mygrep

Создано системой Doxygen 1.8.11

Оглавление

Глава 1

Алфавитный указатель классов

1.1 Классы

Классы с их кратким описанием.

Глава 2

Список файлов

2.1 Файлы

Полный список файлов.

Глава 3

Классы

3.1 Класс LexicalAnalyzer

Лексический анализатор

#include <LexicalAnalyzer.h>

Открытые члены

- LexicalAnalyzer ()
- ~LexicalAnalyzer ()

Открытые статические члены

- static vector< token > analyze (const string &s)

Закрытые статические члены

- static bool isspecial (char ch)
- static string is_num (const string &s)

3.1.1 Подробное описание

Лексический анализатор

Лексический анализатор преобразует исходное регулярное выражение в последовательность лексем, проверяя экранирование специальных символов.

3.1.2 Конструктор(ы)

3.1.2.1 LexicalAnalyzer::LexicalAnalyzer ( ) [inline]

3.1.2.2 LexicalAnalyzer::~LexicalAnalyzer ( ) [inline]

3.1.3 Методы

3.1.3.1 vector< token > LexicalAnalyzer::analyze (const string & s) [static]

Производит лексический анализ регулярного выражения

Аргументы

in	s	Анализируемое выражение

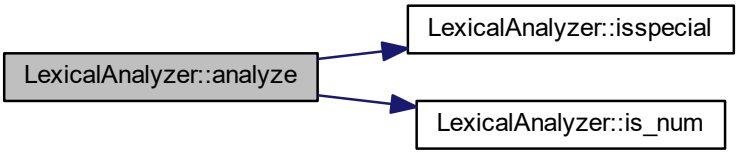
Возвращает

Вектор, составленный из лексем регулярного выражения

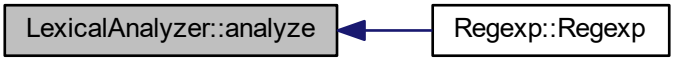
Исключения

std::invalid_argument	В случае некорректных входных данных

Граф вызовов:



Граф вызова функции:



3.1.3.2 string LexicalAnalyzer::is_num (const string & s) [static], [private]

Проверяет, является ли строка числом

Аргументы

in	s	Проверяемая строка

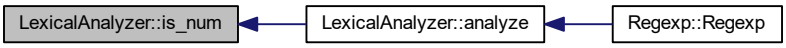
Возвращает

Та же строка

Исключения

std::invalid_argument	В случае, если строка не является числом
std::out_of_range	В случае, если записанное в строке число не может поместиться в тип int

Граф вызова функции:



3.1.3.3 bool LexicalAnalyzer::isspecial (char ch) [static], [private]

Проверяет, является ли символ специальным

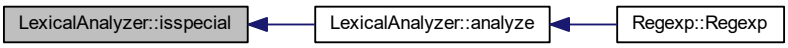
Аргументы

in	ch	Проверяемый символ

Возвращает

Результат проверки

Граф вызова функции:



Объявления и описания членов классов находятся в файлах:

- LexicalAnalyzer.h
- LexicalAnalyzer.cpp

3.2 Структура rc_result

Структура, описывающая формат результата

#include <rc_result.h>

Открытые атрибуты

- bool status

  Результат проверки
- std::string result

  Строка, используемая для хранения найденной подстроки

3.2.1 Подробное описание

Структура, описывающая формат результата

3.2.2 Данные класса

3.2.2.1 std::string rc_result::result

Строка, используемая для хранения найденной подстроки

3.2.2.2 bool rc_result::status

Результат проверки

Объявления и описания членов структуры находятся в файле:

- rc_result.h

3.3 Класс Regexp

Класс регулярного выражения

#include <Regexp.h>

Открытые члены

- Regexp (const string &pattern)
- ~Regexp ()
- bool match (const string &target) const
- rc_result search (const string &target) const

Закрытые данные

- vector< token > sv

  Вектор с преобразованными лексемами

3.3.1 Подробное описание

Класс регулярного выражения

Класс регулярного выражения служит для проверки строк на соответствие регулярному выражению и поиска подстроки, соответствующей регулярному выржению, в заданной строке

3.3.2 Конструктор(ы)

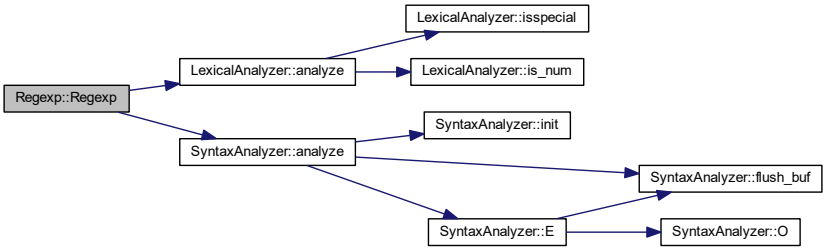
3.3.2.1 Regexp::Regexp (const string & pattern)

Конструктор регулярного выражения

Аргументы

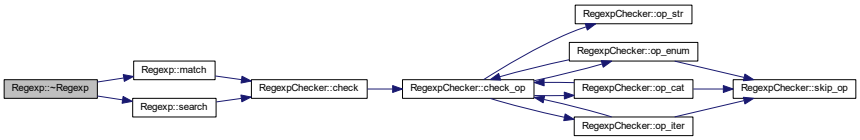
in	pattern	Регулярное выражение в строковом виде

Граф вызовов:



3.3.2.2 Regexp::~Regexp ( ) [inline]

Граф вызовов:



3.3.3 Методы

3.3.3.1 bool Regexp::match (const string & target) const

Проверяет строку на соответствие регулярному выражению

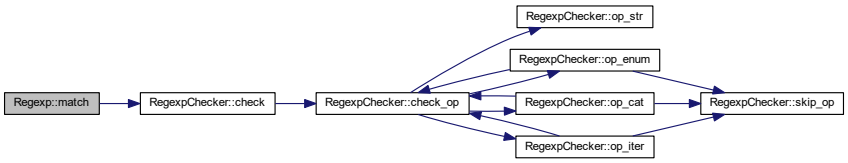
Аргументы

in	target	Строка, которую необходимо проверить

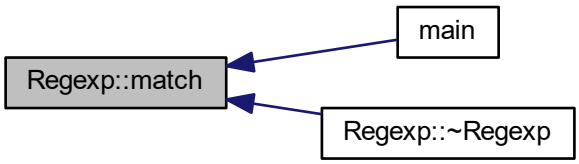
Возвращает

Результат проверки

Граф вызовов:



Граф вызова функции:



3.3.3.2 rc_result Regexp::search (const string & target) const

Ищет подстроку, соответствующую регулярному выражению, в строке

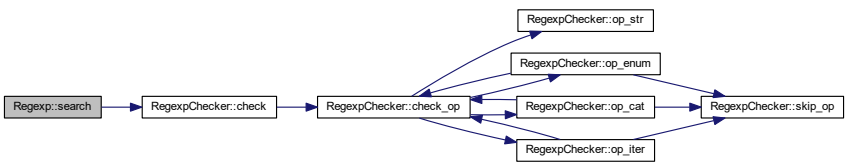
Аргументы

in	target	Строка, в которой производится поиск

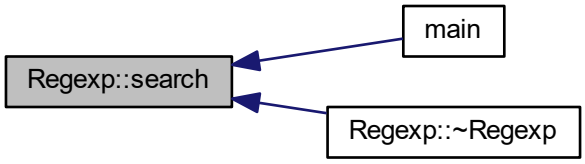
Возвращает

Результат поиска

Граф вызовов:



Граф вызова функции:



3.3.4 Данные класса

3.3.4.1 vector<token> Regexp::sv [private]

Вектор с преобразованными лексемами

Объявления и описания членов классов находятся в файлах:

- Regexp.h
- Regexp.cpp

3.4 Класс RegexpChecker

Класс, проверяющий строку на соответствие регулярному выражению

#include <RegexpChecker.h>

Открытые члены

- RegexpChecker (const vector< token > *v, const string *s, string::const_iterator sit, bool search=false)
- ~RegexpChecker ()
- rc_result check ()

Закрытые члены

- bool check_op ()
- bool op_str ()
- bool op_enum ()
- bool op_cat ()
- bool op_iter (int min=0, int max=-1)
- void skip_op ()

  Пропускает текущую операцию

Закрытые данные

- bool child

  Показывает, является ли даный процесс дочерним
- const vector< token > * sv

  Указатель на вектор с преобразованными лексемами
- vector< token >::const_iterator svit

  Итератор, используемый для обхода вектора лексем
- const string * target

  Указатель на строку, которую необходимо проверить
- const string::const_iterator btit

  Итератор, указывающий на место в строке, с которого начинается проверка
- string::const_iterator tit

  Итератор, используемый для перемещения по строке
- bool search

  Показывает, находится ли класс в режиме поиска подстроки

3.4.1 Подробное описание

Класс, проверяющий строку на соответствие регулярному выражению

Класс последовательно выполняет операции регулярного выражения, с учётом их приоритетов, проверяя таким образом строку на соответствие

3.4.2 Конструктор(ы)

3.4.2.1 RegexpChecker::RegexpChecker (const vector< token > * v, const string * s, string::const_iterator sit, bool search = false)

Конструктор класса

Аргументы

in	v	Указатель на вектор лексем регулярного выражения в префиксной форме
in	s	Указатель на строку, которую необходимо проверить
in	sit	Итератор, который используется для задания начальной позиции для обработки строки
in	search	Определяет режим проверки: сопоставление или поиск

3.4.2.2 RegexpChecker::~RegexpChecker ( ) [inline]

3.4.3 Методы

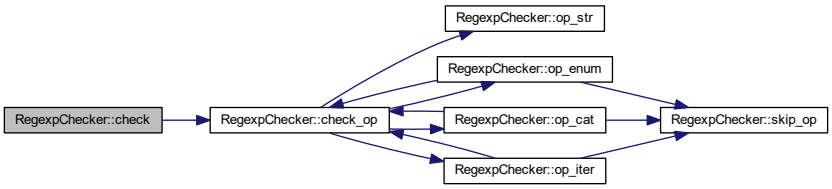
3.4.3.1 rc_result RegexpChecker::check ( )

Запускает проверку

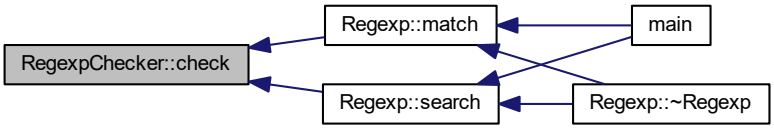
Возвращает

Результат проверки

Граф вызовов:



Граф вызова функции:



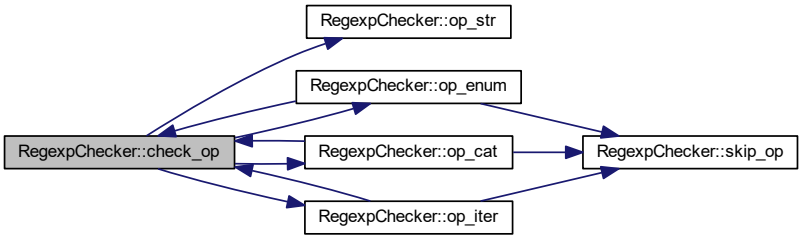
3.4.3.2 bool RegexpChecker::check_op ( ) [private]

Выполняет текующую операцию

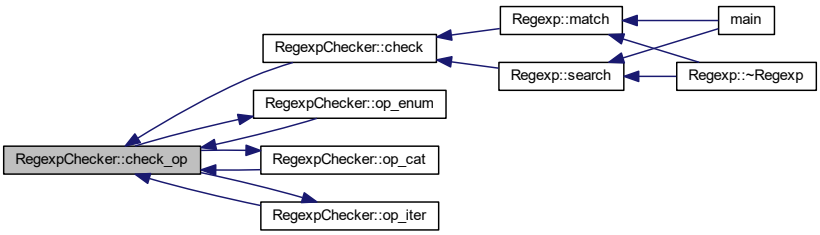
Возвращает

Результат операции

Граф вызовов:



Граф вызова функции:



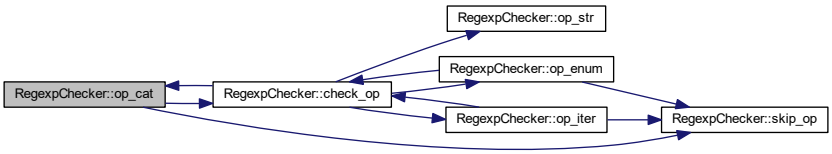
3.4.3.3 bool RegexpChecker::op_cat ( ) [private]

Выполняет операцию конкатенации

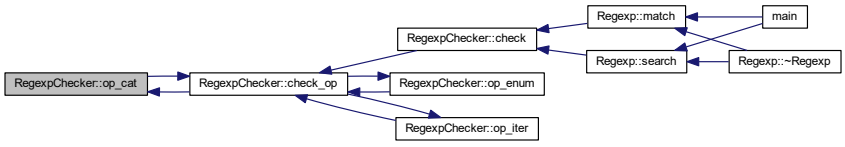
Возвращает

Результат операции

Граф вызовов:



Граф вызова функции:



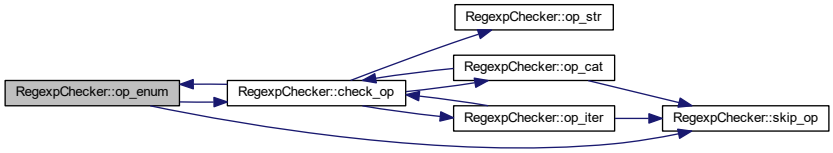
3.4.3.4 bool RegexpChecker::op_enum ( ) [private]

Выполняет операцию перечисления

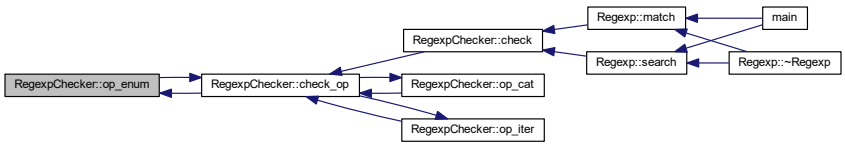
Возвращает

Результат операции

Граф вызовов:



Граф вызова функции:



3.4.3.5 bool RegexpChecker::op_iter (int min = 0, int max = -1) [private]

Выполняет операцию итерирования

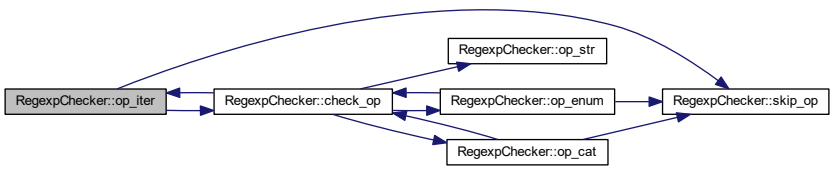
Аргументы

in	min	Минимально необходимое количество итераций
in	max	Максимальное количество итераций

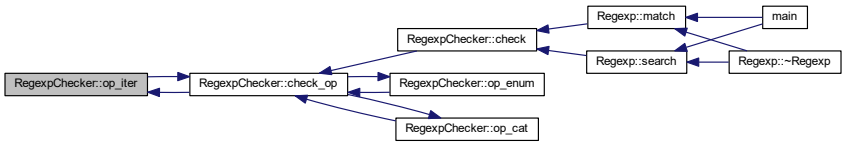
Возвращает

Результат операции

Граф вызовов:



Граф вызова функции:



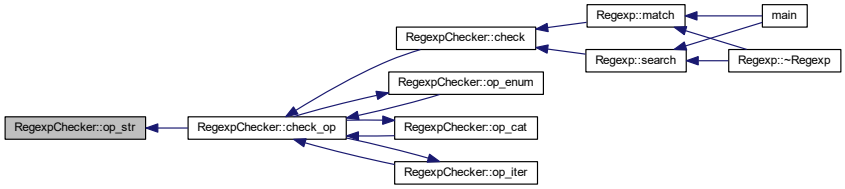
3.4.3.6 bool RegexpChecker::op_str ( ) [private]

Проверяет на соответствие последовательность литералов

Возвращает

Результат операции

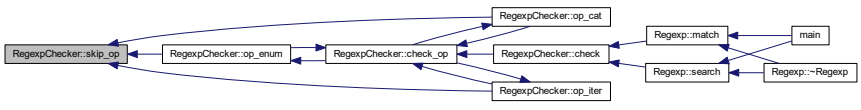
Граф вызова функции:



3.4.3.7 void RegexpChecker::skip_op ( ) [private]

Пропускает текущую операцию

Граф вызова функции:



3.4.4 Данные класса

3.4.4.1 const string::const_iterator RegexpChecker::btit [private]

Итератор, указывающий на место в строке, с которого начинается проверка

3.4.4.2 bool RegexpChecker::child [private]

Показывает, является ли даный процесс дочерним

3.4.4.3 bool RegexpChecker::search [private]

Показывает, находится ли класс в режиме поиска подстроки

3.4.4.4 const vector<token>* RegexpChecker::sv [private]

Указатель на вектор с преобразованными лексемами

3.4.4.5 vector<token>::const_iterator RegexpChecker::svit [private]

Итератор, используемый для обхода вектора лексем

3.4.4.6 const string* RegexpChecker::target [private]

Указатель на строку, которую необходимо проверить

3.4.4.7 string::const_iterator RegexpChecker::tit [private]

Итератор, используемый для перемещения по строке

Объявления и описания членов классов находятся в файлах:

- RegexpChecker.h
- RegexpChecker.cpp

3.5 Класс SyntaxAnalyzer

Синтаксический анализатор

#include <SyntaxAnalyzer.h>

Открытые члены

- SyntaxAnalyzer (const vector< token > &tokens)
- ~SyntaxAnalyzer ()
- vector< token > analyze ()

Закрытые члены

- void init ()

  Инициализирует анализатор
- void E (bool last=false)
- bool O (int pos=-1)
- bool flush_buf (int pos=-1)

Закрытые данные

- vector< token > raw_tokens

  Вектор исходных лексем
- vector< token > pf_tokens

  Вектор с преобразованными лексемами
- vector< token >::iterator it

  Итератор, используемый для обхода вектора исходных лексем
- int brackets_count

  Количество незакрытых скобок
- string buf

  Буфер, используемый для объединения подряд идущих последовательностей литералов

3.5.1 Подробное описание

Синтаксический анализатор

Синтаксический анализатор преобразует последовательность лексем, полученную от лексического анализатора, в префиксную форму, проверяя синтаксис регулярного выражения

3.5.2 Конструктор(ы)

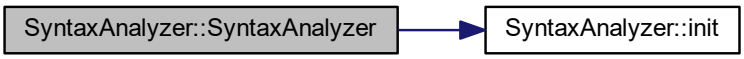
3.5.2.1 SyntaxAnalyzer::SyntaxAnalyzer (const vector< token > & tokens)

Конструктор синтаксического анализатора

Аргументы

in	tokens	Вектор лексем, подлежащий обработке

Граф вызовов:



3.5.2.2 SyntaxAnalyzer::~SyntaxAnalyzer ( ) [inline]

3.5.3 Методы

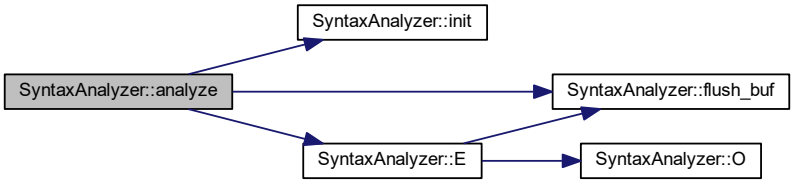
3.5.3.1 vector< token > SyntaxAnalyzer::analyze ( )

Производит синтаксический анализ регулярного выражения

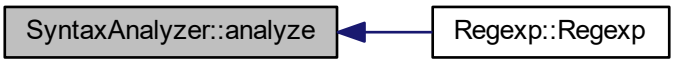
Возвращает

Вектор из лексем, который является префиксной записью регулярного выражения

Граф вызовов:



Граф вызова функции:



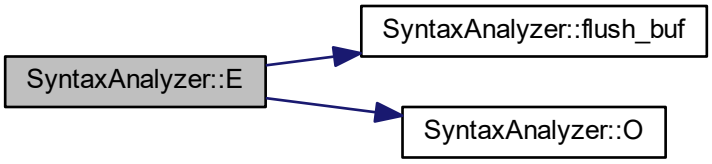
3.5.3.2 void SyntaxAnalyzer::E (bool last = false) [private]

Проверяет выражение и преобразовывает его в префиксную форму

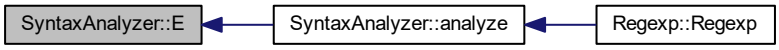
Аргументы

in	last	Показывает, надо ли останавливаться перед проверкой бинарной операции перечисления

Граф вызовов:



Граф вызова функции:



3.5.3.3 bool SyntaxAnalyzer::flush_buf (int pos = -1) [private]

Очищает буфер, занося его содержимое в вектор преобразованных лексем

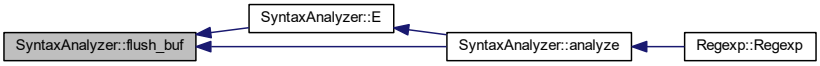
Аргументы

in	pos	Позиция, куда необходимо вставить содержимое буфера

Возвращает

Заполненность буфера на момент вызова

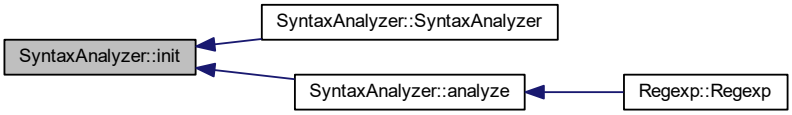
Граф вызова функции:



3.5.3.4 void SyntaxAnalyzer::init ( ) [private]

Инициализирует анализатор

Граф вызова функции:



3.5.3.5 bool SyntaxAnalyzer::O (int pos = -1) [private]

Проверяет наличие операций и добавляет их в вектор лексем

Аргументы

in	pos	Позиция, куда необходимо вставить операции

Возвращает

Наличие операций

Граф вызова функции:

3.5.4 Данные класса

3.5.4.1 int SyntaxAnalyzer::brackets_count [private]

Количество незакрытых скобок

3.5.4.2 string SyntaxAnalyzer::buf [private]

Буфер, используемый для объединения подряд идущих последовательностей литералов

3.5.4.3 vector<token>::iterator SyntaxAnalyzer::it [private]

Итератор, используемый для обхода вектора исходных лексем

3.5.4.4 vector<token> SyntaxAnalyzer::pf_tokens [private]

Вектор с преобразованными лексемами

3.5.4.5 vector<token> SyntaxAnalyzer::raw_tokens [private]

Вектор исходных лексем

Объявления и описания членов классов находятся в файлах:

- SyntaxAnalyzer.h
- SyntaxAnalyzer.cpp

3.6 Структура token

Структура лексемы

#include <token.h>

Открытые атрибуты

- token_type type
  Тип лексемы
- std::string lexeme
  Строковое представление лексемы

3.6.1 Подробное описание

Структура лексемы

3.6.2 Данные класса

3.6.2.1 std::string token::lexeme

Строковое представление лексемы

3.6.2.2 token_type token::type

Тип лексемы

Объявления и описания членов структуры находятся в файле:

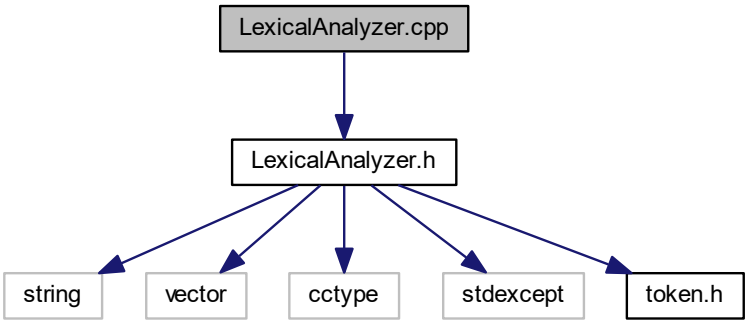
- token.h

Глава 4

Файлы

4.1 Файл LexicalAnalyzer.cpp

#include "LexicalAnalyzer.h"
Граф включаемых заголовочных файлов для LexicalAnalyzer.cpp:

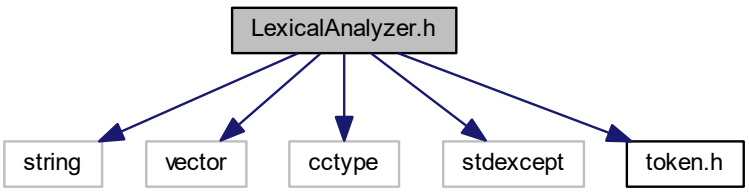


4.2 Файл LexicalAnalyzer.h

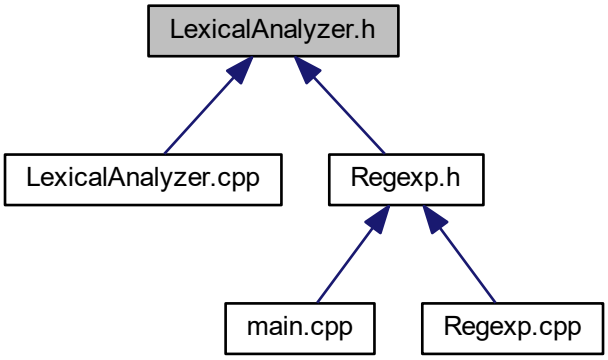
Заголовочный файл лексического анализатора

#include <string>
#include <vector>
#include <cctype>
#include <stdexcept>
#include "token.h"

Граф включаемых заголовочных файлов для LexicalAnalyzer.h:



Граф файлов, в которые включается этот файл:



Классы

- class LexicalAnalyzer

  Лексический анализатор

4.2.1 Подробное описание

Заголовочный файл лексического анализатора

Автор

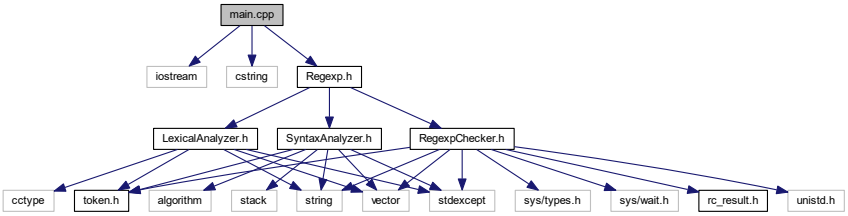
InvalidPointer

Данный файл содержит в себе определение класса лексического анализатора

4.3 Файл main.cpp

```
#include <iostream>
#include <cstring>
#include "Regexp.h"
```

Граф включаемых заголовочных файлов для main.cpp:



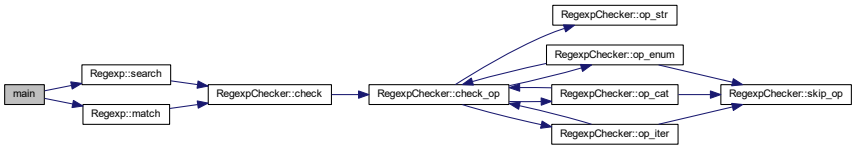
Функции

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

4.3.1 Функции

4.3.1.1 int main (int argc, char ** argv)

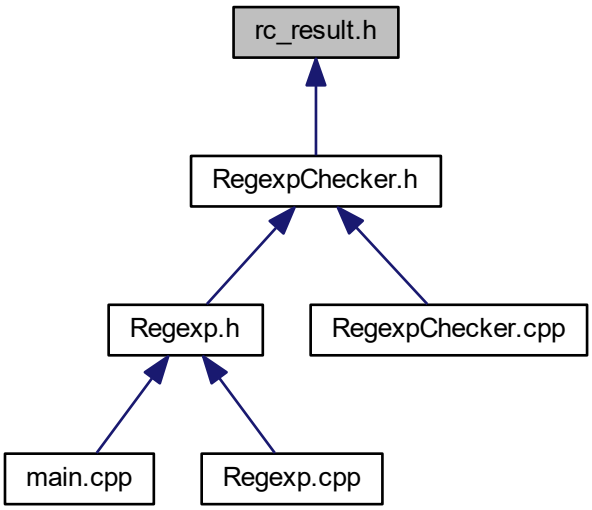
Граф вызовов:



4.4 Файл rc_result.h

Файл с определением результата работы класса проверки строк

Граф файлов, в которые включается этот файл:



Классы

- struct rc_result

  Структура, описывающая формат результата

4.4.1 Подробное описание

Файл с определением результата работы класса проверки строк

Автор

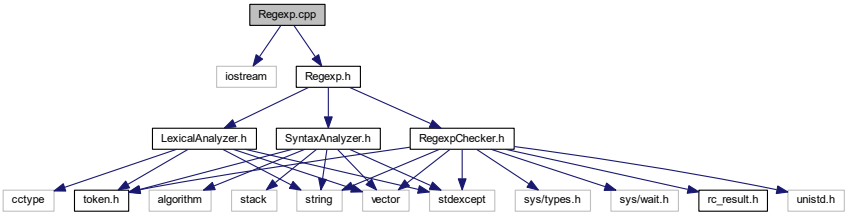
InvalidPointer

Данный файл содержит в себе определение структуры, используемой классом проверки регулярных выражений

4.5 Файл Regexp.cpp

```
#include <iostream>
#include "Regexp.h"
```

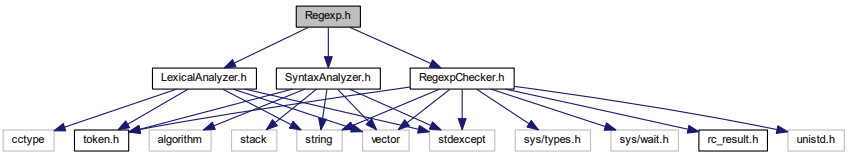
Граф включаемых заголовочных файлов для Regexp.cpp:



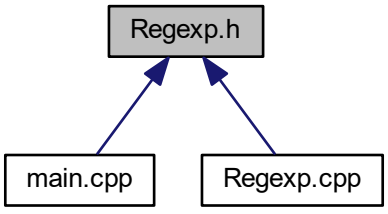
4.6 Файл Regexp.h

Заголовочный файл класса регулярного выражения

#include "LexicalAnalyzer.h"
#include "SyntaxAnalyzer.h"
#include "RegexpChecker.h"
Граф включаемых заголовочных файлов для Regexp.h:



Граф файлов, в которые включается этот файл:



Классы

- class Regexp

  Класс регулярного выражения

4.6.1 Подробное описание

Заголовочный файл класса регулярного выражения

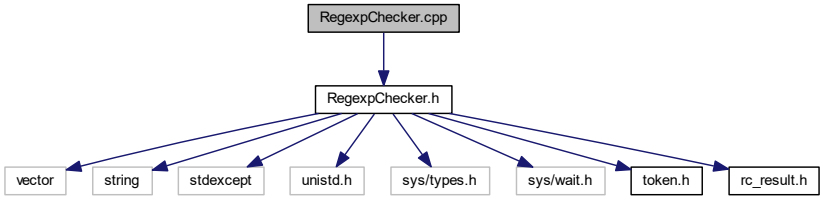
Автор

InvalidPointer

Данный файл содержит в себе определение класса, в котором реализованы обёртки над классом, занимающимся проверкой строк, в виде функций Regexp::match() и Regexp::search()

4.7 Файл RegexpChecker.cpp

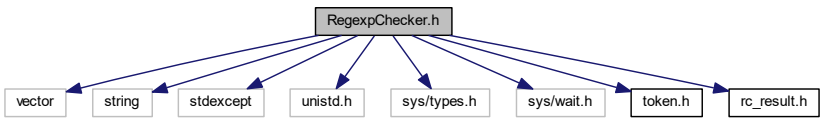
#include "RegexpChecker.h"
Граф включаемых заголовочных файлов для RegexpChecker.cpp:



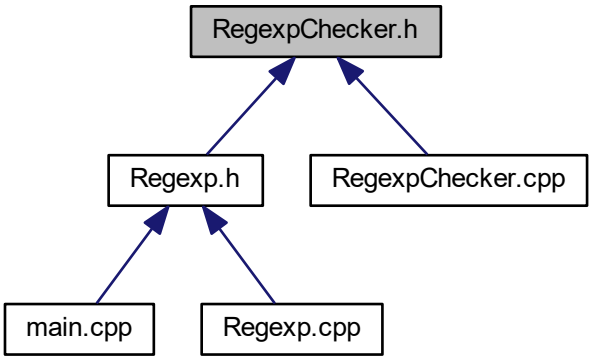
4.8 Файл RegexpChecker.h

Заголовочный файл класса, проверяющего строки

#include <vector>
#include <string>
#include <stdexcept>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include "token.h"
#include "rc_result.h"
Граф включаемых заголовочных файлов для RegexpChecker.h:



Граф файлов, в которые включается этот файл:



Классы

- class RegexpChecker

  Класс, проверяющий строку на соответствие регулярному выражению

4.8.1 Подробное описание

Заголовочный файл класса, проверяющего строки

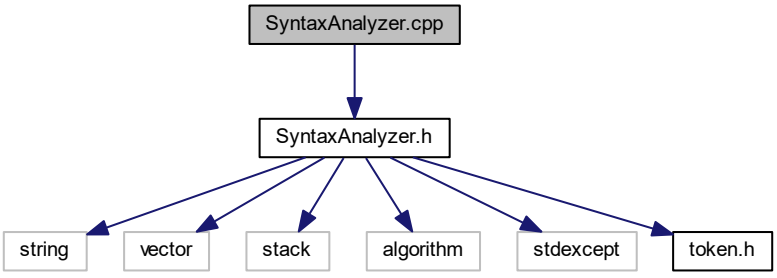
Автор

InvalidPointer

Данный файл содержит в себе определение класса, который реализует механизм проверки соответствия строки с регулярным выражнием

4.9 Файл SyntaxAnalyzer.cpp

#include "SyntaxAnalyzer.h"
Граф включаемых заголовочных файлов для SyntaxAnalyzer.cpp:

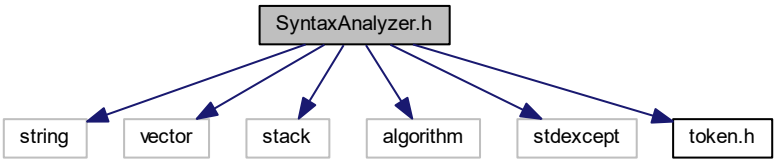


4.10 Файл SyntaxAnalyzer.h

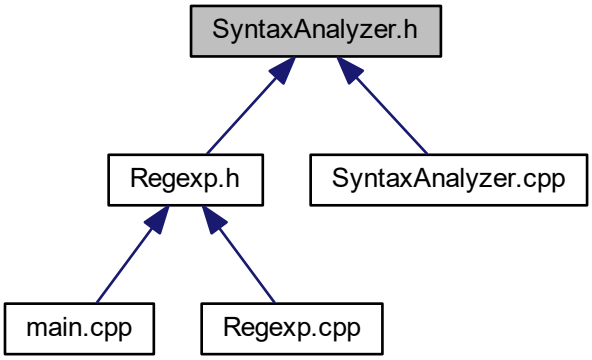
Заголовочный файл синтаксического анализатора

```
#include <string>
#include <vector>
#include <stack>
#include <algorithm>
#include <stdexcept>
#include "token.h"
```

Граф включаемых заголовочных файлов для SyntaxAnalyzer.h:



Граф файлов, в которые включается этот файл:



Классы

- class SyntaxAnalyzer

  Синтаксический анализатор

4.10.1 Подробное описание

Заголовочный файл синтаксического анализатора

Автор

InvalidPointer

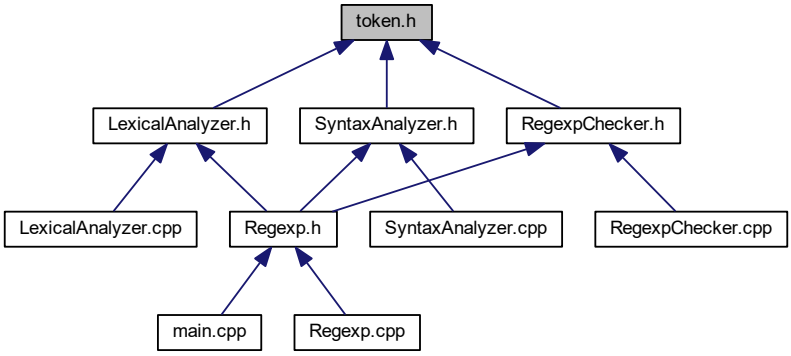
Данный файл содержит в себе определение класса синтаксического анализатора

Грамматика, которую использует анализатор: E -> (E)O | EE | {literal}O | E|E O -> ∗O | {,n}O | {m,}O | _

4.11 Файл token.h

Файл с определением структуры лексемы

Граф файлов, в которые включается этот файл:



Классы

- struct token

  Структура лексемы

Перечисления

- enum token_type {
  O_BR_T, C_BR_T, ENUM_T, STR_T,
  CAT_T, ITER_ZO_T, ITER_OM_T, ITER_ZM_T,
  ITER_N_T }

  Тип лексемы

Переменные

- const int prior [] = {0, 0, 1, 2, 2, 3, 3, 3, 3}

  Приоритеты операций, в порядке описанном выше

4.11.1 Подробное описание

Файл с определением структуры лексемы

Автор

InvalidPointer

Данный файл содержит в себе определение структуры лексемы, их типы и приоритеты

4.11.2 Перечисления

4.11.2.1 enum token_type

Тип лексемы

Элементы перечислений

O_BR_T Открывающая скобка
C_BR_T Закрывающая скобка
ENUM_T Операция перечисления
STR_T Литерал или последовательность литералов
CAT_T Операция конкатенации
ITER_ZO_T Операция итерирования 0 или 1 раз
ITER_OM_T Операция итерирования 1 или более раз
ITER_ZM_T Операция итерирования 0 или более раз
ITER_N_T Операция итерирования

4.11.3 Переменные

4.11.3.1 const int prior[] = {0, 0, 1, 2, 2, 3, 3, 3, 3}

Приоритеты операций, в порядке описанном выше