mygrep

Создано системой Doxygen 1.8.10

Ср 18 Май 2016 04:38:18

Содержание

1 mygrep

Автор

Антон Ханджян, 205 группа ВМК МГУ

2 Алфавитный указатель классов

2.1 Классы

Классы с их кратким описанием.

3 Список файлов

3.1 Файлы

Полный список документированных файлов.

4 Классы

4.1 Класс Lex

Лексический анализатор

#include <lexic.h>

Закрытые члены

- bool isoper (char ch)
  Функция, проверяющая является ли символ оператором
- bool isterm (char ch)
  Функция, проверяющая является ли символ терминальным
- Lex (const string &reg)
  Конструктор от строки с регулярным выражением.
- Lexem get_lex ()
  Функция, последовательно возвращающая лексемы

Закрытые данные

- vector< Lexem > lexems
  Вектор, в который записывается последовательность лексем, эквивалентная исходному регулярному выражению
- unsigned cur = 0
  Текущая позиция в векторе lexems (для функции get_lex())

Закрытые статические данные

- static char possible_op []
  служебный массив символов, имеющих значение операций

Друзья

- class Syntax

4.1.1 Подробное описание

Лексический анализатор

Лексический анализатор преобразует исходную строку в последовательность типизированных лексем и проверяет правильность экранирования символов и написания многосимвольных операторов

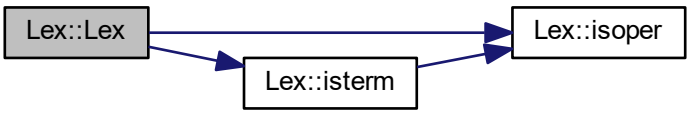
4.1.2 Конструктор(ы)

4.1.2.1 Lex::Lex (const string & reg) [explicit], [private]

Конструктор от строки с регулярным выражением.

Сделан явным, поскольку неясен смысл выражения вида Lex c = "a*";

Граф вызовов:



4.1.3 Методы

4.1.3.1 Lexem Lex::get_lex ( ) [private]

Функция, последовательно возвращающая лексемы

Возвращает

Возвращает очередную лексему

4.1.3.2 bool Lex::isoper (char ch) [private]

Функция, проверяющая является ли символ оператором

Аргументы

in	ch	Символ для проверки

Возвращает

true, если символ является оператором, false — иначе

4.1.3.3 bool Lex::isterm (char ch) [private]

Функция, проверяющая является ли символ терминальным

Аргументы

in	ch	Символ для проверки

Возвращает

true, если символ является терминальным, false — иначе

Граф вызовов:

4.1.4 Данные класса

4.1.4.1 char Lex::possible_op [static], [private]

Инициализатор

```
=
{
    '*',
    '|',
    '(',
    ')',
    '+',
    '?',
    '{',
    '}',
    '[',
    ']',
    '-',
    '\\',
    0
}
```

служебный массив символов, имеющих значение операций

Объявления и описания членов классов находятся в файлах:

- lexic.h
- lexic.cpp

4.2 Класс Lexem

Класс лексем

#include <lexic.h>

Закрытые типы

- enum lexem_types {
  OPEN_BRACE, CLOSE_BRACE, ITER, ITER_N_M,
  LAZY_ITER, TERMINAL, OR, END }

  Возможные типы лексем

Закрытые члены

- Lexem (lexem_types type=END, int param1=0, int param2=0)
- Lexem (string &terminal)
- void print ()

  Функция распечатывает лексемы в исходном виде

Закрытые данные

- string terminal = ""

  Терминальная цепочка
- int param1 = 0

  Первый параметр лексемы
- int param2 = 0

  Второй параметр лексемы
- lexem_types type

  Тип лексемы

Друзья

- class Mygrep
- class Syntax
- class Lex
- class Syn_lexem

4.2.1 Подробное описание

Класс лексем

4.2.2 Перечисления

4.2.2.1 enum Lexem::lexem_types [private]

Возможные типы лексем

Элементы перечислений

OPEN_BRACE Открывающая скобка

CLOSE_BRACE Закрывающая скобка

ITER Итерация

ITER_N_M Итерация от n до m раз

LAZY_ITER Ленивая итерация

TERMINAL Терминальная цепочка

OR Перечисление

END Признак конца

4.2.3 Конструктор(ы)

4.2.3.1 Lexem::Lexem (lexem_types type = END, int param1 = 0, int param2 = 0) [inline], [private]

Конструктор для нетерминальной лексемы

Аргументы

in	type	Тип лексемы. По умолчанию END
in	param1	Первый параметр лексемы. По умолчанию 0
in	param1	Второй параметр лексемы. По умолчанию 0

4.2.3.2 Lexem::Lexem (string & terminal) [inline], [private]

Конструктор для терминальной лексемы

Аргументы

in	terminal	Терминальная цепочка

Объявления и описания членов классов находятся в файлах:

- lexic.h
- lexic.cpp

4.3 Класс Mygrep

Обработчик регулярных выражений

#include <mygrep.h>

Открытые члены

- Mygrep (const string &reg)
  Конструктор от строки с регулярным выражением.
- bool check (const string &s, bool forsearch=false)
  Функция сравнения с регулярным выражением
- bool check (vector< Syn_lexem > &v)
  Функция проверки подвыражения
- string search (const string &s)
  Функция поиска подстроки, соответствующей регулярному выражению
- bool check_next ()
  Проверка следующего

Закрытые типы

- typedef vector< Syn_lexem >::iterator intern_iterator
  Итератор, указывающий на текущее положение во внутреннем представлении

Закрытые члены

- bool selector ()
  Функция выбора операции
- Mygrep (vector< Syn_lexem > &v)
  Конструктор от вектора лексем

Функции-обработчики операций

- void iter (bool lazy=false)
  Итерация
- bool iter_n_m (int n, int m)
  Итерация от n до m раз
- bool terminal ()
  Терминал
- bool alternate ()
  Перечисление

Закрытые данные

- vector< Syn_lexem > internal
  Внутреннее представление регулярного выражения
- intern_iterator internal_it
  Итератор текущего положения в регулярном выражении
- string inp
  Входная строка
- size_t inp_pos = 0
  Позиция во входной строке
- vector< Syn_lexem > v_next
  Вектор синтаксических лексем, следующих за текущей проверяемой лексемой
- bool issearch = false
  Признак режима работы: сравнение или поиск подстроки

4.3.1 Подробное описание

Обработчик регулярных выражений

Проверяет строки на соответствие регулярному выражению либо ищет первое вхождение подстроки, удовлетворяющей регулярному выражению

4.3.2 Конструктор(ы)

4.3.2.1 Mygrep::Mygrep (vector< Syn_lexem > & v) [private]

Конструктор от вектора лексем

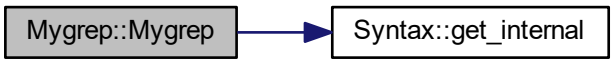
Сделан явным, поскольку неясен смысл выражения вида Mygrep c = "a*";

4.3.2.2 Mygrep::Mygrep (const string & reg) [explicit]

Конструктор от строки с регулярным выражением.

Сделан явным, поскольку неясен смысл выражения вида Mygrep c = "a*";

Граф вызовов:



4.3.3 Методы

4.3.3.1 bool Mygrep::alternate ( ) [private]

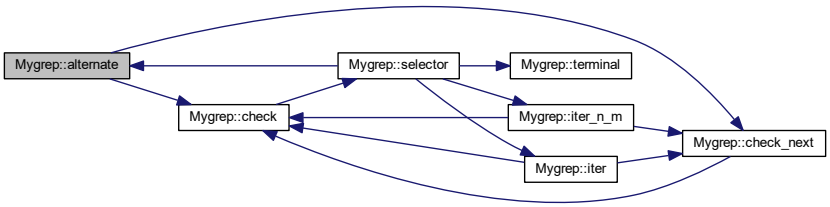
Перечисление

Выполняет проверку соответствия входной строки одному из двух выражений-операндов

Возвращает

true, если соответствует одному из выражений, false, если ни одному из выражений не соответствует

Граф вызовов:



4.3.3.2 bool Mygrep::check (const string & s, bool forsearch = false)

Функция сравнения с регулярным выражением

Проверяет, соответствует ли строка регулярному выражению

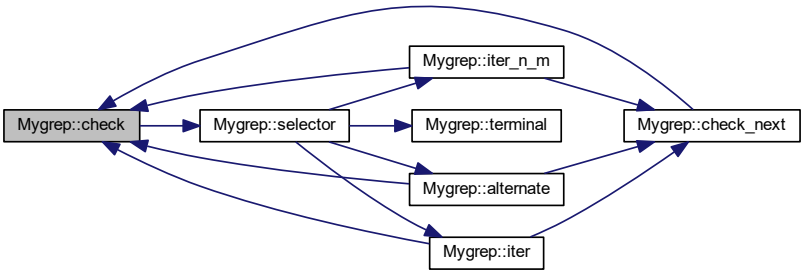
Аргументы

in	s	Строка для проверки
in	forsearch	Признак того, что проверка выполняется для функции search

Возвращает

true, если строка соответствует регулярному выражению, false — иначе

Граф вызовов:



4.3.3.3 bool Mygrep::check (vector< Syn_lexem > & v)

Функция проверки подвыражения

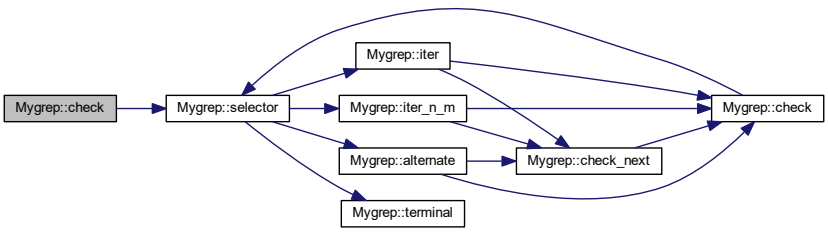
Аргументы

in	v	Вектор лексем для проверки

Возвращает

true, если строка соответствует регулярному выражению, false — иначе

Граф вызовов:



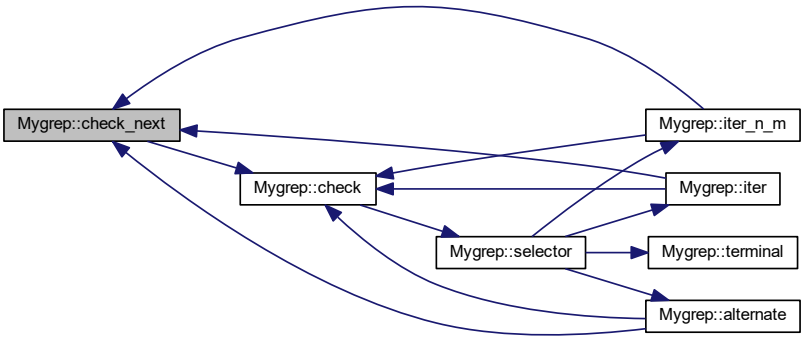
4.3.3.4 bool Mygrep::check_next ( )

Проверка следующего

Возвращает

true, если строка соответствует следующим лексемам, false — иначе

Граф вызовов:



4.3.3.5 void Mygrep::iter (bool lazy = false) [private]

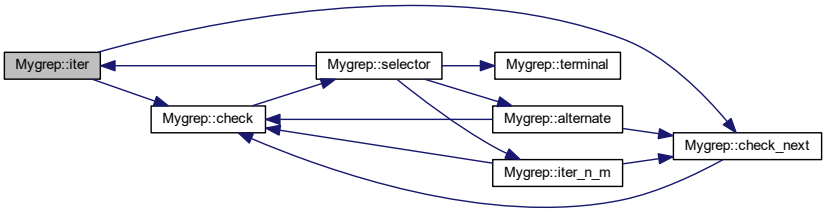
Итерация

Выполняет итерирацию выражения. В режими поиска подстроки выбирает вариант наибольшего количества итераций

Аргументы

in	lazy	Признок ленивости иттерации

Граф вызовов:



4.3.3.6 bool Mygrep::iter_n_m (int n, int m) [private]

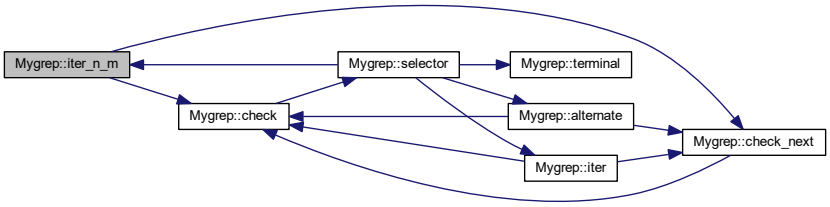
Итерация от n до m раз

Выполняет проверку наличия выражения n раз и выполняет итерацию дополнительно до, возможно, m-того раза

Возвращает

true, если выражение повторяется хотя бы n раз, false — иначе

Граф вызовов:



4.3.3.7 string Mygrep::search (const string & s)

Функция поиска подстроки, соответствующей регулярному выражению

Ищет первое вхождение подстроки, соответствующей регулярному выражению

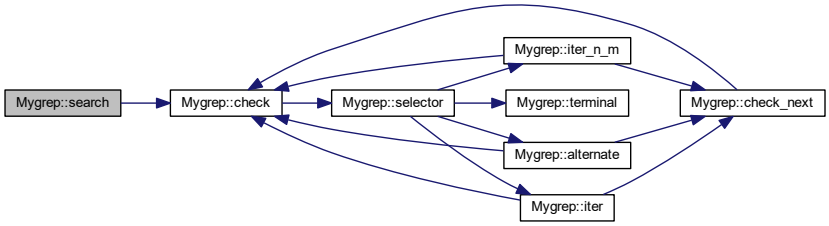
Аргументы

in	s	Строка для поиска

Возвращает

Возвращает подстроку, соответствующую регулярному выражению, либо " " (пробел), если подстрока не найдена

Граф вызовов:



4.3.3.8 bool Mygrep::selector ( ) [private]

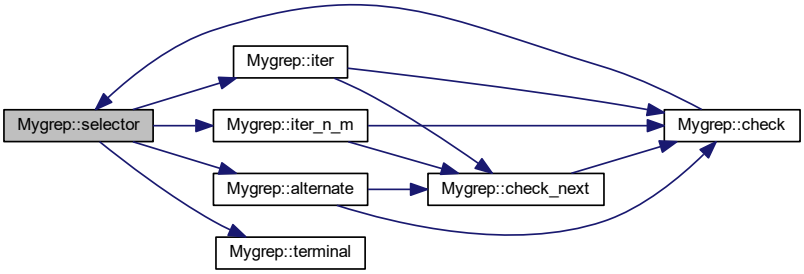
Функция выбора операции

Вызывает функцию, соответствующую текущей лексеме

Возвращает

Возвращает значение вызванной ей функции либо true, если вызывалась неограниченная итерация

Граф вызовов:



4.3.3.9 bool Mygrep::terminal ( ) [private]

Терминал

Выполняет проверку наличия необходимой терминальной последовательности на позиции входной строки, анализируемой в данный момент

Возвращает

true, если терминальная последовательность обнаружена, false — иначе

Объявления и описания членов классов находятся в файлах:

- mygrep.h
- mygrep.cpp

4.4 Класс Syn_lexem

Синтаксические лексемы

#include <syntax.h>

Закрытые члены

- Syn_lexem (Lexem &lexem)
- Syn_lexem (int param2, vector< Syn_lexem > operand)
- void print ()

Закрытые данные

- Lexem::lexem_types type

  Тип лексемы
- string terminal = ""

  Терминальная цепочка
- int param1 = 0

Первый параметр лексемы

- int param2 = 0

  Второй параметр лексемы

- vector< Syn_lexem > operand1

  Первый операнд

- vector< Syn_lexem > operand2

  Второй операнд

Друзья

- class Syntax
- class Mygrep

4.4.1 Подробное описание

Синтаксические лексемы

4.4.2 Конструктор(ы)

4.4.2.1 Syn_lexem::Syn_lexem (Lexem & lexem) [private]

Конструктор построения синтаксической лексемы по обычной

Аргументы

in	lexem	Обычная лексема

4.4.2.2 Syn_lexem::Syn_lexem (int param2, vector< Syn_lexem > operand) [private]

Конструктор лексемы итерация от 0 до param2 раз

Аргументы

in	param2	Максимальное количество итераций
in	operand	Операнд итерации

4.4.3 Методы

4.4.3.1 void Syn_lexem::print ( ) [private]

Печать лексемы

Объявления и описания членов классов находятся в файлах:

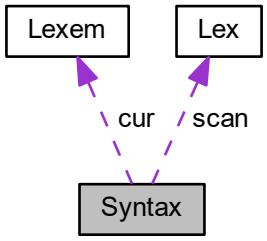
- syntax.h
- syntax.cpp

4.5 Класс Syntax

Синтаксический анализатор

#include <syntax.h>

Граф связей класса Syntax:



Закрытые члены

- void gl ()

  Функция, получающая из лексического анализатора очередную лексему и записывающая её в переменную cur.
- Syntax (const string &reg)

  Конструктор от строки с регулярным выражением.
- vector< Syn_lexem > get_internal ()

  Функция, возвращающая внутреннее представление

Функции РС-метода

- void S ()

  Начальное состояние
- vector< Syn_lexem > S1 (bool concat=false)

  Основное состояние
- vector< Syn_lexem > O (vector< Syn_lexem > &operand)

  Состояние поиска унарного оператора

Закрытые данные

- Lex scan

  Лексический анализатор
- Lexem cur

  Текущая лексема
- int countbrace = 0

  Счётчик скобок
- vector< Syn_lexem > internal

  Внутреннее представление выражения

Друзья

- class Mygrep

4.5.1 Подробное описание

Синтаксический анализатор

Синтаксический анализатор проверяет последовательность типизированных лексем на соответствии грамматике языка регулярных выражений и переводит последотельность во внутреннее представление (последовательность синтаксических лексем)

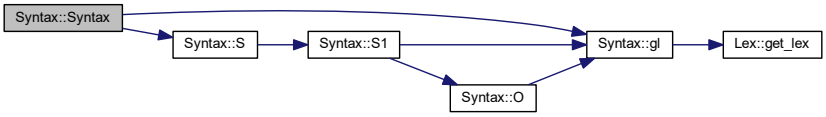
4.5.2 Конструктор(ы)

4.5.2.1 Syntax::Syntax (const string & reg) [explicit], [private]

Конструктор от строки с регулярным выражением.

Сделан явным, поскольку неясен смысл выражения вида Syntax c = "a*";

Граф вызовов:



4.5.3 Методы

4.5.3.1 vector< Syn_lexem > Syntax::get_internal ( ) [private]

Функция, возвращающая внутреннее представление

Возвращает

Возвращают сгенерированное внутреннего представления выражения

4.5.3.2 vector< Syn_lexem > Syntax::O (vector< Syn_lexem > & operand) [private]

Состояние поиска унарного оператора

Возвращает

Возвращают сгенерированную подпоследовательность внутреннего представления

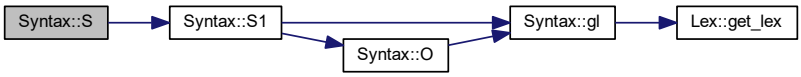
Граф вызовов:

4.5.3.3 void Syntax::S ( ) [private]

Начальное состояние

Либо обнаруживает пустую последовательность, либо переходит в состояние S1

Граф вызовов:



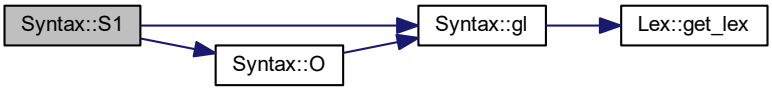
4.5.3.4 vector< Syn_lexem > Syntax::S1 (bool concat = false) [private]

Основное состояние

Возвращает

Возвращают сгенерированную подпоследовательность внутреннего представления

Граф вызовов:



Объявления и описания членов классов находятся в файлах:

- syntax.h
- syntax.cpp

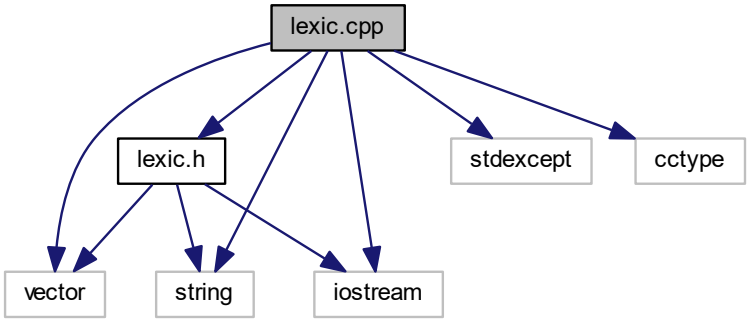
5 Файлы

5.1 Файл lexic.cpp

Файл с реализацией лексического анализатора

```
#include <vector>
#include <string>
#include <iostream>
#include <stdexcept>
#include <cctype>
#include "lexic.h"
```

Граф включаемых заголовочных файлов для lexic.cpp:



5.1.1 Подробное описание

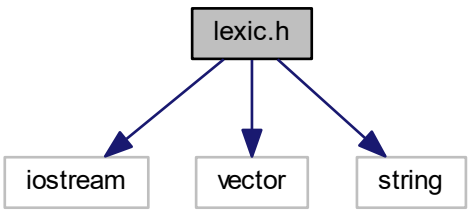
Файл с реализацией лексического анализатора

Данный файл содержит в себе реализацию всех методов классов Lex и Lexem

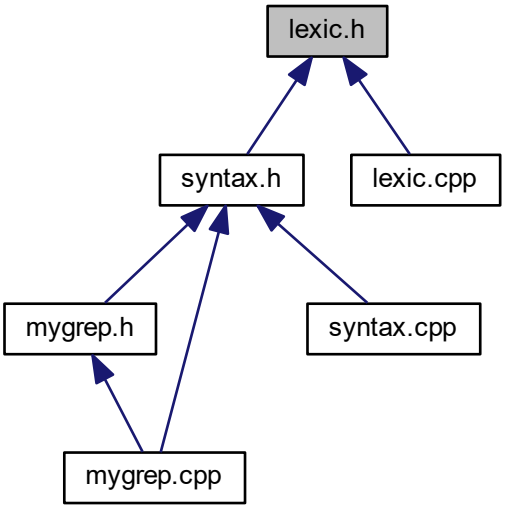
5.2 Файл lexic.h

Заголовочный файл лексического анализатора

#include <iostream>
#include <vector>
#include <string>
Граф включаемых заголовочных файлов для lexic.h:



Граф файлов, в которые включается этот файл:



Классы

- class Lexem

  Класс лексем

- class Lex

  Лексический анализатор

5.2.1 Подробное описание

Заголовочный файл лексического анализатора

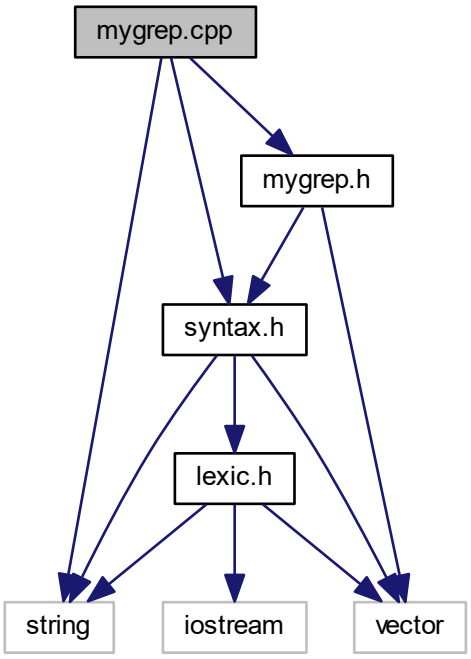
Данный файл содержит в себе определения структуры лексем и класса лексического анализатора

5.3 Файл mygrep.cpp

Файл с реализацией основного функционала программы

```
#include "mygrep.h"
#include "syntax.h"
#include <string>
```

Граф включаемых заголовочных файлов для mygrep.cpp:



5.3.1 Подробное описание

Файл с реализацией основного функционала программы

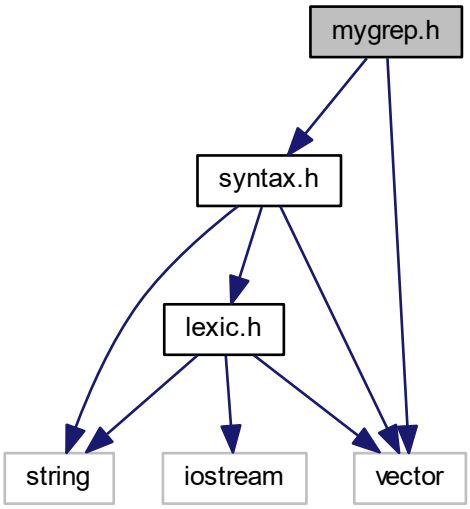
Данный файл содержит в себе реализацию всех методов класса Mygrep

5.4 Файл mygrep.h

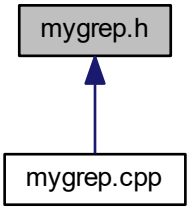
Заголовочный файл пользовательской части программы

```
#include "syntax.h"
#include <vector>
```

Граф включаемых заголовочных файлов для mygrep.h:



Граф файлов, в которые включается этот файл:



Классы

- class Mygrep

  Обработчик регулярных выражений

5.4.1 Подробное описание

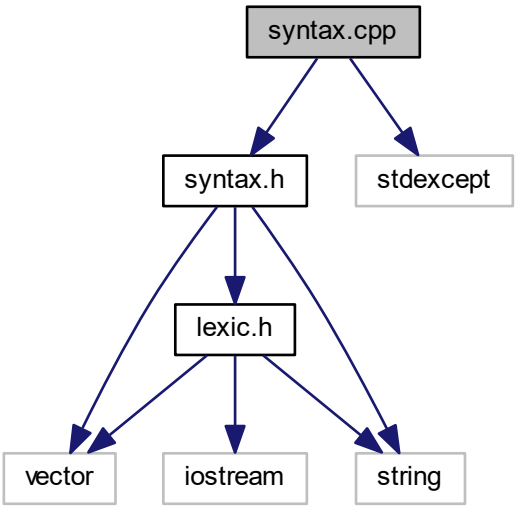
Заголовочный файл пользовательськой части программы

Данный файл содержит в себе определение класса-обработчика регулярных выражений

5.5 Файл syntax.cpp

Файл с реализацией синтаксического анализатора

#include "syntax.h"
#include <stdexcept>
Граф включаемых заголовочных файлов для syntax.cpp:



5.5.1 Подробное описание

Файл с реализацией синтаксического анализатора

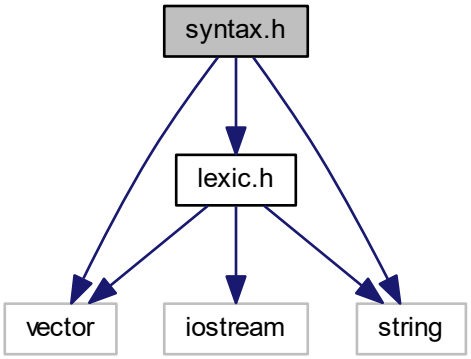
Данный файл содержит в себе реализацию всех методов классов Syntax и Syn_lexem

5.6 Файл syntax.h

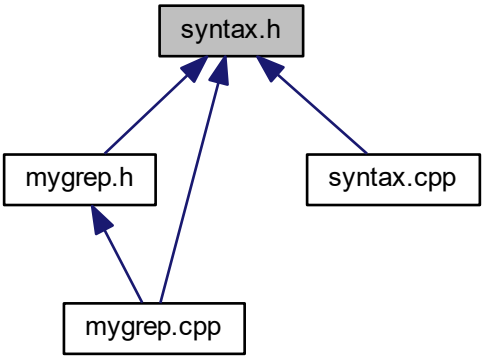
Заголовочный файл синтаксического анализатора

#include "lexic.h"
#include <vector>
#include <string>

Граф включаемых заголовочных файлов для syntax.h:



Граф файлов, в которые включается этот файл:



Классы

- class Syn_lexem

  Синтаксические лексемы
- class Syntax

  Синтаксический анализатор

5.6.1 Подробное описание

Заголовочный файл синтаксического анализатора

Данный файл содержит в себе определения класса синтаксического анализатора

Предметный указатель