. Initializes an empty state.

Syntax

```
public PtrNfaState();
```

3.1.9.1.2 PtrNfaState(PtrNfaState&&) Member Function

Move constructor.

Syntax

```
public nothrow PtrNfaState(PtrNfaState&& that);
```

Parameters

Name	Type	Description
that	PtrNfaState&&	A state to move.

3.1.9.1.3 operator=(PtrNfaState&&) Member Function

A move assignment.

Syntax

```
public nothrow void operator=(PtrNfaState&& that);
```

Parameters

Name	Type	Description
that	PtrNfaState&&	A state to move.

3.1.9.1.4 ~PtrNfaState() Member Function

Destructor.

Syntax

```
public nothrow ~PtrNfaState();
```

3.1.9.1.5 Accepting() const Member Function

Returns true, if this state is an accepting state, false otherwise.

Syntax

```
public inline nothrow bool Accepting() const;
```

Returns

bool

Returns true, if this state is an accepting state, false otherwise.

3.1.9.1.6 AddEdge(const PtrNfaEdge&) Member Function

Adds an edge to this NFA state.

Syntax

```
public void AddEdge(const PtrNfaEdge& edge);
```

Parameters

Name	Type	Description
edge	const PtrNfaEdge&	An edge to add.

3.1.9.1.7 AddEdge(PtrNfaEdge&&) Member Function

Adds an edge by moving it into this NFA state.

Syntax

```
public void AddEdge(PtrNfaEdge&& edge);
```

Parameters

Name	Type	Description
edge	PtrNfaEdge&&	An edge to add.

3.1.9.1.8 Clear() Member Function

Makes this NFA state empty.

Syntax

```
public nothrow void Clear();
```

3.1.9.1.9 Edges() const Member Function

Returns a list of edges this NFA state contains.

Syntax

```
public nothrow const List<PtrNfaEdge>& Edges() const;
```

Returns

const List<PtrNfaEdge>&

Returns a list of edges this NFA state contains.

3.1.9.1.10 Id() const Member Function

Returns the identifier of this NFA state.

Syntax

```
public inline nothrow int Id() const;
```

Returns

int

Returns the identifier of this NFA state.

3.1.9.1.11 Next(char) Member Function

Returns a list of NFA states into which there are transitions with the specified character from this NFA state.

Syntax

```
public List<PtrNfaState*> Next(char c);
```

Parameters

Name	Type	Description
c	char	A character.

Returns

List<PtrNfaState*>

Returns a list of NFA states into which there are transitions with the specified character from this NFA state.

3.1.9.1.12 ResetAccepting() Member Function

Sets this NFA state to nonaccepting.

Syntax

```
public inline nothrow void ResetAccepting();
```

3.1.9.1.13 SetAccepting() Member Function

Sets this NFA state to an accepting state.

Syntax

```
public inline nothrow void SetAccepting();
```

3.1.9.1.14 SetEdges(const List<PtrNfaEdge>&) Member Function

Replaces the contained list of edges with the specified list of edges.

Syntax

```
public void SetEdges(const List<PtrNfaEdge>& edges_);
```

Parameters

Name	Type	Description
edges_	const List<PtrNfaEdge>&	A list of edges.

3.1.9.1.15 SetId(int) Member Function

Sets the identifier of this NFA state.

Syntax

```
public inline nothrow void SetId(int id_);
```

Parameters

Name	Type	Description
id_	int	An identifier.

3.1.10 PtrNfaStateFactory Class

Represents a container of PtrNfaState objects implemented as a singleton.

Syntax

```
public class PtrNfaStateFactory;
```

3.1.10.1 Member Functions

Member Function	Description
PtrNfaStateFactory()	Default constructor. Initializes an empty factory.
~PtrNfaStateFactory()	Destructor. Destroys contained states.
Clear()	Destroys contained states.
FreeState(PtrNfaState*)	Clears the state and puts it into a stack of reusable states.
Instance()	Returns a reference to the singleton factory.
MakeState()	If there are reusable states in the stack returns the top state of the stack, otherwise creates a new PtrNfaState object, adds it to the list of states, and returns a pointer to it.

3.1.10.1.1 PtrNfaStateFactory() Member Function

Default constructor. Initializes an empty factory.

Syntax

```
public nothrow PtrNfaStateFactory();
```

3.1.10.1.2 ~PtrNfaStateFactory() Member Function

Destructor. Destroys contained states.

Syntax

```
public nothrow ~PtrNfaStateFactory();
```

3.1.10.1.3 Clear() Member Function

Destroys contained states.

Syntax

```
public void Clear();
```

3.1.10.1.4 FreeState(PtrNfaState*) Member Function

Clears the state and puts it into a stack of reusable states.

Syntax

```
public void FreeState(PtrNfaState* state);
```

Parameters

Name	Type	Description
state	PtrNfaState*	A state to free.

3.1.10.1.5 Instance() Member Function

Returns a reference to the singleton factory.

Syntax

```
public static PtrNfaStateFactory& Instance();
```

Returns

PtrNfaStateFactory&

Returns a reference to the singleton factory.

3.1.10.1.6 MakeState() Member Function

If there are reusable states in the stack returns the top state of the stack, otherwise creates a new PtrNfaState object, adds it to the list of states, and returns a pointer to it.

Syntax

```
public PtrNfaState* MakeState();
```

Returns

PtrNfaState*

Returns a pointer to an NFA state.

3.1.11 RegEx Class

Represents a regular expression.

Syntax

```
public class RegEx;
```

3.1.11.1 Member Functions

Member Function	Description
RegEx(const RegEx&)	Copy constructor.
RegEx(const String&)	Constructor. Initializes a regular expression with the specified expression pattern and the default flags.
RegEx(RegEx&&)	Move constructor.
RegEx(const String&, RegExFlags)	Constructor. Initializes a regular expression with the specified expression pattern and the specified flags.
operator=(RegEx&&)	Move assignment.
operator=(const RegEx&)	Copy assignment.
∼RegEx()	Destructor.
Compiled() const	Returns true, if this regular expression is compiled to a DFA, false otherwise.
GetDfa() const	If the regular expression is compiled to an DFA, returns a reference to the contained DFA, otherwise returns a reference to an empty DFA.
GetNfa() const	Returns a reference to a contained NFA.
Matches(const String&)	Matches the specified string agains the contained NFA or DFA, and returns true, if the given string matches this regular expression, false otherwise.

Pattern() const	Returns the regular expression pattern.

3.1.11.1.1 RegEx(const RegEx&) Member Function

Copy constructor.

Syntax

```
public RegEx(const RegEx& that);
```

Parameters

Name	Type	Description
that	const RegEx&	A regular expression to copy.

3.1.11.1.2 RegEx(const String&) Member Function

Constructor. Initializes a regular expression with the specified expression pattern and the default flags.

Syntax

```
public RegEx(const String& pattern_);
```

Parameters

Name	Type	Description
pattern_	const String&	A regular expression pattern.

Remarks

Parses the regular expression pattern and creates a PtrNfa representation of the expression and then converts the pointer NFA to a regular a NFA.

3.1.11.1.3 RegEx(RegEx&&) Member Function

Move constructor.

Syntax

```
public nothrow RegEx(RegEx&& that);
```

Parameters

Name	Type	Description
that	RegEx&&	A regular expression to move.

3.1.11.1.4 RegEx(const String&, RegExFlags) Member Function

Constructor. Initializes a regular expression with the specified expression pattern and the specified flags.

Syntax

```
public RegEx(const String& pattern_, RegExFlags flags);
```

Parameters

Name	Type	Description
pattern_	const String&	An expression pattern.
flags	RegExFlags	Flags.

Remarks

Parses the regular expression pattern and creates a PtrNfa representation of the expression. If the specified flags contain the matchLine flag, the expression is converted to expression .*<pattern_>.* first. Then converts the pointer NFA to a regular a NFA. If the specified flags contain the compiled flag, compiles the NFA to a DFA using the subset construction.

3.1.11.1.5 operator=(RegEx&&) Member Function

Move assignment.

Syntax

```
public nothrow void operator=(RegEx&& that);
```

Parameters

Name	Type	Description
that	RegEx&&	A regular expression to move.

3.1.11.1.6 operator=(const RegEx&) Member Function

Copy assignment.

Syntax

```
public void operator=(const RegEx& that);
```

Parameters

Name	Type	Description
that	const RegEx&	A regular expression to copy.

3.1.11.1.7 ~RegEx() Member Function

Destructor.

Syntax

```
public nothrow ~RegEx();
```

3.1.11.1.8 Compiled() const Member Function

Returns true, if this regular expression is compiled to a DFA, false otherwise.

Syntax

```
public nothrow bool Compiled() const;
```

Returns

bool

Returns true, if this regular expression is compiled to a DFA, false otherwise.

3.1.11.1.9 GetDfa() const Member Function

If the regular expression is compiled to an DFA, returns a reference to the contained DFA, otherwise returns a reference to an empty DFA.

Syntax

```
public nothrow const Dfa& GetDfa() const;
```

Returns

const Dfa&

If the regular expression is compiled to an DFA, returns a reference to the contained DFA, otherwise returns a reference to an empty DFA.

3.1.11.1.10 GetNfa() const Member Function

Returns a reference to a contained NFA.

Syntax

```
public nothrow const Nfa& GetNfa() const;
```

Returns

const Nfa&

Returns a reference to a contained NFA.

3.1.11.1.11 Matches(const String&) Member Function

Matches the specified string agains the contained NFA or DFA, and returns true, if the given string matches this regular expression, false otherwise.

Syntax

```
public bool Matches(const String& s);
```

Parameters

Name	Type	Description
s	const String&	A string to match.

Returns

bool

Returns true, if the given string matches this regular expression, false otherwise.

3.1.11.1.12 Pattern() const Member Function

Returns the regular expression pattern.

Syntax

```
public nothrow const String& Pattern() const;
```

Returns

const String&

Returns the regular expression pattern.

3.2 Functions

Function	Description
EpsilonClosure(int, const Nfa&)	Returns the set of states that are reachable from the specified state using epsilon transitions only.
EpsilonClosure(const NfaStateNumberSet&, const Nfa&)	Returns the set of states reachable from the specified set of states using epsilon transitions only.
Matches(const String&, const Nfa&)	Matches the given string against a regular expression by simulating a DFA using an NFA constructed from the regular expression. Returns true, if the specified string matches the regular expression, false otherwise.
Matches(const String&, const Dfa&)	Matches the given string against a regular expression using a DFA constructed from the regular expression. Returns true, if the specified string matches the regular expression, false otherwise.
MatchesSlow(const String&, const Nfa&)	Matches the given string against a regular expression by simulating a DFA using an NFA constructed from the regular expression. Returns true, if the specified string matches the regular expression, false otherwise.
Move(const NfaStateNumberSet&, char, const Nfa&)	Returns the set of states reachable from the specified states using transitions with the specified character and epsilon transitions.
ToDfa(const Nfa&)	Converts the given NFA to a DFA using the subset construction algorithm.
ToNfa(const PtrNfa&)	Converts a pointer NFA to a regular NFA.
ToNfa(const String&, RegExFlags)	Converts the specified regular expression pattern to an NFA using the specified option flags.

ToRanges(Class)	Converts the specified character class enumeration constant to a string that contains ranges of characters that represent the character class enumeration.

3.2.12 EpsilonClosure(int, const Nfa&) Function

Returns the set of states that are reachable from the specified state using epsilon transitions only.

Syntax

```
public NfaStateNumberSet EpsilonClosure(int s, const Nfa& nfa);
```

Parameters

Name	Type	Description
s	int	A state identifier.
nfa	const Nfa&	An NFA.

Returns

NfaStateNumberSet

Returns the set of states that are reachable from the specified state using epsilon transitions only.

3.2.13 EpsilonClosure(const NfaStateNumberSet&, const Nfa&) Function

Returns the set of states reachable from the specified set of states using epsilon transitions only.

Syntax

```
public NfaStateNumberSet EpsilonClosure(const NfaStateNumberSet& states, const
Nfa& nfa);
```

Parameters

Name	Type	Description
states	const NfaStateNumberSet&	A set of states.
nfa	const Nfa&	An NFA.

Returns

NfaStateNumberSet

Returns the set of states reachable from the specified set of states using epsilon transitions only.

3.2.14 Matches(const String&, const Nfa&) Function

Matches the given string against a regular expression by simulating a DFA using an NFA constructed from the regular expression. Returns true, if the specified string matches the regular expression, false otherwise.

Syntax

```
public bool Matches(const String& s, const Nfa& nfa);
```

Parameters

Name	Type	Description
s	const String&	A string to match.
nfa	const Nfa&	An NFA constructed from the regular expression.

Returns

bool

Returns true, if the specified string matches the regular expression, false otherwise.

3.2.15 Matches(const String&, const Dfa&) Function

Matches the given string against a regular expression using a DFA constructed from the regular expression. Returns true, if the specified string matches the regular expression, false otherwise.

Syntax

```
public nothrow bool Matches(const String& s, const Dfa& dfa);
```

Parameters

Name	Type	Description
s	const String&	A string to match.
dfa	const Dfa&	A DFA constructed from the regular expression.

Returns

bool

Returns true, if the specified string matches the regular expression, false otherwise.

3.2.16 MatchesSlow(const String&, const Nfa&) Function

Matches the given string against a regular expression by simulating a DFA using an NFA constructed from the regular expression. Returns true, if the specified string matches the regular expression, false otherwise.

Syntax

```
public bool MatchesSlow(const String& s, const Nfa& nfa);
```

Parameters

Name	Type	Description
s	const String&	A string to match.
nfa	const Nfa&	An NFA constructed from a regular expression.

Returns

bool

Returns true, if the specified string matches the regular expression, false otherwise.

Remarks

Uses slower but simpler matching algorithm than the Matches(const String&, const Nfa&) function.

3.2.17 Move(const NfaStateNumberSet&, char, const Nfa&) Function

Returns the set of states reachable from the specified states using transitions with the specified character and epsilon transitions.

Syntax

```
public NfaStateNumberSet Move(const NfaStateNumberSet& states, char c, const Nfa&
nfa);
```

Parameters

Name	Type	Description
states	const NfaStateNumberSet&	A set of states.

c	char	A character.
nfa	const Nfa&	An NFA.

Returns

NfaStateNumberSet

Returns the set of states reachable from the specified states using transitions with the specified character and epsilon transitions.

3.2.18 ToDfa(const Nfa&) Function

Converts the given NFA to a DFA using the subset construction algorithm.

Syntax

```
public Dfa ToDfa(const Nfa& nfa);
```

Parameters

Name	Type	Description
nfa	const Nfa&	An NFA to convert.

Returns

Dfa

Returns a DFA.

3.2.19 ToNfa(const PtrNfa&) Function

Converts a pointer NFA to a regular NFA.

Syntax

```
public Nfa ToNfa(const PtrNfa& ptrNfa);
```

Parameters

Name	Type	Description
ptrNfa	const PtrNfa&	A pointer NFA to convert.

Returns

Nfa

Returns a regular NFA.

Remarks

In a pointer NFA the edges contain pointers to NFA states. In a regular NFA the transitions are represented using state identifiers.

3.2.20 ToNfa(const String&, RegExFlags) Function

Converts the specified regular expression pattern to an NFA using the specified option flags.

Syntax

```
public Nfa ToNfa(const String& pattern, RegExFlags flags);
```

Parameters

Name	Type	Description
pattern	const String&	A regular expression pattern.
flags	RegExFlags	Conversion flags.

Returns

Nfa

Returns an NFA.

Remarks

First parses the pattern to a pointer NFA, then converts the pointer NFA to a regular NFA and returns it.

3.2.21 ToRanges(Class) Function

Converts the specified character class enumeration constant to a string that contains ranges of characters that represent the character class enumeration.

Syntax

```
public String ToRanges(Class cls);
```

Parameters

Name	Type	Description
cls	Class	A character class enumeration constant.

Returns

String

Returns a string that represent the character class.

3.3 Enumerations

Enumeration	Description
Class	A character class enumeration.
RegExFlags	Options for regular expression construction.

3.3.21.1 Class Enumeration

A character class enumeration.

Enumeration Constants

Constant	Value	Description
alnum	0	An alphanumeric character.
alpha	1	An alphabetic character.
blank	2	A white space character that is not line separator.
cntrl	3	A control character.
digit	4	A decimal digit.
graph	5	A graphical character.
lower	6	A lower-case letter.
print	7	A printable character.
punct	8	A punctuation character.
space	9	A white space character.
upper	10	An upper-case letter.
word	11	A word character (alphanumeric character or underscore.)
xdigit	12	A hexadecimal digit.

3.3.21.2 RegExFlags Enumeration

Options for regular expression construction.

Enumeration Constants

Constant	Value	Description
none	0	Use defaults.
compiled	1	Compile the regular expression to a DFA.

matchLine	2	Construct an automaton that will match a whole line of text. That is: .*<pattern>.*.

3.4 Constants

Constant	Type	Value	Description
eps	char	\0	A character (NUL) that represents an epsilon transition.