RPN

Generated by Doxygen 1.13.2

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Namespace Documentation

4.1 RPN Namespace Reference

Classes

- struct NotationConverter
- struct NotationDeterminer
- struct RPNSolver
- struct TokenReader

Functions

- int sumLetters (const std::string &str)
- double handleDivision (const double &a, const double &b)
- double handleSqrt (const double &a)
- double handleCbrt (const double &a)
- double calculate (const double &a, const double &b, const std::string &op)
- double calculate (const double &a, const std::string &op)
- bool isOperator (const std::string &op)
- bool is1ArgOperator (const std::string &op)
- bool is2ArgOperator (const std::string &op)

Variables

- constexpr int EXP_PREC = 100
- constexpr int TRIG_FUN_PREC = EXP_PREC-1
- constexpr int MULT_DIV_PREC = TRIG_FUN_PREC-1
- constexpr int ADD_SUB_PREC = MULT_DIV_PREC-1
- const std::map< std::string, int > operatorPrecedence
- const std::unordered_set< std::string > one_arg_operators
- const std::unordered_set< std::string > two_arg_operators

4.1.1 Function Documentation

4.1.1.1 calculate() [1/2]

```
double RPN::calculate (
            const double & a,
            const double & b,
            const std::string & op)
```

Given operator and operands calculates the result

Parameters

a	left operand
b	right operand
op	operator

Returns

result

4.1.1.2 calculate() [2/2]

```
double RPN::calculate (
            const double & a,
            const std::string & op)
```

Given operator and operand calculates the result

Parameters

a	operand
op	operator

Returns

result

Integers found in cases of this switch come from the sum of ascii values of letters of the operators.

4.1.1.3 handleCbrt()

```
double RPN::handleCbrt (
            const double & a)
```

Calculates cubic roots and errors on negative numbers.

Parameters

a	

Returns

cbrt(a)

4.1.1.4 handleDivision()

```
double RPN::handleDivision (
            const double & a,
            const double & b)
```

Handler for division. Throws error on divisor = 0.

Parameters

a	
b	

Returns

a/b

4.1.1.5 handleSqrt()

```
double RPN::handleSqrt (
            const double & a)
```

Calculates square roots and errors on negative numbers.

Parameters

a	

Returns

sqrt(a)

4.1.1.6 is1ArgOperator()

```
bool RPN::is1ArgOperator (
            const std::string & op)
```

Checks if given token is an operator that takes only 1 argument, e.g. sqrt(x).

Returns

true if is 1 argument operator.

4.1.1.7 is2ArgOperator()

```
bool RPN::is2ArgOperator (
            const std::string & op)
```

Checks if given token is an operator that takes 2 arguments, e.g. a + b.

Returns

true if is 2 argument operator.

4.1.1.8 isOperator()

```
bool RPN::isOperator (
            const std::string & op)
```

Checks if given string is a valid operator

Parameters

op	

Returns

true if string is an operator

4.1.1.9 sumLetters()

```
int RPN::sumLetters (
            const std::string & str)
```

Intermediate function used by calculate for 1 parameter operators. Sums ascii values of letters to determine which switch case use.

Parameters

str	

Returns

Ascii sum of letters.

4.1.2 Variable Documentation

4.1.2.1 ADD_SUB_PREC

```
int RPN::ADD_SUB_PREC = MULT_DIV_PREC-1  [constexpr]
```

Addition/subtraction precedence score.

4.1.2.2 EXP_PREC

```
int RPN::EXP_PREC = 100  [constexpr]
```

Exponential precedence score.

4.1.2.3 MULT_DIV_PREC

```
int RPN::MULT_DIV_PREC = TRIG_FUN_PREC-1  [constexpr]
```

Multiplication/division precedence score.

4.1.2.4 one_arg_operators

```
const std::unordered_set<std::string> RPN::one_arg_operators
```

Initial value:

```
= {
        "sqrt",
        "cbrt",
        "sin",
        "cos",
        "tan",
    }
```

Operators taking 1 parameter

4.1.2.5 operatorPrecedence

```
const std::map<std::string, int> RPN::operatorPrecedence
```

Initial value:

```
= {
        {"^", EXP_PREC},
        {"sqrt", EXP_PREC},
        {"cbrt", EXP_PREC},
        {"sin", TRIG_FUN_PREC},
        {"cos", TRIG_FUN_PREC},
        {"tan", TRIG_FUN_PREC},
        {"*", MULT_DIV_PREC},
        {"/", MULT_DIV_PREC},
        {"\\", MULT_DIV_PREC},
        {"+", ADD_SUB_PREC},
        {"-", ADD_SUB_PREC},
    }
```

Mapped precedences to operators.

4.1.2.6 TRIG_FUN_PREC

```
int RPN::TRIG_FUN_PREC = EXP_PREC-1  [constexpr]
```

Trigonometric functions precedence score.

4.1.2.7 two_arg_operators

```
const std::unordered_set<std::string> RPN::two_arg_operators
```

Initial value:

```
= {
        "^",
        "*",
        "/",
        "\\",
        "+",
        "-"
    }
```

Operators taking 2 parameters

Chapter 5

Class Documentation

5.1 RPN::NotationConverter Struct Reference

`#include <RPN.h>`

Static Public Member Functions

- static std::string infixToRPN (const std::string &infix)
- static std::string RPNtoInfix (const std::string &RPN)

5.1.1 Detailed Description

Struct able to convert Infix to RPN and vice versa.

5.1.2 Member Function Documentation

5.1.2.1 infixToRPN()

```
std::string RPN::NotationConverter::infixToRPN (
            const std::string & infix)  [static]
```

Given infix equation string, converts it into RPN equation.

Parameters

infix	Infix equation

Returns

RPN equation.

5.1.2.2 RPNtoInfix()

```
std::string RPN::NotationConverter::RPNtoInfix (
            const std::string & RPN)  [static]
```

Given RPN equation string, converts it into infix equation.

Parameters

RPN	RPN equation.

Returns

Infix equation.

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

- lib/RPN.h
- lib/RPN.cpp

5.2 RPN::NotationDeterminer Struct Reference

`#include <RPN.h>`

Static Public Member Functions

- static bool isRPN (const std::string &equation)
- static bool isInfix (const std::string &equation)

5.2.1 Member Function Documentation

5.2.1.1 isInfix()

```
bool RPN::NotationDeterminer::isInfix (
            const std::string & equation)  [static]
```

Determines if string is an Infix equation.

Returns

true if equation is written in Infix.

5.2.1.2 isRPN()

```
bool RPN::NotationDeterminer::isRPN (
            const std::string & equation)  [static]
```

Determines if string is an RPN equation.

Returns

true if equation is written in RPN.

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

- lib/RPN.h
- lib/RPN.cpp

5.3 RPN::RPNSolver Struct Reference

`#include <RPN.h>`

Static Public Member Functions

- static double getResult (const std::string &equation)

5.3.1 Detailed Description

RPN equation solver.

5.3.2 Member Function Documentation

5.3.2.1 getResult()

```
double RPN::RPNSolver::getResult (
            const std::string & equation)  [static]
```

Solves for the result of the RPN equation.

Returns

Result of the RPN equation.

Takes 2 tokens from the stack, removing the first and reassigning the second to the result of the operation.

After the entire algorithm is done the stack should contain only 1 token, which is equal to the result of the equation.

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

- lib/RPN.h
- lib/RPN.cpp

5.4 RPN::TokenReader Struct Reference

`#include <RPN.h>`

Public Member Functions

- TokenReader (const std::string &string)
- std::string getString ()
- std::string next ()
- std::string peek ()
- bool finished () const

5.4.1 Detailed Description

Wrapper over std::stringstream for extracting tokens from the string.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 TokenReader()

```
RPN::TokenReader::TokenReader (
            const std::string & string)  [explicit]
```

Token reader constructor

Parameters

string	Reference to the string from which tokens are read.

5.4.3 Member Function Documentation

5.4.3.1 finished()

```
bool RPN::TokenReader::finished () const
```

Checks if stream came to an end.

Returns

true if stream has finished.

5.4.3.2 getString()

```
std::string RPN::TokenReader::getString ()
```

Returns the entire string from which reader reads.

Returns

Whole string

5.4.3.3 next()

```
std::string RPN::TokenReader::next ()
```

Next tokens in the stream.

Returns

Next token.

5.4.3.4 peek()

```
std::string RPN::TokenReader::peek ()
```

Checks upcoming token in the stream, while keeping the current position in the stream.

Returns

Upcoming token in the stream.

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

- lib/RPN.h
- lib/RPN.cpp

Chapter 6

File Documentation

6.1 build/CMakeFiles/3.30.5/CompilerIdC/CMakeCCompilerId.c File Reference

Macros

- #define __has_include(x)
- #define COMPILER_ID ""
- #define STRINGIFY_HELPER(X)
- #define STRINGIFY(X)
- #define PLATFORM_ID
- #define ARCHITECTURE_ID
- #define DEC(n)
- #define HEX(n)
- #define C_STD_99 199901L
- #define C_STD_11 201112L
- #define C_STD_17 201710L
- #define C_STD_23 202311L
- #define C_VERSION

Functions

- int main (int argc, char *argv[ ])

Variables

- char const * info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
- char const * info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
- char const * info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
- const char * info_language_standard_default
- const char * info_language_extensions_default

6.1.1 Macro Definition Documentation

6.1.1.1 __has_include

```
#define __has_include(
              x)
```

Value:

```
0
```

6.1.1.2 ARCHITECTURE_ID

```
#define ARCHITECTURE_ID
```

6.1.1.3 C_STD_11

```
#define C_STD_11 201112L
```

6.1.1.4 C_STD_17

```
#define C_STD_17 201710L
```

6.1.1.5 C_STD_23

```
#define C_STD_23 202311L
```

6.1.1.6 C_STD_99

```
#define C_STD_99 199901L
```

6.1.1.7 C_VERSION

```
#define C_VERSION
```

6.1.1.8 COMPILER_ID

```
#define COMPILER_ID ""
```

6.1.1.9 DEC

```
#define DEC(
               n)
```

Value:

```
  ('0' + (((n) / 10000000)%10)), \
  ('0' + (((n) / 1000000)%10)),  \
  ('0' + (((n) / 100000)%10)),   \
  ('0' + (((n) / 10000)%10)),    \
  ('0' + (((n) / 1000)%10)),     \
  ('0' + (((n) / 100)%10)),      \
  ('0' + (((n) / 10)%10)),       \
  ('0' +  ((n) % 10))
```

6.1.1.10 HEX

```
#define HEX(
               n)
```

Value:

```
  ('0' + ((n)»28 & 0xF)), \
  ('0' + ((n)»24 & 0xF)), \
  ('0' + ((n)»20 & 0xF)), \
  ('0' + ((n)»16 & 0xF)), \
  ('0' + ((n)»12 & 0xF)), \
  ('0' + ((n)»8  & 0xF)), \
  ('0' + ((n)»4  & 0xF)), \
  ('0' + ((n)     & 0xF))
```

6.1.1.11 PLATFORM_ID

```
#define PLATFORM_ID
```

6.1.1.12 STRINGIFY

```
#define STRINGIFY(
               X)
```

Value:

```
STRINGIFY_HELPER(X)
```

6.1.1.13 STRINGIFY_HELPER

```
#define STRINGIFY_HELPER(
               X)
```

Value:

```
#X
```

6.1.2 Function Documentation

6.1.2.1 main()

```
int main (
             int argc,
             char * argv[])
```

6.1.3 Variable Documentation

6.1.3.1 info_arch

```
char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
```

6.1.3.2 info_compiler

```
char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
```

6.1.3.3 info_language_extensions_default

```
const char* info_language_extensions_default
```

Initial value:

```
= "INFO" ":" "extensions_default["


  "OFF"
"]"
```

6.1.3.4 info_language_standard_default

```
const char* info_language_standard_default
```

Initial value:

```
=
  "INFO" ":" "standard_default[" C_VERSION "]"
```

6.1.3.5 info_platform

```
char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
```

6.2 build/CMakeFiles/3.31.0/CompilerIdC/CMakeCCompilerId.c File Reference

Macros

- #define __has_include(x)
- #define COMPILER_ID ""
- #define STRINGIFY_HELPER(X)
- #define STRINGIFY(X)
- #define PLATFORM_ID
- #define ARCHITECTURE_ID
- #define DEC(n)
- #define HEX(n)
- #define C_STD_99 199901L
- #define C_STD_11 201112L
- #define C_STD_17 201710L
- #define C_STD_23 202311L
- #define C_VERSION

Functions

- int main (int argc, char ∗argv[ ])

Variables

- char const ∗ info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
- char const ∗ info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
- char const ∗ info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
- const char ∗ info_language_standard_default
- const char ∗ info_language_extensions_default

6.2.1 Macro Definition Documentation

6.2.1.1 __has_include

```
#define __has_include(
              x)
```

Value:

```
0
```

6.2.1.2 ARCHITECTURE_ID

```
#define ARCHITECTURE_ID
```

6.2.1.3 C_STD_11

```
#define C_STD_11 201112L
```

6.2.1.4 C_STD_17

```
#define C_STD_17 201710L
```

6.2.1.5 C_STD_23

```
#define C_STD_23 202311L
```

6.2.1.6 C_STD_99

```
#define C_STD_99 199901L
```

6.2.1.7 C_VERSION

```
#define C_VERSION
```

6.2.1.8 COMPILER_ID

```
#define COMPILER_ID ""
```

6.2.1.9 DEC

```
#define DEC(
              n)
```

Value:

```
  ('0' + (((n) / 10000000)%10)), \
  ('0' + (((n) / 1000000)%10)),  \
  ('0' + (((n) / 100000)%10)),   \
  ('0' + (((n) / 10000)%10)),    \
  ('0' + (((n) / 1000)%10)),     \
  ('0' + (((n) / 100)%10)),      \
  ('0' + (((n) / 10)%10)),       \
  ('0' +  ((n) % 10))
```

6.2.1.10 HEX

```
#define HEX(
              n)
```

Value:

```
  ('0' + ((n)»28 & 0xF)), \
  ('0' + ((n)»24 & 0xF)), \
  ('0' + ((n)»20 & 0xF)), \
  ('0' + ((n)»16 & 0xF)), \
  ('0' + ((n)»12 & 0xF)), \
  ('0' + ((n)»8  & 0xF)), \
  ('0' + ((n)»4  & 0xF)), \
  ('0' + ((n)     & 0xF))
```

6.2.1.11 PLATFORM_ID

```
#define PLATFORM_ID
```

6.2.1.12 STRINGIFY

```
#define STRINGIFY(
              X)
```

Value:

```
STRINGIFY_HELPER(X)
```

6.2.1.13 STRINGIFY_HELPER

```
#define STRINGIFY_HELPER(
              X)
```

Value:

```
#X
```

6.2.2 Function Documentation

6.2.2.1 main()

```
int main (
              int argc,
              char * argv[])
```

6.2.3 Variable Documentation

6.2.3.1 info_arch

```
char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
```

6.2.3.2 info_compiler

```
char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
```

6.2.3.3 info_language_extensions_default

```
const char* info_language_extensions_default
```

Initial value:

```
= "INFO" ":" "extensions_default["



  "OFF"
"]"
```

6.2.3.4 info_language_standard_default

```
const char* info_language_standard_default
```

Initial value:

```
=
  "INFO" ":" "standard_default[" C_VERSION "]"
```

6.2.3.5 info_platform

```
char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
```

6.3 build/CMakeFiles/3.30.5/CompilerIdCXX/CMakeCXXCompilerId.cpp File Reference

Macros

- #define __has_include(x)
- #define COMPILER_ID ""
- #define STRINGIFY_HELPER(X)
- #define STRINGIFY(X)
- #define PLATFORM_ID
- #define ARCHITECTURE_ID
- #define DEC(n)
- #define HEX(n)
- #define CXX_STD_98 199711L
- #define CXX_STD_11 201103L
- #define CXX_STD_14 201402L
- #define CXX_STD_17 201703L
- #define CXX_STD_20 202002L
- #define CXX_STD_23 202302L
- #define CXX_STD __cplusplus

Functions

- int main (int argc, char *argv[])

Variables

- char const * info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
- char const * info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
- char const * info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
- const char * info_language_standard_default
- const char * info_language_extensions_default

6.3.1 Macro Definition Documentation

6.3.1.1 __has_include

```
#define __has_include(
              x)
```

Value:

```
0
```

6.3.1.2 ARCHITECTURE_ID

```
#define ARCHITECTURE_ID
```

6.3.1.3 COMPILER_ID

```
#define COMPILER_ID ""
```

6.3.1.4 CXX_STD

```
#define CXX_STD __cplusplus
```

6.3.1.5 CXX_STD_11

```
#define CXX_STD_11 201103L
```

6.3.1.6 CXX_STD_14

```
#define CXX_STD_14 201402L
```

6.3.1.7 CXX_STD_17

```
#define CXX_STD_17 201703L
```

6.3.1.8 CXX_STD_20

```
#define CXX_STD_20 202002L
```

6.3.1.9 CXX_STD_23

```
#define CXX_STD_23 202302L
```

6.3.1.10 CXX_STD_98

```
#define CXX_STD_98 199711L
```

6.3.1.11 DEC

```
#define DEC(
              n)
```

Value:

```
  ('0' + (((n) / 10000000)%10)), \
  ('0' + (((n) / 1000000)%10)),  \
  ('0' + (((n) / 100000)%10)),   \
  ('0' + (((n) / 10000)%10)),    \
  ('0' + (((n) / 1000)%10)),     \
  ('0' + (((n) / 100)%10)),      \
  ('0' + (((n) / 10)%10)),       \
  ('0' +  ((n) % 10))
```

6.3.1.12 HEX

```
#define HEX(
              n)
```

Value:

```
  ('0' + ((n)»28 & 0xF)), \
  ('0' + ((n)»24 & 0xF)), \
  ('0' + ((n)»20 & 0xF)), \
  ('0' + ((n)»16 & 0xF)), \
  ('0' + ((n)»12 & 0xF)), \
  ('0' + ((n)»8  & 0xF)), \
  ('0' + ((n)»4  & 0xF)), \
  ('0' + ((n)     & 0xF))
```

6.3.1.13 PLATFORM_ID

```
#define PLATFORM_ID
```

6.3.1.14 STRINGIFY

```
#define STRINGIFY(
              X)
```

Value:

```
STRINGIFY_HELPER(X)
```

6.3.1.15 STRINGIFY_HELPER

```
#define STRINGIFY_HELPER(
              X)
```

Value:

```
#X
```

6.3.2 Function Documentation

6.3.2.1 main()

```
int main (
             int argc,
             char * argv[])
```

6.3.3 Variable Documentation

6.3.3.1 info_arch

```
char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
```

6.3.3.2 info_compiler

```
char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
```

6.3.3.3 info_language_extensions_default

```
const char* info_language_extensions_default
```

Initial value:

```
= "INFO" ":" "extensions_default["


  "OFF"
"]"
```

6.3.3.4 info_language_standard_default

```
const char* info_language_standard_default
```

Initial value:

```
= "INFO" ":" "standard_default["









  "98"
"]"
```

6.3.3.5 info_platform

```
char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
```

6.4 build/CMakeFiles/3.31.0/CompilerIdCXX/CMakeCXXCompilerId.cpp File Reference

Macros

- #define __has_include(x)
- #define COMPILER_ID ""
- #define STRINGIFY_HELPER(X)
- #define STRINGIFY(X)
- #define PLATFORM_ID
- #define ARCHITECTURE_ID

- #define DEC(n)
- #define HEX(n)
- #define CXX_STD_98 199711L
- #define CXX_STD_11 201103L
- #define CXX_STD_14 201402L
- #define CXX_STD_17 201703L
- #define CXX_STD_20 202002L
- #define CXX_STD_23 202302L
- #define CXX_STD __cplusplus

Functions

- int main (int argc, char *argv[])

Variables

- char const * info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
- char const * info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
- char const * info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
- const char * info_language_standard_default
- const char * info_language_extensions_default

6.4.1 Macro Definition Documentation

6.4.1.1 __has_include

```
#define __has_include(
              x)
```

Value:

```
0
```

6.4.1.2 ARCHITECTURE_ID

```
#define ARCHITECTURE_ID
```

6.4.1.3 COMPILER_ID

```
#define COMPILER_ID ""
```

6.4.1.4 CXX_STD

```
#define CXX_STD __cplusplus
```

6.4.1.5 CXX_STD_11

```
#define CXX_STD_11 201103L
```

6.4.1.6 CXX_STD_14

```
#define CXX_STD_14 201402L
```

6.4.1.7 CXX_STD_17

```
#define CXX_STD_17 201703L
```

6.4.1.8 CXX_STD_20

```
#define CXX_STD_20 202002L
```

6.4.1.9 CXX_STD_23

```
#define CXX_STD_23 202302L
```

6.4.1.10 CXX_STD_98

```
#define CXX_STD_98 199711L
```

6.4.1.11 DEC

```
#define DEC(
              n)
```

Value:

```
  ('0' + (((n) / 10000000)%10)), \
  ('0' + (((n) / 1000000)%10)),  \
  ('0' + (((n) / 100000)%10)),   \
  ('0' + (((n) / 10000)%10)),    \
  ('0' + (((n) / 1000)%10)),     \
  ('0' + (((n) / 100)%10)),      \
  ('0' + (((n) / 10)%10)),       \
  ('0' +  ((n) % 10))
```

6.4.1.12 HEX

```
#define HEX(
              n)
```

Value:

```
  ('0' + ((n)»28 & 0xF)), \
  ('0' + ((n)»24 & 0xF)), \
  ('0' + ((n)»20 & 0xF)), \
  ('0' + ((n)»16 & 0xF)), \
  ('0' + ((n)»12 & 0xF)), \
  ('0' + ((n)»8  & 0xF)), \
  ('0' + ((n)»4  & 0xF)), \
  ('0' + ((n)     & 0xF))
```

6.4.1.13 PLATFORM_ID

```
#define PLATFORM_ID
```

6.4.1.14 STRINGIFY

```
#define STRINGIFY(
              X)
```

Value:

```
STRINGIFY_HELPER(X)
```

6.4.1.15 STRINGIFY_HELPER

```
#define STRINGIFY_HELPER(
              X)
```

Value:

```
#X
```

6.4.2 Function Documentation

6.4.2.1 main()

```
int main (
             int argc,
             char * argv[])
```

6.4.3 Variable Documentation

6.4.3.1 info_arch

```
char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
```

6.4.3.2 info_compiler

```
char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
```

6.4.3.3 info_language_extensions_default

```
const char* info_language_extensions_default
```

Initial value:

```
= "INFO" ":" "extensions_default["



  "OFF"

"]"
```

6.4.3.4 info_language_standard_default

```
const char* info_language_standard_default
```

Initial value:

```
= "INFO" ":" "standard_default["
```

```
  "98"
```

```
"]"
```

6.4.3.5 info_platform

```
char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
```

6.5 build/CMakeFiles/RPN.dir/main.cpp.obj.d File Reference

6.6 build/lib/CMakeFiles/RPN_LIB.dir/RPN.cpp.obj.d File Reference

6.7 lib/RPN.cpp File Reference

```
#include <iostream>
#include <memory>
#include <sstream>
#include <string>
#include <stack>
#include <unordered_set>
#include <climits>
#include <cmath>
#include <map>
#include "RPN.h"
```

Namespaces

- namespace RPN

Functions

- int RPN::sumLetters (const std::string &str)
- double RPN::handleDivision (const double &a, const double &b)
- double RPN::handleSqrt (const double &a)
- double RPN::handleCbrt (const double &a)
- double RPN::calculate (const double &a, const double &b, const std::string &op)
- double RPN::calculate (const double &a, const std::string &op)
- bool RPN::isOperator (const std::string &op)
- bool RPN::is1ArgOperator (const std::string &op)
- bool RPN::is2ArgOperator (const std::string &op)

Variables

- constexpr int RPN::EXP_PREC = 100
- constexpr int RPN::TRIG_FUN_PREC = EXP_PREC-1
- constexpr int RPN::MULT_DIV_PREC = TRIG_FUN_PREC-1
- constexpr int RPN::ADD_SUB_PREC = MULT_DIV_PREC-1
- const std::map< std::string, int > RPN::operatorPrecedence
- const std::unordered_set< std::string > RPN::one_arg_operators
- const std::unordered_set< std::string > RPN::two_arg_operators

6.8 lib/RPN.h File Reference

```
#include <string>
#include <unordered_set>
#include <sstream>
```

Classes

- struct RPN::TokenReader
- struct RPN::RPNSolver
- struct RPN::NotationConverter
- struct RPN::NotationDeterminer

Namespaces

- namespace RPN

6.9 RPN.h

Go to the documentation of this file.

```
00001 #pragma once
00002 #include <string>
00003 #include <unordered_set>
00004 #include <sstream>
00005
00006 namespace RPN {
00010     struct TokenReader {
00015         explicit TokenReader(const std::string& string);
00020         std::string getString();
00025         std::string next();
00031         std::string peek();
00036         bool finished() const;
00037     private:
00038         std::string string_;
00039         std::stringstream stream;
00040     };
00041
00045     struct RPNSolver {
00050         static double getResult(const std::string& equation);
00051     };
00052
00056     struct NotationConverter {
00062         static std::string infixToRPN(const std::string& infix);
00068         static std::string RPNtoInfix(const std::string& RPN);
00069     private:
00078         static std::string wrapInParentheses(const std::string& a, const std::string& b, const
      std::string& op);
00087         static std::string aopb(const std::string &a, const std::string &b, const std::string &op);
00088         static std::string onlyParentheses(const std::string &a);
00089     };
00090
00091     struct NotationDeterminer {
00096         static bool isRPN(const std::string& equation);
00101         static bool isInfix(const std::string& equation);
00102     };
00103 }
```

6.10 main.cpp File Reference

```
#include <iostream>
#include <cstdio>
#include <fstream>
#include <string>
#include "RPN.h"
```

Macros

- #define DEBUG 0

Functions

- void help ()
- void setFlags (const std::string &flags)
- void solveForOutput (const std::string &sourceEquation, std::string &outputEquation, double &result)
- int main (const int argc, char ∗argv[ ])

Variables

- int inputFilePos = -1
- int outputFilePos = -1
- bool isInteractive = false
- bool isRPNOutput = false

6.10.1 Macro Definition Documentation

6.10.1.1 DEBUG

```
#define DEBUG 0
```

6.10.2 Function Documentation

6.10.2.1 help()

```
void help ()
```

Outputs help when executable has no parameters.

6.10.2.2 main()

```
int main (
            const int argc,
            char * argv[])
```

6.10.2.3 setFlags()

```
void setFlags (
            const std::string & flags)
```

Reads flags and configures their values.

Parameters

flags	

6.10.2.4 solveForOutput()

```
void solveForOutput (
            const std::string & sourceEquation,
            std::string & outputEquation,
            double & result)
```

6.10.3 Variable Documentation

6.10.3.1 inputFilePos

```
int inputFilePos = -1
```

6.10.3.2 isInteractive

```
bool isInteractive = false
```

6.10.3.3 isRPNOutput

```
bool isRPNOutput = false
```

6.10.3.4 outputFilePos

```
int outputFilePos = -1
```

Index