Optional Homework - Computational Logic

Generated by Doxygen 1.8.20

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Class Documentation

3.1 Number Class Reference

Class implementing the Number data-type interface.

```
#include <number.hpp>
```

Public Member Functions

- Number (const unsigned int base, const std::string &value_string)
- Number (const Number &other)
- Number (Number &&other)
- std::string get_value () const
- bool operator== (const Number &other) const
- Number & operator= (const Number &other)
- Number & operator= (Number &&other)
- Number operator+ (const Number &other) const
- Number operator- (const Number &other) const
- Number operator∗ (const Number &digit) const
- std::pair< Number, Number > operator/ (const Number &digit) const

Static Public Member Functions

- static bool validate_value_string (const unsigned int base, const std::string &value_string)

Public Attributes

- const unsigned int base

3.1.1 Detailed Description

Class implementing the Number data-type interface.

The class implements number handling in various bases (currently 2-16 are supported).

Basic functionality includes value validation and arithmetic operations:

- addition
- subtraction (with subtrehand $>$ minuend)
- multiplication (by digit)
- division (by digit)

3.1.2 Constructor & Destructor Documentation

3.1.2.1 Number() `[1/3]`

```
Number::Number (
            const unsigned int base,
            const std::string & value_string )
```

Constructor the Number instance.

Parameters

value_string	the value of the created number
base	the base of the created number

See also

validate_value_string()

Exceptions

std::runtime_error	on validation vail

3.1.2.2 Number() `[2/3]`

```
Number::Number (
            const Number & other )
```

Copy constructor

3.1.2.3 Number() [3/3]

```
Number::Number (
            Number && other )
```

Move constructor

3.1.3 Member Function Documentation

3.1.3.1 get_value()

```
std::string Number::get_value ( ) const
```

Returns the value string of the Number.

3.1.3.2 operator*()

```
Number Number::operator* (
            const Number & digit ) const
```

Multiplication operator

Implements the multiplication operation, using the algorithm studied in class.

The multiplication can be done only by a digit (other)

Parameters

other	the digit Number instance to multiply by

Exceptions

std::runtime_error	on other not being a digit and on different bases

Returns

the Number representing the sum of the numbers.

3.1.3.3 operator+()

```
Number Number::operator+ (
            const Number & other ) const
```

Addition operator

Implements the addition operation, using the algorithm studied in class.

Parameters

other	the Number instance to add with

Exceptions

std::runtime_error	on subtrahend < minuend and on different bases

Returns

the Number representing the sum of the numbers.

3.1.3.4 operator-()

```
Number Number::operator- (
            const Number & other ) const
```

Subtraction operator

Implements the subtraction operation, using the algorithm studied in class.

The subtrahend (this) must be equal or bigger than the minuend (other).

Parameters

other	the Number instance to be subtracted

Exceptions

std::runtime_error	on subtrahend < minuend. and on different bases

Returns

the Number representing the sum of the numbers.

3.1.3.5 operator/()

```
std::pair< Number, Number > Number::operator/ (
            const Number & digit ) const
```

Division operator

Implements the division operation, using the algorithm studied in class.

The division can be done only by a digit (other)

Parameters

other	the digit Number instance to multiply by

Exceptions

std::runtime_error	on other not being a digit and on different bases

Returns

pair of Number instances, where first is the quotinent and second the remainder

3.1.3.6 operator=() [1/2]

```
Number & Number::operator= (
             const Number & other )
```

Assignment operator

Exceptions

std::runtime_error	on different bases

3.1.3.7 operator=() [2/2]

```
Number & Number::operator= (
             Number && other )
```

Move assignment operator

Exceptions

std::runtime_error	on different bases

3.1.3.8 operator==()

```
bool Number::operator== (
             const Number & other ) const
```

Equality operator

Checks if two Number instances have the same base and string value.

Returns

true on equality, false otherwise

3.1.3.9 validate_value_string()

```
bool Number::validate_value_string (
             const unsigned int base,
             const std::string & value_string )  [static]
```

Checks if a value string is valid in the given base

A valid value string is defined as one that has each "digit" contained in the set of characters used by the base.

Parameters

base	the base to be checked in
value_string	the value string to be checked

Returns

true on success, false on fail

3.1.4 Member Data Documentation

3.1.4.1 base

```
const unsigned int Number::base
```

The base of the Number.

The documentation for this class was generated from the following files:

- src/number.hpp
- src/number.cpp

Chapter 4

File Documentation

4.1 src/convert.hpp File Reference

Base conversion algorithms.

```
#include "number.hpp"
```

Functions

- Number convert_fast (unsigned int dstBase, const Number &number)

    Converts number in another base using rapid conversion.
- Number convert_substitution (unsigned int dstBase, const Number &number)

    Converts number in another base using the substitution method.
- Number convert_successive_division (unsigned int dstBase, const Number &number)

    Converts number in another base using the successive division method.
- Number convert_base (unsigned int dstBase, const Number &number)

    General base conversion dispatcher.

4.1.1 Detailed Description

Base conversion algorithms.

Author

Stefan Stefanache (916/2)

Date

11.12.2020

4.1.2 Function Documentation

4.1.2.1 convert_base()

```
Number convert_base (
            unsigned int dstBase,
            const Number & number )
```

General base conversion dispatcher.

Under the hood, the following conversion implementations are used:

- convert_fast if the source and destination bases are powers of two
- convert_substitution if srcBase $<$ dstBase
- convert_successive if srcBase $>$ dstBase

Parameters

dstBase	the destination base
number	the Number to be converted

Exceptions

std::runtime_error	if dstBase is not supported

See also

isBaseSupported()

Returns

converted Number instance

4.1.2.2 convert_fast()

```
Number convert_fast (
            unsigned int dstBase,
            const Number & number )
```

Converts number in another base using rapid conversion.

Warning

This should be used only if both the source base and the destination base are powers of 2.

Parameters

dstBase	the destination base
number	the Number to be converted

Exceptions

std::runtime_error	if dstBase is not supported or the bases are not powers of two

Returns

converted Number in dstBase

4.1.2.3 convert_substitution()

```
Number convert_substitution (
            unsigned int dstBase,
            const Number & number )
```

Converts number in another base using the substitution method.

Warning

This should be used only if dstBase $>$ srcBase

Parameters

dstBase	the destination base
number	the Number to be converted

Exceptions

std::runtime_error	if dstBase is not supported or dstBase $<=$ srcBase

Returns

converted Number in dstBase

4.1.2.4 convert_successive_division()

```
Number convert_successive_division (
            unsigned int dstBase,
            const Number & number )
```

Converts number in another base using the successive division method.

Warning

This should be used only if dstBase $<$ srcBase

Parameters

dstBase	the destination base
number	the Number to be converted

Exceptions

std::runtime_error	if dstBase is not supported or dstBase >= srcBase

Returns

converted Number in dstBase

4.2 src/number.hpp File Reference

Implements generic base number interface.

```
#include <string>
#include <stdexcept>
#include "tools.hpp"
```

Classes

- class Number

  Class implementing the Number data-type interface.

4.2.1 Detailed Description

Implements generic base number interface.

Author

Stefan Stefanache (916/2)

Date

11.12.2020

4.3 src/tools.hpp File Reference

Implements various helper functions used in Number and convert implementations.

```
#include <string>
#include <vector>
```

Functions

- std::string get_base_characters (const unsigned int base)

    Returns std::vector of characters used in the given base.
- bool isBaseSupported (const unsigned int base)

    Checks if the given base is supported.
- bool is_power_of_two (const unsigned int number)

    Checks if a given number is a power of two.
- unsigned int digitToValue (const char character)

    Returns the decimal value of a digit.
- char valueToDigit (const unsigned int value)

    Returns the coresponding digit for a decimal value.
- unsigned int get_the_power_of_two (const unsigned int number)

    Returns log_2(n) if the given number is a power of 2.

Variables

- constexpr unsigned int BINARY_DIGIT_MAX_LENGTH = 4

    The maximum binary digit "pack" size used in fast conversion.
- const std::string HEX_BASE_CHARACTERS

    The characters used in hexadecimal representations.
- const std::vector< std::string > RAPID_CONVERSION_STRINGS

    The binary digit packs used in fast conversion.

4.3.1 Detailed Description

Implements various helper functions used in Number and convert implementations.

Author

Stefan Stefanache (916/2)

Date

11.12.2020

4.3.2 Function Documentation

4.3.2.1 digitToValue()

```
unsigned int digitToValue (
              const char character )
```

Returns the decimal value of a digit.

e.g. digitToValue('E') = 14

4.3.2.2 get_base_characters()

```
std::string get_base_characters (
              const unsigned int base )
```

Returns std::vector of characters used in the given base.

e.g. get_base_characters(4) returns { '0', '1', '2', '3' }

Parameters

base	the base to be used.

4.3.2.3 get_the_power_of_two()

```
unsigned int get_the_power_of_two (
            const unsigned int number )
```

Returns log_2(n) if the given number is a power of 2.

Otherwise, returns 0

4.3.2.4 isBaseSupported()

```
bool isBaseSupported (
            const unsigned int base )
```

Checks if the given base is supported.

Currently only bases in the interval 2-16 are supported.

4.3.2.5 valueToDigit()

```
char valueToDigit (
            const unsigned int value )
```

Returns the coresponding digit for a decimal value.

e.g. valueToDigit(14) = 'E'

Index