минобрнауки россии

федеральное государственное бюджетное

образовательное учреждение высшего образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

|  |  |  |
| --- | --- | --- |
| Институт (факультет) | | Институт информационных технологий |
| Кафедра | Математического и программного обеспечения ЭВМ | |

КУРСОВАЯ РАБОТА

|  |  |
| --- | --- |
| по дисциплине | Модуль: Информатика |

|  |  |
| --- | --- |
| на тему | «Разработка алгоритмического обеспечения и построение лексического и синтаксического анализаторов» |

|  |
| --- |
| Выполнила студент группы |
| 1ПИб-01-1оп-21 |
| направление подготовки (специальности) |
| 09.03.04., Программная инженерия |
| шифр, наименование |
| Фомина Ксения Сергеевна |
| фамилия, имя, отчество |

|  |
| --- |
| Руководитель  Ганичева Оксана Георгиевна |
|  |
| фамилия, имя, отчество  доцент, кандидат наук |
|  |
| должность |

|  |
| --- |
| Дата представления работы |
| « » \_\_\_­­­­­\_\_\_\_\_\_\_\_\_\_\_ 2023 г. |
| Заключение о допуске к защите |
|  |
|  |
|  |
| Оценка \_\_\_\_\_, \_\_\_\_\_\_ количество баллов |
|  |
| Подпись преподавателя \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Череповец, 2023

Оглавление

[1. Введение 3](#_Toc138127092)

[2. Описание предметной области 5](#_Toc138127093)

[3. Основная часть 14](#_Toc138127094)

[4. Логическое проектирование 45](#_Toc138127095)

[5. Физическое проектирование 52](#_Toc138127096)

[6. Проектирование интерфейса 54](#_Toc138127097)

[7. Тестирование программного обеспечения 56](#_Toc138127098)

[8. Заключение. 58](#_Toc138127099)

[9. Список литературы. 59](#_Toc138127100)

[Приложение 1. Техническое задание 60](#_Toc138127101)

[Приложение 2. Текст программы 67](#_Toc138127102)

[Приложение 3. Руководство пользователя 93](#_Toc138127104)

[Приложение 4. Таблица переходов 95](#_Toc138127106)

[Приложение 5. Граф 100](#_Toc138127107)

# 1. Введение

Каждая электронная-вычислительная машина имеет свой машинный язык, язык низкого уровня. Эта машина может исполнять программы, написанные только на этом машинном языке. Любой алгоритм можно написать на этом языке, но подобная задача довольно сложная, потому что для ее выполнения необходимо знать устройство и функционал этой электронно-вычислительной машины. Языки высокого уровня позволяют решить поставленную задачу. Они обладают развитой структурой данных и средствами их обработки. Языки высокого уровня упрощаю работу специалистов и разработчиков.

Язык высокого уровня – это входной язык, описанная на языке высокого уровня – исходная программа.

Для перевода программы с высокоуровневого языка на машинный язык нужен языковой процессор. Языковой процессор – это программа на машинном языке, которая позволяет электронно-вычислительной машине понимать и выполнять программы на входном языке. Языковые процессоры делятся на интерпретаторы и трансляторы.

Интерпретаторы – программа, на вход которой поступает код на входном языке, распознает и реализует конструкции этого входного языка, выдает на выходе результаты вычислений.

Транслятор – программа, которая на выход выдает программу на объектном(машинном) языке эквивалентную программе, которая подавалась на вход.

Компилятор – транслятор для языка высокого уровня.

Компиляция состоит из двух основных этапов: анализ и синтез. Результатом компиляции является объектный код.

Анализ включает в себя лексический, синтаксический и семантический анализы, а синтез – подготовку к генерации, саму генерацию и оптимизацию.

На этапе анализа происходит распознавание текста программы, создание и заполнение таблицы идентификаторов. Т.е. создание некоторого внутреннего представления, понятного компилятору.

Таблица идентификаторов (символов) – специальным образом организованные наборы данных, служащие для хранения информации об элементах исходной программы, которые потом используются для порождения текста результирующей программы.

На этапе синтеза на основании внутреннего представления программы и информации, содержащейся в таблице символов, порождается текс результирующей программы, т.е. объектный код.

# 2. Описание предметной области

В курсовой работе создается модель лексического и синтаксического анализаторов.

Лексический анализ – это первая фаза компиляции, процесс предварительной обработки исходной программы, на котором лексические единицы программы – лексемы приводятся к единому формату и заменяются условными кодами или ссылками на соответствующие таблицы.

Лексема – это структурная единица языка, которая состоит из элементарных символов языка и не содержит в своем составе другие структурные единицы. Лексемами языка программирования являются идентификаторы, константы, ключевые слова, операторы и т.д.

Состав лексем определяется синтаксисом языка.

Цели лексического анализа:

1. Перевод исходной программы на язык, понятный для компилятора, приведение всех лексем к единому формату и замена этих лексем дескрипторами.
2. Лексический контроль – выявление в программе недопустимых слов.

Лексический анализатор – это часть компилятора, которая читает исходную программу, выделяет в ее тексте лексемы.

На выходе информация, которая поступает н вход лексического анализатора – промежуточное представление.

Дескрипторы – это условные коды, на которые заменяются лексемы. Дескрипторы делятся на два типа: числовой и символьный. Дескрипторы выглядят следующим образом: тип\_лексемы, указатель на лексему.

Теоретически лексический анализатор не является обязательной частью компилятора, т.к. его функции могут быть выполнены на этапе синтаксического анализатора. Но все-таки чаще всего лексический анализатор реализуется отдельно. На это есть следующие причины:

1. Отдельно реализованный лексический анализатор упрощает работу с текстом исходной программы на этапе синтаксического анализа, т.к. на этапе лексического анализатора код исходной программы очищается от лишних комментариев, пробелов и переносов строки. После лексического анализа взаимодействовать с текстом проще и удобнее.
2. Увеличение эффективности компилятора. Отдельный лексический анализатор позволяет создать специализированный и потенциально более эффективный процессор для решения поставленной задачи.
3. Увеличение переносимости компилятора. Особенности входного алфавита и другие специфичные характеристики используемых устройств могут ограничивать возможности лексического анализатора.

Набор функций лексического анализатора определяется разработчиком и могут меняться в зависимости от версии компилятора и нужд.

В стандартный набор функций лексического анализатора входят:

1. Удаление лишних пробелов;
2. удаление пустых строк;
3. удаление комментариев;
4. выделение лексем;
5. подсчет строк.

В большинстве компиляторов лексический анализатор и синтаксический анализатор являются взаимосвязанными частями. У них есть несколько вариантов взаимодействия:

1. Последовательное взаимодействие

Лексический анализатор просматривает весь текст исходной программы от начала и до конца, преобразует его структурированный набор данных (таблицу лексем). В таблице лексем все лексемы заменены на специально оговоренные коды.

В этом случае лексический анализатор просматривает текст исходной программы 1 раз и таблица лексем составляется сразу вся, целиком.

Дальнейшая обработка выполняется на следующих фазах компиляции.

1. Параллельное взаимодействие (рис.1)

Лексический анализ исходного текста выполняется поэтапно так, что синтаксический анализ выполняет разбиение очередной конструкции языка, обращается к сканеру за следующей лексемой. При этом он может передавать информацию о том, какую лексему ему необходимо ожидать. При ошибке может происходить «откат» чтобы попытаться выполнить анализ на другой основе.

Только после успешного синтаксического анализа, конструктор лексического анализатора помещает найденные лексемы в таблицу символов и продолжает разбор. Иногда лексический анализатор называют «сканером»

Параллельное взаимодействие анализаторов используется наиболее часто, так как является более эффективным и простым.

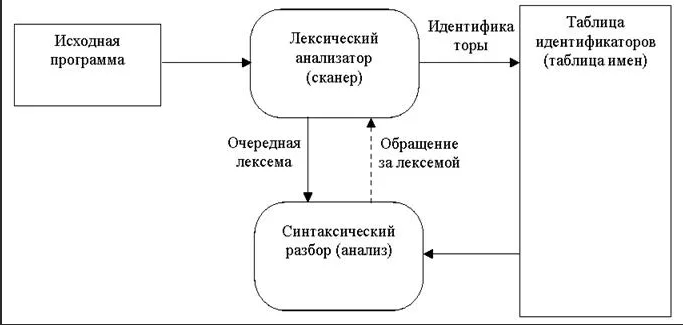


Рис.1. Параллельное взаимодействие лексического и синтаксического анализаторов

Определение границ лексемы – выделение тех строк в общем потоке, для которых нужно выполнить распознавание. Зачастую данная вызывает определенную проблему. В большинстве языков границы лексем задаются терминальными символами, ими могут быть пробелы, запятые, знаки операций, символы комментариев и т.д.

Лексический анализ, как правило, работает по следующему принципу:

1. очередной символ из входного потока данных добавляется в лексему всегда, когда он может быть туда добавлен;
2. как только символ не может быть добавлен в лексему, то считается, что она является границей этой лексемы и началом следующей.

При этом от пользователя требуется обозначить границы лексем пробелами. Данный подход возможен для большинства входных языков.

Способы реализации лексического анализатора:

1. Использование генератора лексических анализаторов (например LEX).

Строится на основе специфик, на основе регулярных выражений. В этом случае входная лексема сравнивается с шаблоном, подготовленным заранее. Генератор предоставляет функции для чтения и буферизации ввода.

1. Написание лексического анализатора на подходящем языке программирования с использованием возможностей ввода-вывода данного языка для чтения входной информации.
2. Написание лексического анализатора на языке ассемблера и явное управление процессом чтения входной информации.

Способы расположены в порядке усложнения их реализации.

Лексический анализатор и конечный автомат:

Разработка лексического анализатора выполняется с использованием математического аппарата, а именно: теории регулярных языков и конечного автомата.

Конечный автомат — математическая абстракция в теории алгоритмов, модель дискретного устройства, в каждый момент времени находящегося в одном состоянии из множества возможных. Является частным случаем абстрактного дискретного автомата, число возможных внутренних состояний которого конечно.

Регулярные выражения – это способ записи, который позволяет точно определить множества слов, принадлежащих языку. Это формула(шаблон), точно показывающая, как было построено соответствующее ей регулярное множество с помощью операций. Эта формула конечна.

Построенная регулярная грамматика является источником, по которому в дальнейшем конструируется вычислительное устройство, реализующее функцию распознавания языка, порождаемого этой грамматикой.

Для регулярных языков таким устройством является конечный автомат. Язык называется автоматным (регулярным) и относится к 3 классу языков.

Формальными порождаемыми грамматиками называются 4 компонента вида:

G = (N, T, P, S), где

N – конечное непустое множество нетерминальных символов, образующих нетерминальный словарь грамматики. Нетерминалы представляют собой переменные, которые обозначают множество строк, которые определяют язык, порожденный грамматикой.

T – конечное не пустое множество терминальных символов, которые образуют терминальный словарь грамматики. Терминал представляют собой базовые символы, из которых формируются строки.

P – множество продукций грамматики. Продукция определяет способ, которым терминалы и нетерминалы могут объединяться для создания строк.

S – начальный/стартовый символ грамматики (аксиома грамматики). Стартовый символ всегда является нетерминалом.

Цепочки символов обозначаются греческими буквами. Цепочка – это некоторая упорядоченная последовательность символов из алфавита нетерминалов и терминалов.

Язык – это множество всех цепочек терминалов, вывод из аксиомы грамматики (язык, выводимый этой грамматикой).

Таким образом грамматика выводит или порождает строки, начиная с аксиомы, неоднократно замещая нетерминалы правыми частями продукций нетерминалов.

Конечный распознаватель – конечный автомат, результатом работы которого является указание на то, допустима ли входная последовательность символов.

Синтаксический анализатор (синтаксический разбор) - это часть компилятора, которая отвечает за выявление основных синтаксических конструкций входного языка.

В задачу синтаксического анализа входит:

1. найти и выделить основные синтаксические конструкции в тексте входной программы,

2. установить тип и проверить правильность каждой синтаксической конструкции,

3. представить синтаксические конструкции в виде, удобном для дальнейшей генерации текста результирующей программы.

В основе синтаксического анализатора лежит распознаватель текста входной программы на основе грамматики входного языка. Как правило, синтаксические конструкции языков программирования могут быть описаны с помощью КС-грамматик, реже встречаются языки, которые могут быть описаны с помощью регулярных грамматик.

Распознаватель дает ответ на вопрос о том, принадлежит или нет цепочка входных символов заданному языку – это основная задача синтаксического анализатора. Кроме этого, синтаксический анализатор должен иметь некий выходной язык, с помощью которого он передает следующим фазам компиляции всю информацию о найденных и разобранных синтаксических структурах.

Синтаксический разбор — это основная часть компилятора на этапе анализа. Без выполнения синтаксического разбора работа компилятора бессмысленна, в то время как лексический разбор в принципе является необязательной фазой. Все задачи по проверке синтаксиса входного языка могут быть решены на этапе синтаксического разбора. Лексический анализатор только позволяет избавить сложный по структуре синтаксический анализатор от решения примитивных задач по выявлению и запоминанию лексем входной программы.

Выходом лексического анализатора является таблица лексем (или цепочка лексем). Эта таблица образует вход синтаксического анализатора, который исследует только один компонент каждой лексемы — ее тип. Остальная информация о лексемах используется на более поздних фазах компиляции при семантическом анализе, подготовке к генерации и генерации кода результирующей программы.

Синтаксический анализатор воспринимает выход лексического анализатора и разбирает его в соответствии с грамматикой входного языка. Однако в грамматике входного языка программирования обычно не уточняется, какие конструкции следует считать лексемами. Примерами конструкций, которые обычно распознаются во время лексического анализа, служат ключевые слова, константы и идентификаторы. Но эти же конструкции могут распознаваться и синтаксическим анализатором. На практике не существует жесткого правила, определяющего, какие конструкции должны распознаваться на лексическом уровне, а какие надо оставлять синтаксическому анализатору. Обычно это определяет разработчик компилятора исходя из технологических аспектов программирования, а также синтаксиса и семантики входного языка.

Основу любого синтаксического анализатора всегда составляет распознаватель, построенный на основе какого-либо класса КС-грамматик. Поэтому главную роль в том, как функционирует синтаксический анализатор и какой алгоритм лежит в его основе, играют принципы построения распознавателей КС-языков. Без применения этих принципов невозможно выполнить эффективный синтаксический разбор предложений входного языка.

Синтаксис конструкций языка программирования может быть описан с помощью контекстно-свободных грамматик или нотации БНФ (форма Бэкуса Наура). Грамматики обеспечивают значительные преимущества разработчикам языков программирования и создателям компиляторов:

• Грамматика дает точную и при этом простую для понимания синтаксическую спецификацию языка программирования;

• Для некоторых классов грамматик можно автоматически построить эффективный синтаксический анализатор, который определяет, корректна ли структура исходной программы;

• Правильно построенная грамматика придает языку программирования структуру, которая способствует облегчению трансляции исходной программы в объектный код, выявлению ошибок.

Со временем языки эволюционируют, обогащаясь новыми конструкциями, и выполняют новые задачи. Добавление конструкций в язык окажется более простой задачей, если существующая реализация языка основана на его грамматическом описании.

# 3. Основная часть

1. Постановка задачи



Рис.2. Вариант задания

Класс - модель для создания объектов определённого типа, описывающая их структуру (набор полей) и определяющая алгоритмы (функции или методы) для работы с этими объектами. Открытая часть класса (обозначается ключевым словом public) чаще всего содержит методы и функции для работы с экземплярами класса называется интерфейсной частью класса.

Основополагающей функцией класса является конструктор. Конструктор - это специальная функция-член класса, объявленная с таким же именем, как и класс. Она предназначена для создания объектов класса и заполнения их полей по умолчанию (конструктор по умолчанию) или в соответствии с параметрами (конструктор с параметрами).

Тестовая программа создает объект класса Student и заполняет поля этого объекта: имя и оценки, которые представляют из себя массивы символов и чисел соответственно. Также программа выводит информацию о созданном объекте в консоль.

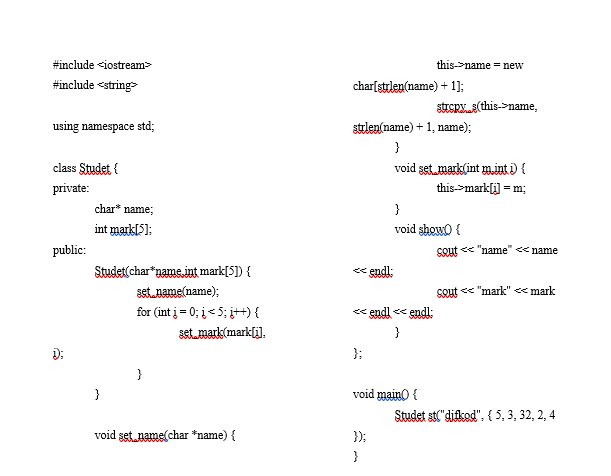


Рис.3. Код тестовой программы

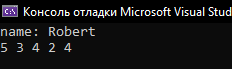


Рис. 4. Результат работы тестовой программы

1. Описание конечного автомата

Конечный автомат (КА) — математическая абстракция, модель дискретного устройства, имеющего один вход, один выход и в каждый момент времени, находящегося в одном состоянии из множества возможных. детерминированные КА — автоматы, в которых следующее состояние однозначно определяется текущим состоянием и входным символом и выход зависит только от текущего состояния и текущего входа.

Конечный автомат задается выражение A = (X, S, F, S0, ∂), где

X – конечное непустое множество входных символов

S – конечное множество состояний

F – множество выделенных заключительных состояний

S0 – начальное состояние автомата

∂ - отображение вида (S, X) -> S

A = (X, S, F, S0, ∂), где

X = {a..z, 0..9, (, ), {, }, [, ], <, >, -, =, +, /, :, ; }

S ={S0}

F = {S116, S217, S119, S201, S22}, где

S116 – ключевые слова;

S217 –идентификаторы;

S119 – константы;

S201 – разделители;

S222 – операции;

∂ = {

(S0, -) -> S1

(S0, 0…9) -> S2

(S0, a) -> S10

(S0, b) -> S29

(S0, c) -> S44

(S0, d) -> S117

(S0, e) -> S118

(S0, f) -> S219

(S0, g) -> S120

(S0, i) -> S121

(S0, l) -> S122

(S0, m) -> S123

(S0, n) -> S124

(S0, o) -> S125

(S0, p) -> S126

(S0, r) -> S127

(S0, s) -> S128

(S0, t) -> S129

(S0, u) -> S130

(S0, v) -> S131

(S0, w) -> S132

(S0, x) -> S133

(S0, <) -> S203

(S0, >) -> S203

(S0, -) -> S203

(S0, =) -> S203

(S0, +) -> S203

(S0, /) -> S203

(S0,!) -> S201

(S0,@) -> S201

(S0,#) -> S201

(S0,$) -> S201

(S0,^) -> S201

(S0,&) -> S201

(S0,() -> S201

(S0,)) -> S201

(S0,{) -> S201

(S0,}) -> S201

(S0,[) -> S201

(S0,]) -> S201

(S0,;) -> S201

(S0,:) -> S201

(S0,,) -> S201

(S0,.) -> S201

(S0,”) -> S201

(S0,’) -> S201

(S1,0…9) -> S2

(S2,0…9) -> S2

(S2, e) -> S3

(S2, l) -> S6

(S2, L) -> S6

(S2, f) -> S6

(S2, F) -> S6

(S2, .) -> S7

(S2, разд.) -> S119

(S3,-) -> S4

(S3,0...9) -> S5

(S4, 0…9) -> S5

(S5, 0…9) -> S5

(S5, разд.) -> S119

(S6, разд.) -> S119

(S7, 0…9) -> S8

(S8, 0…9) -> S8

(S8, e) -> S3

(S8, L|F|f|l) -> S6

(S8, разд.) -> S119

(S10,l) -> S11

(S10,n) -> S12

(S10,s) -> S13

(S10,u) -> S14

(S10, др.) -> S217

(S11, i) -> S15

(S11, др.) -> S217

(S15, g.) -> S16

(S15, др.) -> S217

(S16, n) -> S17

(S16, др.) -> S217

(S17, a) -> S18

(S17, o) -> S19

(S17, др.) -> S217

(S18, s) -> S20

(S18, др.) -> S217

(S20, разд.) -> S116

(S19, f) -> S21

(S19, др.) -> S217

(S21, разд.) -> S116

(S12, d) -> S22

(S12, др.) -> S217

(S22, разд.) -> S116

(S22,\_) -> S23

(S23, e) -> S24

(S23, др.) -> 217

(S24, q) -> S25

(S24, др.) -> 217

(S25, разд.) -> S116

(S13, m) -> S26

(S13, др.) -> 217

(S26, разд.) -> S116

(S14, t) -> S27

(S14, др.) -> 217

(S27, o) -> S28

(S27, др.) -> S217

(S28, разд.) -> S116

(S29,i) -> S30

(S29, o) -> S31

(S29, r) -> S32

(S29, др.) -> S217

(S30, t) -> S33

(S30, др.) -> S217

(S33, a) -> S34

(S33, o) -> S35

(S33, др.) -> S217

(S34, n) -> S36

(S34, др.) -> S217

(S36, d) -> S37

(S36, др.) -> S217

(S37, разд.) -> S116

(S37, r) -> S38

(S37, др.) -> S217

(S38, разд.) -> S116

(S31, o) -> S39

(S31, др.) -> S217

(S39, l) -> S40

(S39, др.) -> S217

(S40, разд.) -> S116

(S32, e) -> S41

(S32, др.) -> S217

(S41, a) -> S42

(S41, др.) -> S217

(S42, k) -> S43

(S42, др.) -> S217

(S43, разд.) -> S116

(S44, a) -> S45

(S44, h) -> S46

(S44, l) -> S47

(S44, o) -> S48

(S44, др.) -> S217

(S45, s) -> S49

(S45, t) -> S50

(S45, др.) -> S217

(S49, e) -> S51

(S49, др.) -> S217

(S51, разд.) -> S116

(S50, c) -> S52

(S50, др.) -> S217

(S52, h) -> S53

(S52, др.) -> S217

(S53, разд.) -> S116

(S46, a) -> S54

(S46, др.) -> S217

(S54, r) -> S55

(S54, др.) -> S217

(S55, разд.) -> S116

(S55, 8) -> S56

(S55, 1) -> S57

(S55, 3) -> S58

(S56, \_) -> S59

(S56, др.) -> S217

(S59, t) -> S60

(S59, др.) -> S217

(S60, разд.) -> S116

(S57, 6) -> S61

(S57, др.) -> S217

(S61, \_) -> S62

(S61, др.) -> S217

(S62, t) -> S63

(S62, др.) -> S217

(S63, разд.) -> S116

(S58, 2) -> S64

(S58, др.) -> S217

(S64, \_) -> S65

(S64, др.) -> S217

(S65, t) -> S66

(S65, др.) -> S217

(S66, разд.) -> S116

(S47, a) -> S67

(S47, др.) -> S217

(S67, s) -> S68

(S67, др.) -> S217

(S68, s) -> S69

(S68, др.) -> S217

(S69, разд.) -> S116

(S48, m) -> S70

(S48, n) -> S71

(S48, \_) -> S72

(S48, др.) -> S217

(S70, p) -> S73

(S70, др.) -> S217

(S73, l) -> S74

(S73, др.) -> S217

(S74, разд.) -> S116

(S71, c) -> S75

(S71, s) -> S76

(S71, n) -> S77

(S71, др.) -> S217

(S75, e) -> S78

(S75, др.) -> S217

(S78, p) -> S79

(S78, др.) -> S217

(S79, t) -> S80

(S79, др.) -> S217

(S80, разд.) -> S116

(S76, t) -> S81

(S76, др.) -> S217

(S81, разд.) -> S116

(S81, \_) -> S82

(S81, e) -> S83

(S81, i.) -> S84

(S82, c) -> S85

(S82, др.) -> S217

(S85, a) -> S86

(S85, др.) -> S217

(S86, s) -> S87

(S86, др.) -> S217

(S87, t) -> S88

(S87, др.) -> S217

(S88, разд.) -> S116

(S83, v) -> S89

(S83, x) -> S90

(S83, др.) -> S217

(S89, a) -> S91

(S89, др.) -> S217

(S91, l) -> S92

(S91, др.) -> S217

(S90, p) -> S93

(S90, др.) -> S217

(S93, r) -> S94

(S93, др.) -> S217

(S94, разд.) -> S116

(S84, n) -> S95

(S84, др.) -> S217

(S95, i) -> S96

(S95, u) -> S97

(S95, др.) -> S217

(S96, t) -> S98

(S96, др.) -> S217

(S98, разд.) -> S116

(S97, e) -> S99

(S97, др.) -> S217

(S99, разд.) -> S116

(S72, a) -> S100

(S72, r) -> S101

(S72, y) -> S102

(S72, др.) -> S217

(S100, w) -> S103

(S100, др.) -> S217

(S103, a) -> S104

(S103, др.) -> S217

(S104, i) -> S105

(S104, др.) -> S217

(S105, t) -> S106

(S105, др.) -> S217

(S106, разд.) -> S116

(S101, e) -> S107

(S101, др.) -> S217

(S107, t) -> S108

(S107, др.) -> S217

(S108, u) -> S109

(S108, др.) -> S217

(S109, r) -> S110

(S109, др.) -> S217

(S110, n) -> S111

(S110, др.) -> S217

(S111, разд.) -> S116

(S102, i) -> S112

(S102, др.) -> S217

(S112, e) -> S113

(S112, др.) -> S217

(S113, l) -> S114

(S102, др.) -> S217

(S114, d) -> S115

(S114, др.) -> S217

(S115, разд.) -> S116

(S117, e) -> S134

(S117, o) -> S135

(S117, др.) -> S217

(S134, c) -> S138

(S134, f) -> S139

(S134, l) -> S140

(S134, др.) -> S217

(S138, l) -> S141

(S138, др.) -> S217

(S141, t) -> S142

(S141, др.) -> S217

(S142, y) -> S143

(S142, др.) -> S217

(S143, p) -> S144

(S143, др.) -> S217

(S144, e) -> S145

(S144, др.) -> S217

(S145, разд.) -> S116

(S146, u) -> S147

(S146, др.) -> S217

(S147, l) -> S148

(S147, др.) -> S217

(S148, t) -> S149

(S148, др.) -> S217

(S149, разд.) -> S116

(S140, e) -> S150

(S140, др.) -> S217

(S150, t) -> S151

(S150, др.) -> S217

(S151, e) -> S152

(S151, др.) -> S217

(S152, разд.) -> S116

(S135, разд.) -> S116

(S135, u) -> S136

(S135, др.) -> S217

(S136, b) -> S153

(S136, др.) -> S217

(S153, l) -> S154

(S153, др.) -> S217

(S154, e) -> S155

(S154, др.) -> S217

(S155, разд.) -> S116

(S118, l) -> S156

(S118, др.) -> S217

(S156, s) -> S157

(S156, др.) -> S217

(S157, e) -> S158

(S157, др.) -> S217

(S158, разд.) -> S116

(S219, a) -> S159

(S219, l) -> S160

(S219, o) -> S161

(S219, др.) -> S217

(S159, l) -> S162

(S159, др.) -> S217

(S162, s) -> S163

(S162, др.) -> S217

(S163, e) -> S164

(S163, др.) -> S217

(S164, разд.) -> S116

(S160, o) -> S165

(S160, др.) -> S217

(S165, a) -> S166

(S165, др.) -> S217

(S166, t) -> S167

(S166, др.) -> S217

(S167, разд.) -> S116

(S161, r) -> S168

(S161, др.) -> S217

(S168, разд.) -> S116

(S120, o) -> S169

(S120, др.) -> S217

(S169, t) -> S170

(S169, др.) -> S217

(S170, o) -> S171

(S170, др.) -> S217

(S171, разд.) -> S116

(S121, f) -> S172

(S121, n) -> S173

(S121, др.) -> S217

(S172, разд.) -> S116

(S173, t) -> S174

(S173, др.) -> S217

(S174, разд.) -> S116

(S122, o) -> S175

(S122, др.) -> S217

(S175, n) -> S176

(S175, др.) -> S217

(S176, g) -> S177

(S176, др.) -> S217

(S177, разд.) -> S116

(S123) -> S217

(S124) -> S217

(S126, r) -> S178

(S126, u) -> S184

(S126, др.) -> S217

(S178, i) -> S179

(S178, др.) -> S217

(S179, v) -> S180

(S179, др.) -> S217

(S180, a) -> S181

(S180, др.) -> S217

(S181, t) -> S182

(S181, др.) -> S217

(S182, e) -> S183

(S182, др.) -> S217

(S183, разд.) -> S116

(S184, b) -> S185

(S184, др.) -> S217

(S185, l) -> S186

(S185, др.) -> S217

(S186, i) -> S187

(S186, др.) -> S217

(S187, c) -> S188

(S187, др.) -> S217

(S188, разд.) -> S116

(S127, u) -> S189

(S127, др.) -> S217

(S189, t) -> S190

(S189, др.) -> S217

(S190, u) -> S191

(S190, др.) -> S217

(S191, r) -> S192

(S191, др.) -> S217

(S192, n) -> S193

(S192, др.) -> S217

(S193, разд.) -> S116

(S125) -> S217

(S128) -> S217

(S129) -> S217

(S130) -> S217

(S131, o) -> S194

(S131, др.) -> S217

(S194, i) -> S195

(S194, др.) -> S217

(S195, d) -> S196

(S195, др.) -> S217

(S196, разд.) -> S116

(S132, h) -> S197

(S132, др.) -> S217

(S197, i) -> S198

(S197, др.) -> S217

(S198, l) -> S199

(S198, др.) -> S217

(S199, e) -> S200

(S199, др.) -> S217

(S200, разд.) -> S116

(S203, +) -> S204

(S203, -) -> S205

(S203, =) -> S206

(S203, \*) -> S207

(S203, /) -> S208

(S203, <) -> S223

(S203, >) -> S224

(S203, др.) -> S201

(S204,+) ->S210

(S204, др.) -> 209

(S205,-) ->S210

(S205, др.) -> 209

(S206,=) ->S210

(S206, др.) -> 209

(S207,разд.) ->S221

(S207, др.) -> 222

(S208,разд.) ->S221

(S208, др.) -> 222

(S133) -> 217

(S209, не буква) -> 221

(S209, др.) -> 222

(S210,не разд.) ->221

(S210, др.) -> 222

(S223, пробел) -> 222

(S223, <) -> 201

(S224, пробел) -> 201

(S224, >) -> 201

}

1. Построение грамматики по конечному автомату

Понятие грамматики изначально было формализовано лингвистами при изучении естественных языков. Предполагалось, что это мюжет помочь при их автоматической трансляции. Однако, наилучшие результаты в этом направлении достигнуты при описании не естественных языков, а языков программирования.

Грамматика это четверка G=(N,T,P,S)

N-афлаваит нетерминальных символов

Нетерминал (нетерминальный символ) — объект, обозначающий какую-либо сущность языка (например: формула, арифметическое выражение, команда) и не имеющий конкретного символьного значения.

Т-афлаваит терминальных символов

Терминал (терминальный символ) — объект, непосредственно присутствующий в словах языка, соответствующего грамматике, и имеющий конкретное, неизменяемое значение (обобщение понятия «буквы»). Базовые символы, из которых формируются строки.

P - конечное множество правил вывода, называемых продукциями.

S - начальный символ грамматики (аксиома языка) (S∈N);

По иерархии Хомского, грамматики делятся на 4 типа, каждый последующий является более ограниченным подмножеством предыдущего (но и легче поддающимся анализу):тип 0. неограниченные грамматики — возможны любые правилатип 1. контекстно-зависимые грамматики — левая часть может содержать один нетерминал, окруженный «контекстом» (последовательности символов, в том же виде присутствующие в правой части); сам нетерминал заменяется непустой последовательностью символов в правой части. Тип 2. контекстно-свободные грамматики — левая часть состоит из одного нетерминала. Тип 3. регулярные грамматики — более простые, эквивалентны конечным автоматам.

Регулярная грамматика, строящаяся на основе конечного автомата, задается следующим образом:

G=(N,T,P,S)

T = {a..z, 0..9, (, ), {, }, [, ], <, >, -, =, +, /, :, ; }

N ={S0…S224}

S=S0

F = {S116, S217, S119, S201, S222}, где

S116 – ключевые слова;

S217 –идентификаторы;

S119 – константы;

S201 – разделители;

S222 – операции;

Отображение продукций грамматики строится по определенным правилам:

* Каждому правилу конечного автомата вида (A, a) -> B должна соответствовать продукция грамматики вида A -> a B;
* Каждому правилу конечного автомата вида (A, a) -> Z, где Z – является конечным состоянием, должна соответствовать продукция грамматики вида A -> a.

Таблица 1.

Формальное описание конечного автомата и регулярной грамматики по заданному конечному автомату

|  |  |
| --- | --- |
| Переход по конечному автомату | Переход по регулярной грамматике |
| (S0, -) -> S1 | S0 -> - S1 |
| (S0, 0…9) -> S2 | S0 -> 0..9 S2 |
| (S0, a) -> S10 | S0 -> a S10 |
| (S0, b) -> S29 | S0 -> b S29 |
| (S0, c) -> S44 | S0 -> c S44 |
| (S0, d) -> S117 | S0 -> d S117 |
| (S0, e) -> S118 | S0 -> e S118 |
| (S0, f) -> S219 | S0 -> f S219 |
| (S0, g) -> S120 | S0 -> g S120 |
| (S0, i) -> S121 | S0 -> i S121 |
| (S0, l) -> S122 | S0 -> l S122 |
| (S0, m) -> S123 | S0 -> m S123 |
| (S0, n) -> S124 | S0 -> n S124 |
| (S0, o) -> S125 | S0 -> o S125 |
| (S0, p) -> S126 | S0 -> p S126 |
| (S0, r) -> S127 | S0 -> r S127 |
| (S0, s) -> S128 | S0 -> s S128 |
| (S0, t) -> S129 | S0 -> t S129 |
| (S0, u) -> S130 | S0 -> u S130 |
| (S0, v) -> S131 | S0 -> v S131 |
| (S0, w) -> S132 | S0 -> w S132 |
| (S0, x) -> S133 | S0 -> x S133 |
| (S0, <) -> S203 | S0 -> < S203 |
| (S0, >) -> S203 | S0 -> > S203 |
| (S0, -) -> S203 | S0 -> - S203 |

Продолжение таблицы 1

|  |  |
| --- | --- |
| Переход по конечному автомату | Переход по регулярной грамматике |
| (S0, =) -> S203 | S0 -> = S203 |
| (S0, +) -> S203 | S0 -> + S203 |
| (S0, /) -> S203 | S0 -> / S203 |
| (S0,!) -> S201 | S0 -> ! |
| (S0,@) -> S201 | S0 -> @ |
| (S0,#) -> S201 | S0 -> # |
| (S0,$) -> S201 | S0 -> $ |
| (S0,^) -> S201 | S0 -> ^ |
| (S0,&) -> S201 | S0 -> & |
| (S0,() -> S201 | S0 -> ( |
| (S0,)) -> S201 | S0 -> ) |
| (S0,{) -> S201 | S0 -> { |
| (S0,}) -> S201 | S0 -> } |
| (S0,[) -> S201 | S0 -> [ |
| (S0,]) -> S201 | S0 -> ] |
| (S0,;) -> S201 | S0 -> ; |
| (S0,:) -> S201 | S0 -> : |
| (S0,,) -> S201 | S0 -> , |
| (S0,.) -> S201 | S0 -> . |
| (S0,”) -> S201 | S0 -> “ |
| (S0,’) -> S201 | S0 -> ‘ |
| (S1,0…9) -> S2 | S1 -> 0..9 S2 |
| (S2,0…9) -> S2 | S2 -> 0..9 S2 |
| (S2, e) -> S3 | S2 -> e S3 |
| (S2, l) -> S6 | S2 -> l S6 |
| (S2, L) -> S6 | S2 -> L S6 |
| (S2, f) -> S6 | S2 -> f S6 |
| (S2, F) -> S6 | S2 -> F S6 |
| (S2, .) -> S7 | S2 -> . S7 |
| (S2, разд.) -> S119 | S2 -> разд. |
| (S3,-) -> S4 | S3 -> - S4 |

Продолжение таблицы 1

|  |  |
| --- | --- |
| Переход по конечному автомату | Переход по регулярной грамматике |
| (S3,0...9) -> S5 | S3 -> 0..9 S5 |
| (S4, 0…9) -> S5 | S4 -> 0..9 S5 |
| (S5, 0…9) -> S5 | S5 -> 0..9 S5 |
| (S5, разд.) -> S119 | S5 -> разд. |
| (S6, разд.) -> S119 | S6 -> разд. |
| (S7, 0…9) -> S8 | S7 -> 0..9 S8 |
| (S8, 0…9) -> S8 | S8 -> 0..9 S8 |
| (S8, e) -> S3 | S8 -> e S3 |
| (S8, L|F|f|l) -> S6 | S8 -> L|F|f|l S6 |
| (S8, разд.) -> S119 | S8 -> разд. |
| (S10,l) -> S11 | S10 -> l S11 |
| (S10,n) -> S12 | S10 -> n S12 |
| (S10,s) -> S13 | S10 -> s S13 |
| (S10,u) -> S14 | S10 -> u S14 |
| (S10, др.) -> S217 | S10 -> др. |
| (S11, i) -> S15 | S11 -> i S15 |
| (S11, др.) -> S217 | S11 -> др. |
| (S15, g.) -> S16 | S15 -> g S16 |
| (S15, др.) -> S217 | S15 -> др. |
| (S16, n) -> S17 | S16 -> n S17 |
| (S16, др.) -> S217 | S16 -> др. |
| (S17, a) -> S18 | S17 -> a S18 |
| (S17, o) -> S19 | S17 -> o S19 |
| (S17, др.) -> S217 | S17 -> др. S218 |
| (S18, s) -> S20 | S18 -> s S20 |
| (S18, др.) -> S217 | S18 -> др. |
| (S20, разд.) -> S116 | S20 -> разд. |
| (S19, f) -> S21 | S19 -> f S21 |
| (S19, др.) -> S217 | S19 -> др. |
| (S21, разд.) -> S116 | S21 -> разд. |
| (S12, d) -> S22 | S12 -> d S22 |

Продолжение таблицы 1

|  |  |
| --- | --- |
| Переход по конечному автомату | Переход по регулярной грамматике |
| (S12, др.) -> S217 | S12 -> др. |
| (S22, разд.) -> S116 | S22 -> разд. |
| (S22,\_) -> S23 | S22 -> S22 |
| (S23, e) -> S24 | S23 -> e S24 |
| (S23, др.) -> 217 | S23 -> др. |
| (S24, q) -> S25 | S24 -> q S25 |
| (S24, др.) -> 217 | S24 -> др. |
| (S25, разд.) -> S116 | S25 -> разд. |
| (S13, m) -> S26 | S24 -> m S26 |
| (S13, др.) -> 217 | S13 -> др. |
| (S26, разд.) -> S116 | S26 -> разд. |
| (S14, t) -> S27 | S14 -> t S27 |
| (S14, др.) -> 217 | S14 -> др. |
| (S27, o) -> S28 | S27 -> o S28 |
| (S27, др.) -> S217 | S27 -> др. |
| (S28, разд.) -> S116 | S28 -> разд. |
| (S29,i) -> S30 | S29 -> i S30 |
| (S29, o) -> S31 | S29 -> o S31 |
| (S29, r) -> S32 | S29 -> r S32 |
| (S29, др.) -> S217 | S29 -> др. |
| (S30, t) -> S33 | S30 -> t S33 |
| (S30, др.) -> S217 | S30 -> др. |
| (S33, a) -> S34 | S33 -> a S34 |
| (S33, o) -> S35 | S33 -> o S35 |
| (S33, др.) -> S217 | S33 -> др. |
| (S34, n) -> S36 | S34 -> n S36 |
| (S34, др.) -> S217 | S34 -> др. |
| (S36, d) -> S35 | S36 -> d S35 |
| (S36, др.) -> S217 | S36 -> др. |
| (S37, разд.) -> S116 | S37 -> разд. |
| (S35, r) -> S38 | S35 -> r S38 |

Продолжение таблицы 1

|  |  |
| --- | --- |
| Переход по конечному автомату | Переход по регулярной грамматике |
| (S35, др.) -> S217 | S35 -> др. |
| (S38, разд.) -> S116 | S38 -> разд. |
| (S31, o) -> S39 | S31 -> o S39 |
| (S31, др.) -> S217 | S31 -> др. |
| (S39, l) -> S40 | S39 -> l S40 |
| (S39, др.) -> S217 | S39 -> др. |
| (S40, разд.) -> S116 | S40 -> разд. |
| (S32, e) -> S41 | S32 -> e S41 |
| (S32, др.) -> S217 | S32 -> др. |
| (S41, a) -> S42 | S41 -> a S42 |
| (S41, др.) -> S217 | S41 -> др. |
| (S42, k) -> S43 | S42 -> k S43 |
| (S42, др.) -> S217 | S42 -> др. |
| (S43, разд.) -> S116 | S43 -> разд. |
| (S44, a) -> S45 | S44 -> a S45 |
| (S44, h) -> S46 | S44 -> h S46 |
| (S44, l) -> S47 | S44 -> l S47 |
| (S44, o) -> S48 | S44 -> o S48 |
| (S44, др.) -> S217 | S44 -> др. |
| (S45, s) -> S49 | S45 -> s S49 |
| (S45, t) -> S50 | S45 -> t S50 |
| (S45, др.) -> S217 | S45 -> др. |
| (S49, e) -> S51 | S49 -> e S51 |
| (S49, др.) -> S217 | S49 -> др. |
| (S51, разд.) -> S116 | S51 -> разд. |
| (S50, c) -> S52 | S50 -> c S52 |
| (S50, др.) -> S217 | S50 -> др. |
| (S52, h) -> S53 | S52 -> h S53 |
| (S52, др.) -> S217 | S52 -> др. |
| (S53, разд.) -> S116 | S53 -> разд. |
| (S46, a) -> S54 | S46 -> a S54 |

Продолжение таблицы 1

|  |  |
| --- | --- |
| Переход по конечному автомату | Переход по регулярной грамматике |
| (S46, др.) -> S217 | S46 -> др. |
| (S54, r) -> S55 | S54 -> r S55 |
| (S54, др.) -> S217 | S54 -> др. |
| (S55, разд.) -> S116 | S55 -> разд. |
| (S55, 8) -> S56 | S55 -> 8 S56 |
| (S55, 1) -> S57 | S55 -> 1 S57 |
| (S55, 3) -> S58 | S55 -> 3 S58 |
| (S56, \_) -> S59 | S56 -> \_ S59 |
| (S56, др.) -> S217 | S56 -> др. |
| (S59, t) -> S60 | S59 -> t S60 |
| (S59, др.) -> S217 | S59 -> др. |
| (S60, разд.) -> S116 | S60 -> разд. |
| (S57, 6) -> S61 | S57 -> 6 S60 |
| (S57, др.) -> S217 | S57 -> др. |
| (S61, \_) -> S62 | S61 -> \_ S62 |
| (S61, др.) -> S217 | S61 -> др. |
| (S62, t) -> S63 | S62 -> t S63 |
| (S62, др.) -> S217 | S62 -> др. |
| (S63, разд.) -> S116 | S63 -> разд. |
| (S58, 2) -> S64 | S58 -> 2 S64 |
| (S58, др.) -> S217 | S58 -> др. |
| (S64, \_) -> S65 | S64 -> \_ S65 |
| (S64, др.) -> S217 | S64 -> др. |
| (S65, t) -> S66 | S65 -> t S66 |
| (S65, др.) -> S217 | S65 -> др. |
| (S66, разд.) -> S116 | S66 -> разд. |
| (S47, a) -> S67 | S47 -> a S67 |
| (S47, др.) -> S217 | S47 -> др. |
| (S67, s) -> S68 | S67 -> s S68 |
| (S67, др.) -> S217 | S67 -> др. |
| (S68, s) -> S69 | S68 -> s S69 |

Продолжение таблицы 1

|  |  |
| --- | --- |
| Переход по конечному автомату | Переход по регулярной грамматике |
| (S68, др.) -> S217 | S68 -> др. |
| (S69, разд.) -> S116 | S69 -> разд. |
| (S48, m) -> S70 | S48 -> m S70 |
| (S48, n) -> S71 | S48 -> n S71 |
| (S48, \_) -> S72 | S48 -> \_ S71 |
| (S48, др.) -> S217 | S48 -> др. |
| (S70, p) -> S73 | S70 -> p S73 |
| (S70, др.) -> S217 | S70 -> др. |
| (S73, l) -> S74 | S73 -> l S74 |
| (S73, др.) -> S217 | S73 -> др. |
| (S74, разд.) -> S116 | S74 -> разд. |
| (S71, c) -> S75 | S71 -> c S75 |
| (S71, s) -> S76 | S71 -> s S76 |
| (S71, n) -> S77 | S71 -> n S77 |
| (S71, др.) -> S217 | S71 -> др. |
| (S75, e) -> S78 | S75 -> e S78 |
| (S75, др.) -> S217 | S75 -> др. |
| (S78, p) -> S79 | S78 -> p S79 |
| (S78, др.) -> S217 | S78 -> др. |
| (S79, t) -> S80 | S79 -> t S80 |
| (S79, др.) -> S217 | S79 -> др. |
| (S80, разд.) -> S116 | S80 -> разд. |
| (S76, t) -> S81 | S76 -> t S81 |
| (S76, др.) -> S217 | S76 -> др. |
| (S81, разд.) -> S116 | S81 -> разд. |
| (S81, \_) -> S82 | S81 -> \_ S82 |
| (S81, e) -> S83 | S81 -> e S83 |
| (S81, i.) -> S84 | S81 -> i S84 |
| (S82, c) -> S85 | S82 -> c S85 |
| (S82, др.) -> S217 | S82 -> др. |
| (S85, a) -> S86 | S85 -> a S86 |

Продолжение таблицы 1

|  |  |
| --- | --- |
| Переход по конечному автомату | Переход по регулярной грамматике |
| (S85, др.) -> S217 | S85 -> др. |
| (S86, s) -> S87 | S86 -> s S87 |
| (S86, др.) -> S217 | S86 -> др. |
| (S87, t) -> S88 | S87 -> t S88 |
| (S87, др.) -> S217 | S87 -> др. |
| (S88, разд.) -> S116 | S88 -> разд. |
| (S83, v) -> S89 | S83 -> v S89 |
| (S83, x) -> S90 | S83 -> x S90 |
| (S83, др.) -> S217 | S83 -> др. |
| (S89, a) -> S91 | S89 -> a S91 |
| (S89, др.) -> S217 | S89 -> др. |
| (S91, l) -> S92 | S91 -> l S92 |
| (S91, др.) -> S217 | S91 -> др. |
| (S90, p) -> S93 | S90 -> p S93 |
| (S90, др.) -> S217 | S90 -> др. |
| (S93, r) -> S94 | S93 -> r S94 |
| (S93, др.) -> S217 | S93 -> др. |
| (S94, разд.) -> S116 | S94 -> разд. |
| (S84, n) -> S95 | S84 -> n S95 |
| (S84, др.) -> S217 | S84 -> др. |
| (S95, i) -> S96 | S95 -> i S96 |
| (S95, u) -> S97 | S95 -> u S97 |
| (S95, др.) -> S217 | S95 -> др. |
| (S96, t) -> S98 | S96 -> t S98 |
| (S96, др.) -> S217 | S96 -> др. |
| (S98, разд.) -> S116 | S98 -> разд. |
| (S97, e) -> S99 | S97 -> e S99 |
| (S97, др.) -> S217 | S97 -> др. |
| (S99, разд.) -> S116 | S99 -> разд. |
| (S72, a) -> S100 | S72 -> a S100 |
| (S72, r) -> S101 | S72 -> r S101 |

Продолжение таблицы 1

|  |  |
| --- | --- |
| Переход по конечному автомату | Переход по регулярной грамматике |
| (S72, y) -> S102 | S72 -> y S102 |
| (S72, др.) -> S217 | S72 -> др. |
| (S100, w) -> S103 | S100 -> w S103 |
| (S100, др.) -> S217 | S100 -> др. |
| (S103, a) -> S104 | S103 -> a S104 |
| (S103, др.) -> S217 | S103 -> др. |
| (S104, i) -> S105 | S104 -> i S105 |
| (S104, др.) -> S217 | S104 -> др. |
| (S105, t) -> S106 | S105 -> t S106 |
| (S105, др.) -> S217 | S105 -> др. |
| (S106, разд.) -> S116 | S106 -> разд. |
| (S101, e) -> S107 | S101 -> e S107 |
| (S101, др.) -> S217 | S101 -> др. |
| (S107, t) -> S108 | S107 -> t S108 |
| (S107, др.) -> S217 | S107 -> др. |
| (S108, u) -> S109 | S108 -> u S109 |
| (S108, др.) -> S217 | S108 -> др. |
| (S109, r) -> S110 | S109 -> r S110 |
| (S109, др.) -> S217 | S109 -> др. |
| (S110, n) -> S111 | S110 -> n S111 |
| (S110, др.) -> S217 | S110 -> др. |
| (S111, разд.) -> S116 | S111 -> разд. |
| (S102, i) -> S112 | S102 -> i S112 |
| (S102, др.) -> S217 | S102 -> др. |
| (S112, e) -> S113 | S112 -> e S113 |
| (S112, др.) -> S217 | S112 -> др. |
| (S113, l) -> S114 | S113 -> l S114 |
| (S113, др.) -> S217 | S113 -> др. |
| (S114, d) -> S115 | S114 -> d S115 |
| (S114, др.) -> S217 | S114 -> др. |
| (S115, разд.) -> S116 | S115 -> разд. |

Продолжение таблицы 1

|  |  |
| --- | --- |
| Переход по конечному автомату | Переход по регулярной грамматике |
| (S117, e) -> S134 | S117 -> e S134 |
| (S117, o) -> S135 | S117 -> o S135 |
| (S117, др.) -> S217 | S117 -> др. |
| (S134, c) -> S138 | S134 -> c S138 |
| (S134, f) -> S139 | S134 -> f S139 |
| (S134, l) -> S140 | S134 -> l S140 |
| (S134, др.) -> S217 | S134 -> др. |
| (S138, l) -> S141 | S138 -> l S141 |
| (S138, др.) -> S217 | S138 -> др. |
| (S141, t) -> S142 | S141 -> t S142 |
| (S141, др.) -> S217 | S141 -> др. |
| (S142, y) -> S143 | S142 -> y S143 |
| (S142, др.) -> S217 | S142 -> др. |
| (S143, p) -> S144 | S143 -> p S144 |
| (S143, др.) -> S217 | S143 -> др. |
| (S144, e) -> S145 | S144 -> e S145 |
| (S144, др.) -> S217 | S144 -> др. |
| (S145, разд.) -> S116 | S145 -> разд. |
| (S146, u) -> S147 | S146 -> u S147 |
| (S146, др.) -> S217 | S146 -> др. |
| (S147, l) -> S148 | S147 -> l S148 |
| (S147, др.) -> S217 | S147 -> др. |
| (S148, t) -> S149 | S148 -> t S149 |
| (S148, др.) -> S217 | S148 -> др. |
| (S149, разд.) -> S116 | S149 -> разд. |
| (S140, e) -> S150 | S140 -> e S150 |
| (S140, др.) -> S217 | S140 -> др. |
| (S150, t) -> S151 | S150 -> t S151 |
| (S150, др.) -> S217 | S150 -> др. |
| (S151, e) -> S152 | S151 -> e S152 |
| (S151, др.) -> S217 | S151 -> др. |

Продолжение таблицы 1

|  |  |
| --- | --- |
| Переход по конечному автомату | Переход по регулярной грамматике |
| (S152, разд.) -> S116 | S152 -> разд. |
| (S135, u) -> S136 | S135 -> u S136 |
| (S135, др.) -> S217 | S135 -> др. |
| (S136, b) -> S153 | S136 -> b S153 |
| (S136, др.) -> S217 | S136 -> др. |
| (S153, l) -> S154 | S153 -> l S154 |
| (S153, др.) -> S217 | S153 -> др. |
| (S154, e) -> S155 | S154 -> e S155 |
| (S154, др.) -> S217 | S154 -> др. |
| (S155, разд.) -> S116 | S155 -> разд. |
| (S118, l) -> S156 | S118 -> l S156 |
| (S118, др.) -> S217 | S118 -> др. |
| (S156, s) -> S157 | S156 -> s S157 |
| (S156, др.) -> S217 | S156 -> др. |
| (S157, e) -> S158 | S157 -> e S158 |
| (S157, др.) -> S217 | S157 -> др. |
| (S158, разд.) -> S116 | S158 -> разд. |
| (S219, a) -> S159 | S219 -> a S159 |
| (S219, l) -> S160 | S219 -> l S160 |
| (S219, o) -> S161 | S219 -> o S161 |
| (S219, др.) -> S217 | S219 -> др. |
| (S159, l) -> S162 | S159 -> l S162 |
| (S159, др.) -> S217 | S159 -> др. |
| (S162, s) -> S163 | S162 -> s S163 |
| (S162, др.) -> S217 | S162 -> др. |
| (S163, e) -> S164 | S163 -> e S164 |
| (S163, др.) -> S217 | S163 -> др. |
| (S164, разд.) -> S116 | S164 -> разд |
| (S160, o) -> S165 | S160 -> o S165 |
| (S160, др.) -> S217 | S160 -> др. |
| (S165, a) -> S166 | S165 -> a S166 |

Продолжение таблицы 1

|  |  |
| --- | --- |
| Переход по конечному автомату | Переход по регулярной грамматике |
| (S165, др.) -> S217 | S165 -> др. |
| (S166, t) -> S167 | S166 -> t S167 |
| (S166, др.) -> S217 | S166 -> др. |
| (S167, разд.) -> S116 | S167 -> разд |
| (S161, r) -> S168 | S161 -> r S168 |
| (S161, др.) -> S217 | S161 -> др. |
| (S168, разд.) -> S116 | S168 -> разд |
| (S120, o) -> S169 | S120 -> o S169 |
| (S120, др.) -> S217 | S120 -> др. |
| (S169, t) -> S170 | S169 -> t S170 |
| (S169, др.) -> S217 | S169 -> др. |
| (S170, o) -> S171 | S170 -> o S171 |
| (S170, др.) -> S217 | S170 -> др. |
| (S171, разд.) -> S116 | S171 -> разд |
| (S121, f) -> S172 | S121 -> f S172 |
| (S121, n) -> S172 | S121 -> n S172 |
| (S121, др.) -> S217 | S121 -> др. |
| (S172, разд.) -> S116 | S121 -> разд |
| (S173, t) -> S174 | S173 -> t S174 |
| (S173, др.) -> S217 | S173 -> др. |
| (S174, разд.) -> S116 | S174 -> разд |
| (S122, o) -> S175 | S122 -> o S175 |
| (S122, др.) -> S217 | S122 -> др. |
| (S175, n) -> S176 | S175 -> n S176 |
| (S175, др.) -> S217 | S175 -> др. |
| (S176, g) -> S177 | S176 -> g S177 |
| (S176, др.) -> S217 | S176 -> др. |
| (S177, разд.) -> S116 | S177 -> разд |
| (S123) -> S217 | (S123) -> S217 |
| (S124) -> S217 | (S124) -> S217 |
| (S126, r) -> S178 | S126 -> r S178 |

Продолжение таблицы 1

|  |  |
| --- | --- |
| Переход по конечному автомату | Переход по регулярной грамматике |
| (S126, u) -> S184 | S126 -> u S184 |
| (S126, др.) -> S217 | S126 -> др. |
| (S178, i) -> S179 | S178 -> i S179 |
| (S178, др.) -> S217 | S178 -> др. |
| (S179, v) -> S180 | S179 -> v S180 |
| (S179, др.) -> S217 | S179 -> др. |
| (S180, a) -> S181 | S180 -> a S181 |
| (S180, др.) -> S217 | S180 -> др. |
| (S181, t) -> S182 | S181 -> t S182 |
| (S181, др.) -> S217 | S181 -> др. |
| (S182, e) -> S183 | S182 -> e S183 |
| (S182, др.) -> S217 | S182 -> др. |
| (S183, разд.) -> S116 | S183 -> разд |
| (S184, b) -> S185 | S184 -> b S185 |
| (S184, др.) -> S217 | S184 -> др. |
| (S185, l) -> S186 | S185 -> l S186 |
| (S185, др.) -> S217 | S185 -> др. |
| (S186, i) -> S187 | S186 -> i S187 |
| (S186, др.) -> S217 | S186 -> др. |
| (S187, c) -> S188 | S187 -> c S188 |
| (S187, др.) -> S217 | S187 -> др. |
| (S188, разд.) -> S116 | S188 -> разд |
| (S127, u) -> S189 | S127 -> u S189 |
| (S127, др.) -> S217 | S127 -> др. |
| (S189, t) -> S190 | S189 -> t S190 |
| (S189, др.) -> S217 | S189 -> др. |
| (S190, u) -> S191 | S190 -> u S191 |
| (S190, др.) -> S217 | S190 -> др. |
| (S191, r) -> S192 | S191 -> r S192 |
| (S191, др.) -> S217 | S191 -> др. |
| (S192, n) -> S193 | S192 -> n S193 |

Продолжение таблицы 1

|  |  |  |
| --- | --- | --- |
| Переход по конечному автомату | | Переход по регулярной грамматике |
| (S192, др.) -> S217 | | S192 -> др. |
| (S193, разд.) -> S116 | | S193 -> разд |
| (S125) -> S217 | | (S125) -> S217 |
| (S128) -> S217 | | (S128) -> S217 |
| (S129) -> S217 | | (S129) -> S217 |
| (S130) -> S217 | | (S130) -> S217 |
| (S131, o) -> S194 | | S131 -> o S194 |
| (S131, др.) -> S217 | | S131 -> др. |
| (S194, i) -> S195 | | S194 -> i S195 |
| (S194, др.) -> S217 | | S194 -> др. |
| (S195, d) -> S196 | | S195 -> d S196 |
| (S195, др.) -> S217 | | S195 -> др. |
| (S196, разд.) -> S116 | | S196 -> разд |
| (S132, h) -> S197 | | S132 -> h S197 |
| (S132, др.) -> S217 | | S132 -> др. |
| (S197, i) -> S198 | | S197 -> i S198 |
| (S197, др.) -> S217 | | S197 -> др. |
| (S198, l) -> S199 | | S198 -> l S199 |
| (S198, др.) -> S217 | | S198 -> др. |
| (S199, e) -> S200 | S199 -> e S200 | |
| (S199, др.) -> S217 | S199 -> др. | |
| (S200, разд.) -> S116 | S200 -> разд | |
| (S203, +) -> S204 | S203 -> + S204 | |
| (S203, -) -> S205 | S203 -> - S204 | |
| (S203, =) -> S206 | S203 -> = S206 | |
| (S203, \*) -> S207 | S203 -> \* S204 | |
| (S203, /) -> S208 | S203 -> / S208 | |
| (S203, <) -> S223 | S203 -> < S223 | |
| (S203, >) -> S224 | S203 -> > S224 | |
| (S203, др.) -> S201 | S203 -> др. | |
| (S204,+) ->S210 | S204 -> + S210 | |

Продолжение таблицы 1

|  |  |
| --- | --- |
| Переход по конечному автомату | Переход по регулярной грамматике |
| (S204, др.) -> S209 | S204 -> др. |
| (S205,-) ->S210 | S205 -> - S210 |
| (S205, др.) -> S209 | S205 -> др. |
| (S206,=) ->S210 | S206 -> = S210 |
| (S206, др.) -> S209 | S206 -> др. |
| (S207,разд.) ->S221 | S207 -> разд |
| (S207, др.) -> S222 | S207 -> др. |
| (S208,разд.) ->S221 | S208 -> разд |
| (S208, др.) -> S222 | S208 -> др. |
| (S133) -> S217 | S133 -> S217 |
| (S209, не буква) -> S221 | S209 -> не буква S221 |
| (S209, др.) -> S222 | S209 -> др. |
| (S210,не разд.) ->S221 | S210 -> не разд |
| (S210, др.) -> S222 | S210 -> др. |
| (S223, пробел) -> S222 | S223 -> пробел |
| (S223, <) -> S201 | S223 -> < |
| (S224, пробел) -> S201 | S224 -> пробел |

Таблица переходов по конечному автомату (приложение 4)

1. Построение грамматики и дерева порождения

Контекстно-свободная грамматика — частный случай формальной грамматики (тип 2 по иерархии Хомского), у которой левые части всех продукций являются одиночными нетерминалами и не имеющими.

Смысл термина «контекстно-свободная» заключается в том, что есть возможность применить продукцию к нетерминалу, причём независимоот контекста этого нетерминала (в отличие от общего случая неограниченной грамматики

Хомского).

Язык, который может быть задан КС-грамматикой, называется контекстно-свободным языком или КС-языком.Язык определяется либо с помощью построения дерева вывода, либо с помощью порождения. В порождении правилами грамматики являются правила, в которых нетерминал из левой части заменяется строкой из правой части продукции. Из стартового символа грамматики выводится строка, содержащая терминальные символы. Такая строка называется предложением грамматики.

Построение порождения выполняется следующим образом: на каждом шаге делается одно из действий:

- выбирается заменяемый нетерминал;

- выбирается продукция для этого терминала.

Деревом вывода является графическим представлением, из которого удалена информация о порядке замещения. Каждый внутренний узел дерева помечается некоторым нетерминалом А, а узлы слева направо – символами из правой части продукции для этого нетерминала. Листья дерева вывода помечены нетерминалами или терминалами и, будучи прочитаны слева направо, образуют сентенциальную форму.

КС-грамматика:

G = {N,T,P,S}

N- множество нетерминальных символов (S, CONSTRUCT, FUNCTION, CLASS, BLOCK,I,T,C,O,P, LOGZNACK,ZNACK,OPERATION)

T – множество терминальных символов (void,int,float,id,private,public,class,{,},[,],const,for,=,++,<, char,\*)

P – множество правил порождений

S – целевой символ грамматики

Правила:

S -> class I{(CLASS)};

CLASS -> private: (O)|public: (FUNCTION)|private: (O) public: (FUNCTION)

O->T I; O|T\* I; O|T I;|T\*I;| T I [C];| T I [C]; O

FUNCTION -> CONSTRUCT FUNCTION|T I()|T I(){BLOCK} FUNCTION |T I(P){BLOCK} FUNCTION | CONSTRUCT| T I(){BLOCK}| T I(P){BLOCK}

CONSTRUCT-> I(P){BLOCK}

P->O | O,P

BLOCK -> I(I); BLOCK|for(T I=C;I LOGZNACK C;I ZNAC){BLOCK}|T I = C;| T I = OPERATION;| I = OPERATION;| I = C;

OPERATION ->C ZNACK C; I ZNACK C; |C ZNACK I;| I ZNACK I; || I ZNACK;||ZNACK I;

T -> int|void|float|double|char

ZNACK -> +|-|\*|/|%|++|--

LOGZNACK -> =|<|>|<=|>=|!=

I -> id | id, I| id[I]|id[I], I

C->const

class Studet {

private:

char\* name;

int mark[5];

public:

Studet(char\*name,int mark[5]) {

set\_name (name);

for (int i = 0; i < 5; i++) {

set\_mark(mark[i], i);

}}

};

S=>class I{**CLASS**};=>

=> class I{private: O public: **FUNCTION**}; =>

=> class I{private: O public: **CONSTRUCT**}; =>

=> class I{private: O public: I(P){**BLOCK**}}; =>

=> class I{private: O public: I(P){ I(I); **BLOCK** }}; =>

=> class I{private: O public: I(P){ I(I); for(T I=C;I LOGZNACK C;I ZNAC){ I(**I**); }}}; =>

=> class I{private: O public: I(P){ I(I); for(T I=C;I LOGZNACK C;I ZNAC){ I(**I**,id1); }}}; =>

=> class I{private: O public: I(P){ I(I); for(T I=C;I LOGZNACK C;I ZNAC){ I(id2[**I**],id1); }}}; =>

=> class I{private: O public: I(P){ I(I); for(T I=C;I LOGZNACK C;I ZNAC){ **I**(id2[id3],id1); }}}; =>

=> class I{private: O public: I(P){ I(I); for(T I=C;I LOGZNACK C;I **ZNAC**){ id4(id2[id3],id1); }}}; =>

=> class I{private: O public: I(P){ I(I); for(T I=C;I LOGZNACK C;**I++**){ id4(id2[id3],id1); }}}; =>

=> class I{private: O public: I(P){ I(I); for(T I=C;I LOGZNACK **C**;id5**++**){ id4(id2[id3],id1); }}}; =>

=> class I{private: O public: I(P){ I(I); for(T I=C;I **LOGZNACK** const1;id5**++**){ id4(id2[id3],id1); }}}; =>

=> class I{private: O public: I(P){ I(I); for(T I=C;**I** **<** const1;id5**++**){ id4(id2[id3],id1); }}}; =>

=> class I{private: O public: I(P){ I(I); for(T I=**C**;**i**d5 **<** const1;id5**++**){ id4(id2[id3],id1); }}}; =>

=> class I{private: O public: I(P){ I(I); for(T **I**=const2;**i**d5 **<** const1;id5**++**){ id4(id2[id3],id1); }}}; =>

=> class I{private: O public: I(P){ I(I); for(**T** id5=const2;**i**d5 **<** const1;id5**++**){ id4(id2[id3],id1); }}}; =>

=> class I{private: O public: I(P){ I(**I**); for(int id5=const2;**i**d5 **<** const1;id5**++**){ id4(id2[id3],id1); }}}; =>

=> class I{private: O public: I(P){ **I**(id6); for(int id5=const2;**i**d5 **<** const1;id5**++**){ id4(id2[id3],id1); }}}; =>

=> class I{private: O public: I(**P**){ id7(id6); for(int id5=const2;**i**d5 **<** const1;id5**++**){ id4(id2[id3],id1); }}}; =>

=> class I{private: O public: I(O,**P**){ id7(id6); for(int id5=const2;**i**d5 **<** const1;id5**++**){ id4(id2[id3],id1); }}}; =>

=> class I{private: O public: I(O,**O**){ id7(id6); for(int id5=const2;**i**d5 **<** const1;id5**++**){ id4(id2[id3],id1); }}}; =>

=> class I{private: O public: I(O,T I[**C]**){ id7(id6); for(int id5=const2;**i**d5 **<** const1;id5**++**){ id4(id2[id3],id1); }}}; =>

=> class I{private: O public: I(O,T **I**[const1**]**){ id7(id6); for(int id5=const2;**i**d5 **<** const1;id5**++**){ id4(id2[id3],id1); }}}; =>

=> class I{private: O public: I(O,**T** id2[const1]){ id7(id6); for(int id5=const2;**i**d5 **<** const1;id5**++**){ id4(id2[id3],id1); }}}; =>

=> class I{private: O public: I(**O**,int id2[const1]){ id7(id6); for(int id5=const2;**i**d5 **<** const1;id5**++**){ id4(id2[id3],id1); }}}; =>

=> class I{private: O public: I(T\* **I**,int id2[const1]){ id7(id6); for(int id5=const2;**i**d5 **<** const1;id5**++**){ id4(id2[id3],id1); }}}; =>

=> class I{private: O public: I(**T**\* id6,int id2[const1]){ id7(id6); for(int id5=const2;**i**d5 **<** const1;id5**++**){ id4(id2[id3],id1); }}}; =>

=> class I{private: O public: **I**(char\* id6,int id2[const1]){ id7(id6); for(int id5=const2;**i**d5 **<** const1;id5**++**){ id4(id2[id3],id1); }}}; =>

=> class I{private: **O** public:id7(char\* id6,int id2[const1]){ id7(id6); for(int id5=const2;**i**d5 **<** const1;id5**++**){ id4(id2[id3],id1); }}}; =>

=> class I{private: T\* I; **O** public:id7(char\* id6,int id2[const1]){ id7(id6); for(int id5=const2;**i**d5 **<** const1;id5**++**){ id4(id2[id3],id1); }}}; =>

=> class I{private: T\* I; T I [**C**]; public:id7(char\* id6,int id2[const1]){ id7(id6); for(int id5=const2;**i**d5 **<** const1;id5**++**){ id4(id2[id3],id1); }}}; =>

=> class I{private: T\* I; T **I** [id3]; public:id7(char\* id6,int id2[const1]){ id7(id6); for(int id5=const2;**i**d5 **<** const1;id5**++**){ id4(id2[id3],id1); }}}; =>

=> class I{private: T\* I; **T** id2 [id3]; public:id7(char\* id6,int id2[const1]){ id7(id6); for(int id5=const2;**i**d5 **<** const1;id5**++**){ id4(id2[id3],id1); }}}; =>

=> class I{private: T\* **I**; intid2 [id3]; public:id7(char\* id6,int id2[const1]){ id7(id6); for(int id5=const2;**i**d5 **<** const1;id5**++**){ id4(id2[id3],id1); }}}; =>

=> class I{private: **T**\* id6; intid2 [id3]; public:id7(char\* id6,int id2[const1]){ id7(id6); for(int id5=const2;**i**d5 **<** const1;id5**++**){ id4(id2[id3],id1); }}}; =>

=> class **I**{private: char\* id6; intid2 [id3]; public:id7(char\* id6,int id2[const1]){ id7(id6); for(int id5=const2;**i**d5 **<** const1;id5**++**){ id4(id2[id3],id1); }}}; =>

=> classid7{private: char\* id6; intid2 [id3]; public:id7(char\* id6,int id2[const1]){ id7(id6); for(int id5=const2;**i**d5 **<** const1;id5**++**){ id4(id2[id3],id1); }}}; =>

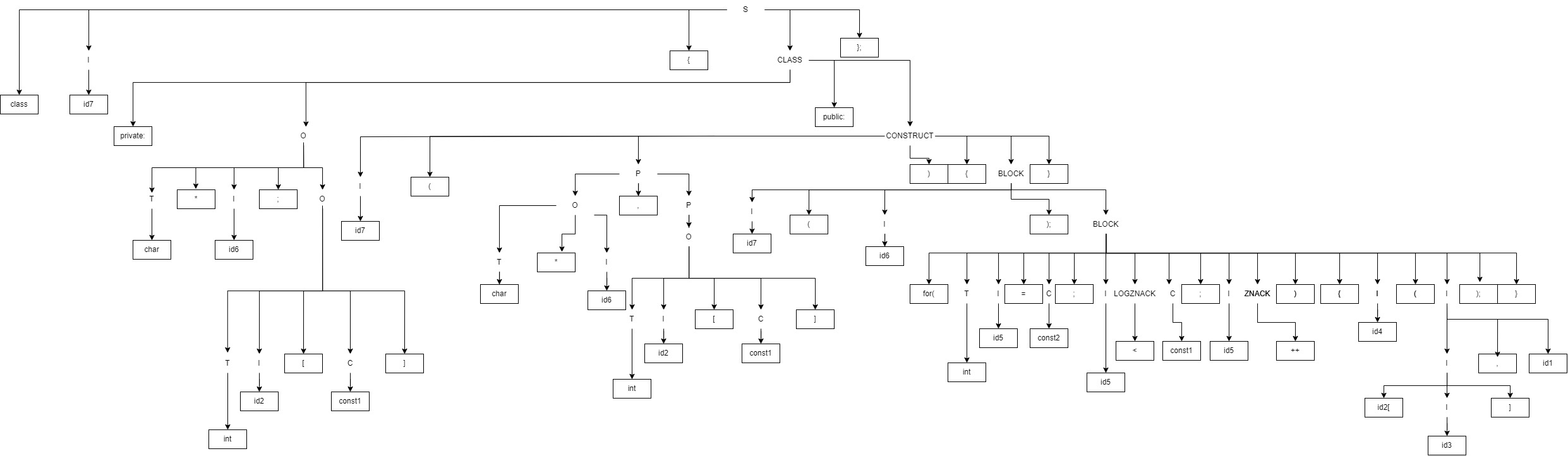


Рис.5. Дерево порождения

# 4. Логическое проектирование

1. Блок-схемы алгоритмов

С помощью блок-схем можно рассмотреть работу алгоритма программы в целом или в конкретные ее функции. Ниже представлена общая блок-схема алгоритма программы (рис. 6).

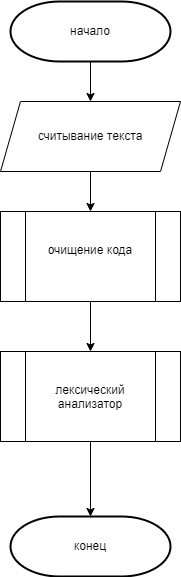


Рис.6. Общая блок-схема

Далее представлена блок-схема алгоритма очистки кода от комментариев и лишних пробелов (рис. 7-8).

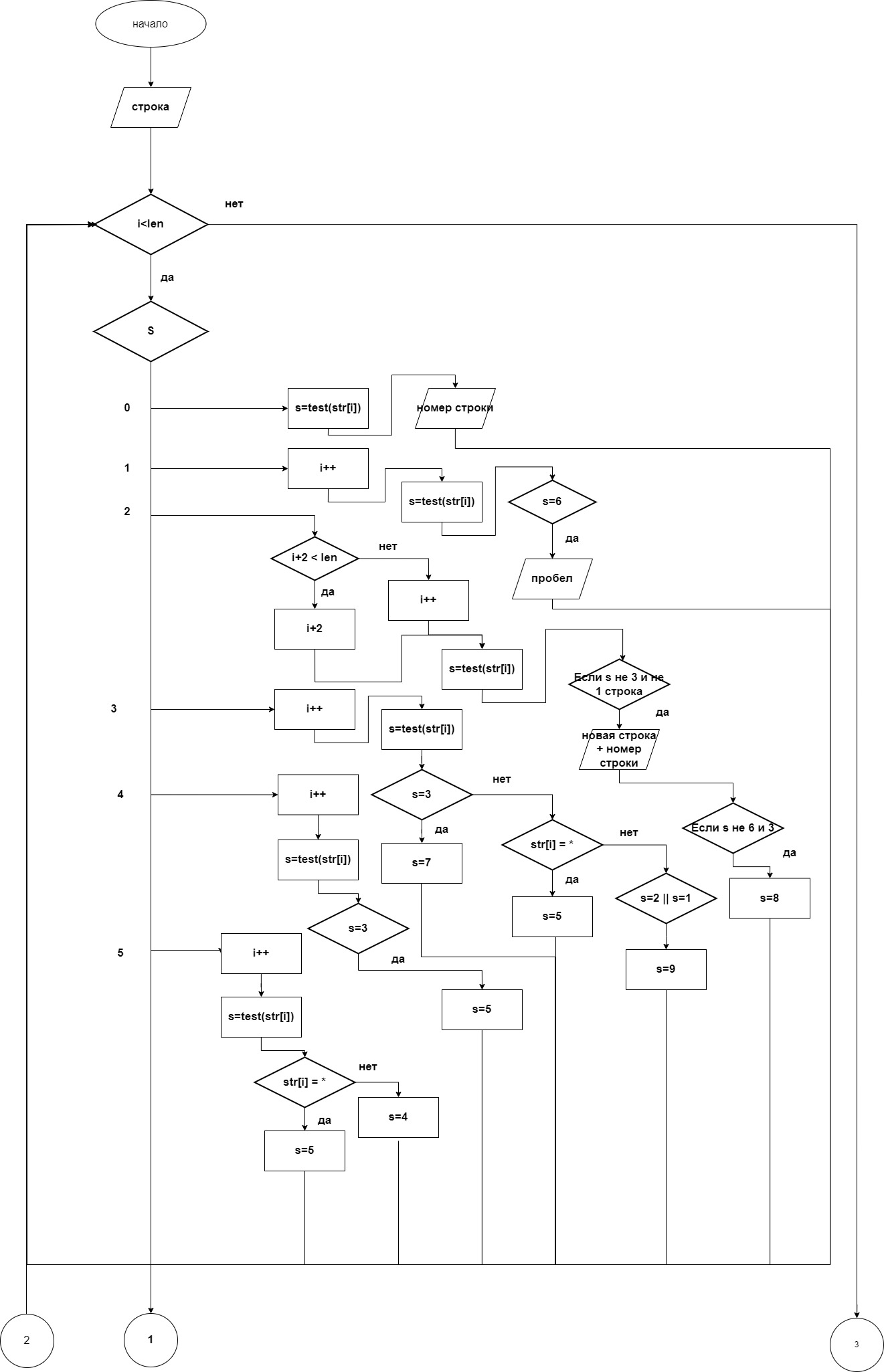


Рис.7. Блок-схема очищения кода. Часть 1

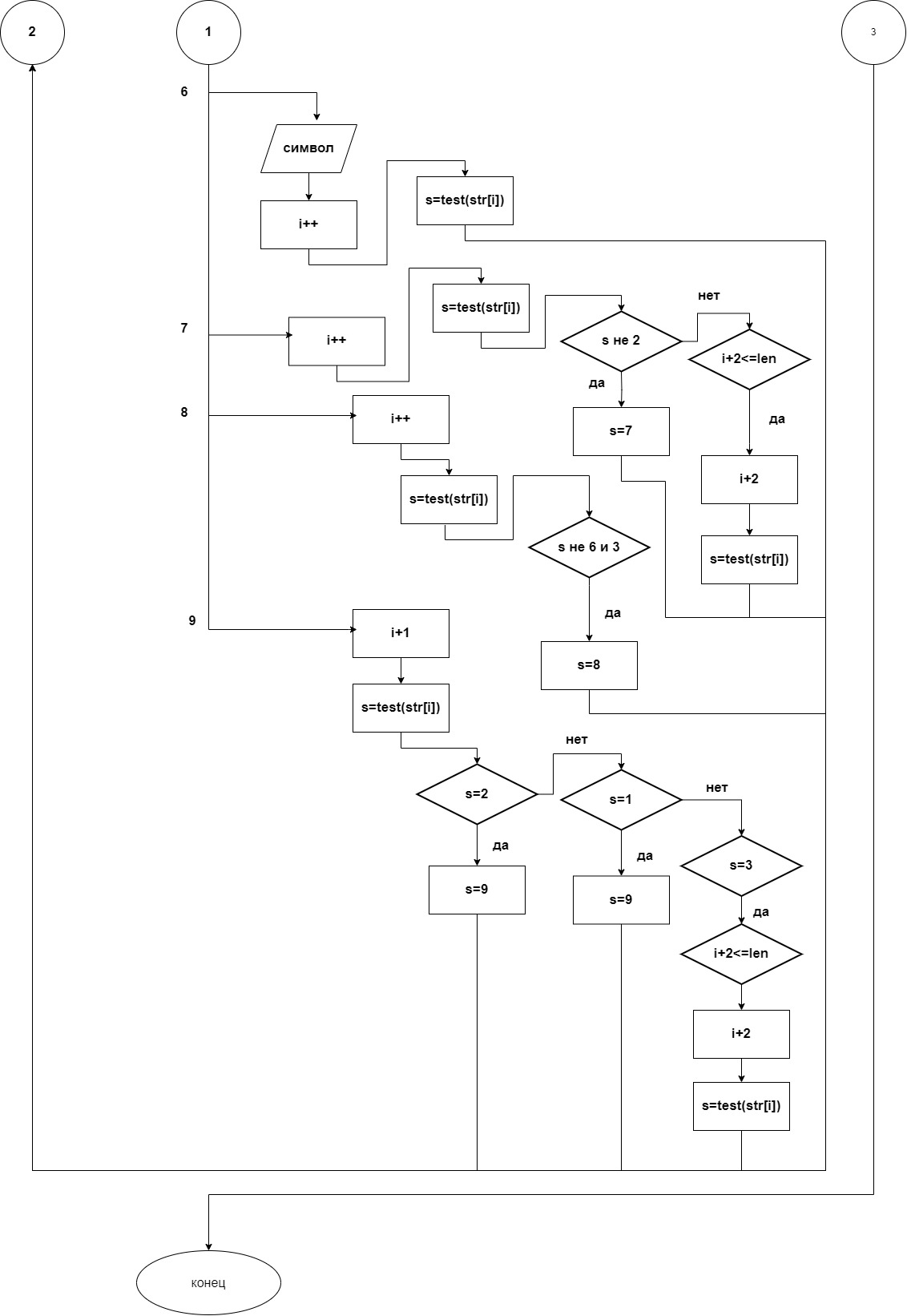


Рис.8. Блок-схема очищение кода. Часть 2

Далее представлена блок-схема функции test, которая используется для очистки входного кода от комментариев и лишних пробелов. Функция предназначена для определения дальнейшего состояния автомата в соответствии с входным символом (рис.9).

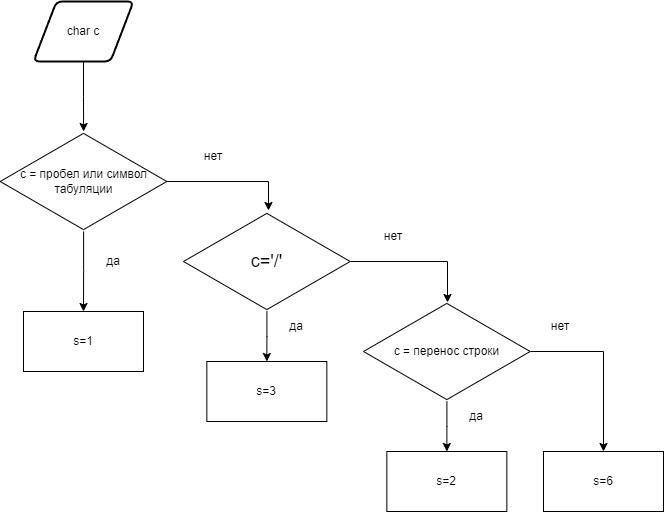


Рис.9. Блок-схема функции test

Далее представлена блок-схема функции, выполняющей роль лексического анализатора (рис.10) и блок-схема оператора switch из этой функции (рис. 11).

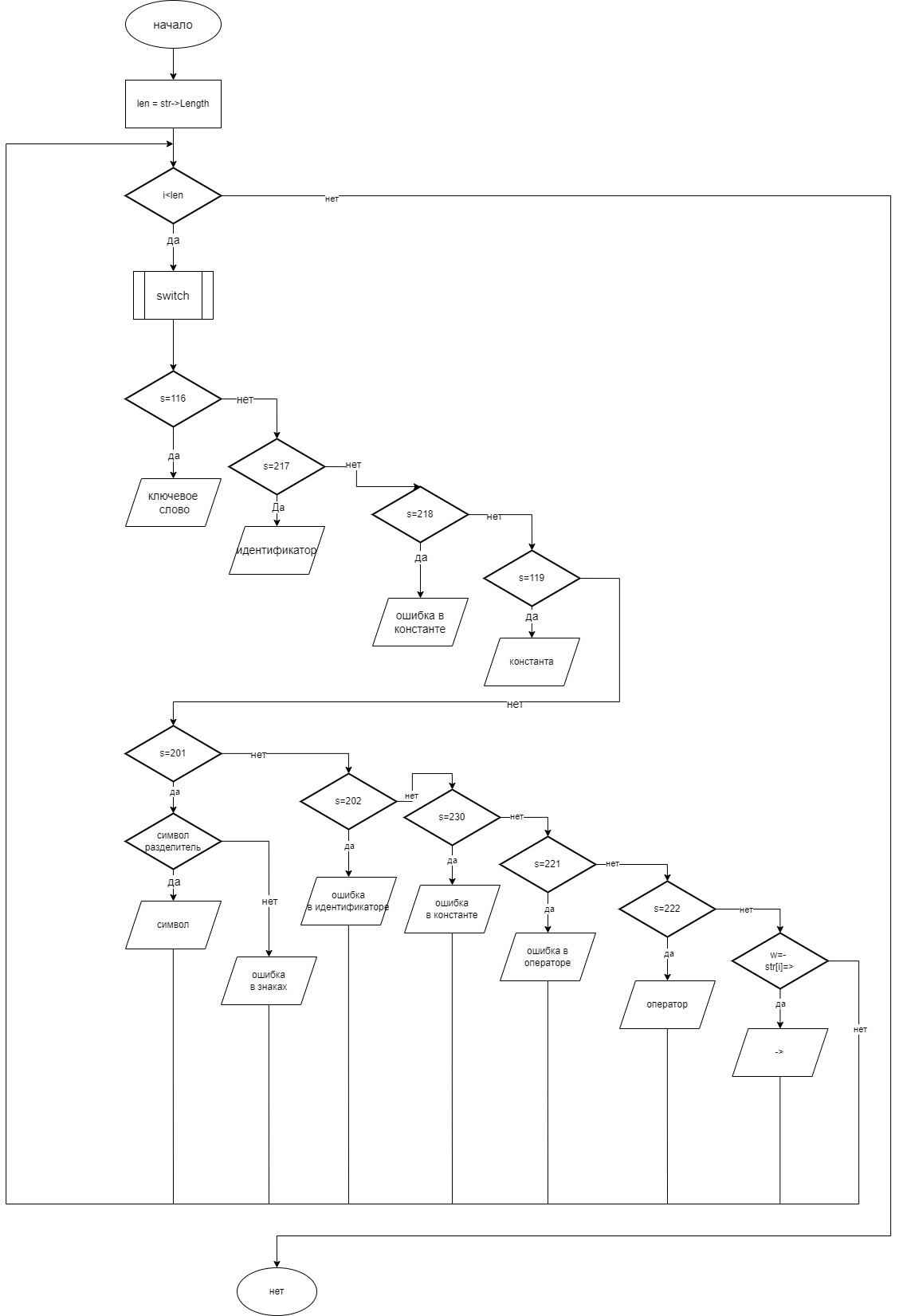


Рис.10. Блок-схема функции лексического оператора

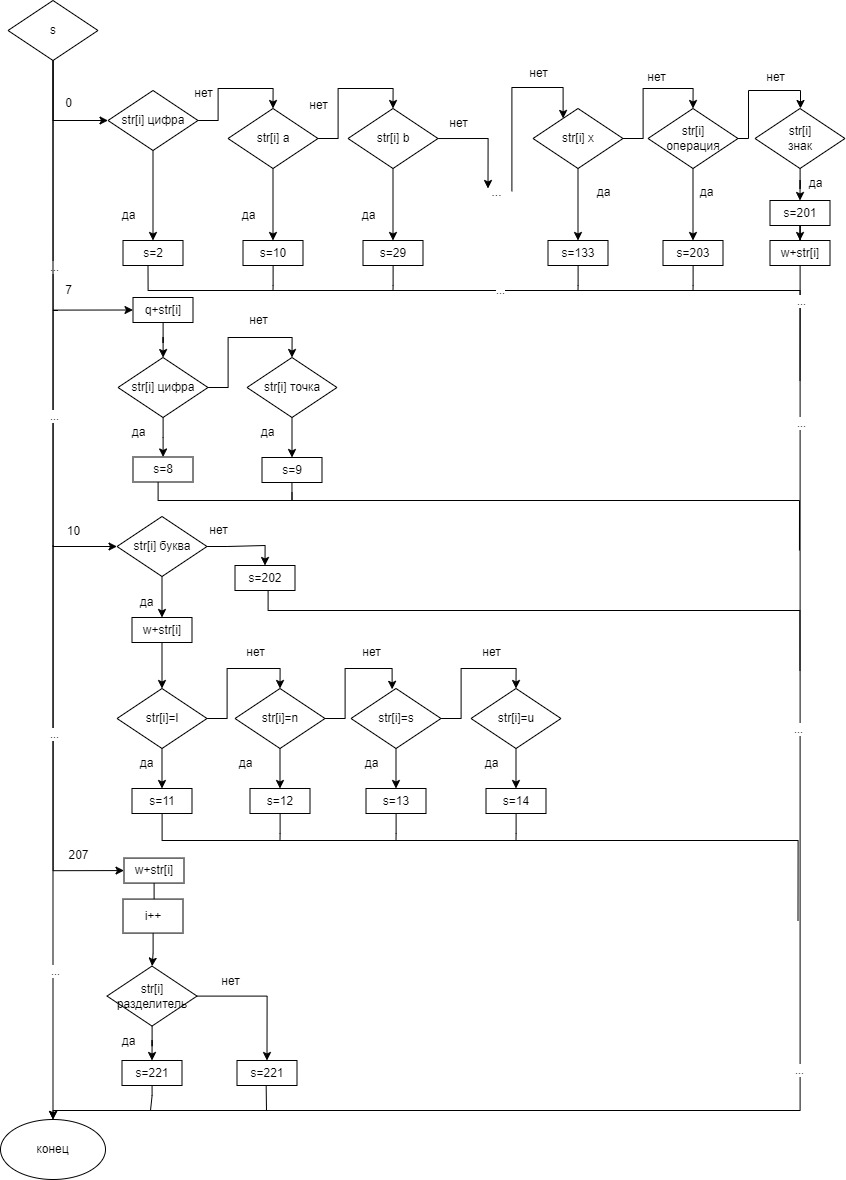


Рис. 11. Блок-схема блока switch

1. Словесное описание алгоритмов

На вход программа получает строку символов, каждый из которых анализируется отдельно с помощью цикла while, который продолжает работать до тех пор, пока не будет считан последний символ строки. От входного символа зависит дальнейшие действия программы, которые регулируются с помощью оператора switch. Программа реализует функцию удаления лишних пробелов, переносов строк и комментариев. Встретив символ, отвечающий за начало комментария, символы, следующие после не выводятся в поле «очищенный текст» до тех пор, пока не будет считан перенос строки, в случае однострочного комментария, или закрытие комментария в случае многострочного.

Для реализации функции лексического анализатора также используется оператор switch. Программа считывает символы до тех пор, пока не придет в конечное состояние, т.е. не определит входную лексему языка. Считанная лексема заносится в соответствующую таблицу лексем (которая определяется по состоянию автомата s) если ее прежде там не было, а также в соответствующие поля выводится дескрипторный и псевдокод данной лексемы. Так программа может определять следующие лексемы: ключевые слова, идентификаторы, числовые константы, знаки операции и другие символы. После ввода лексемы в соответствующие поля программа возвращается в состояние 0 и продолжает считывать символы для определения следующей лексемы.

Помимо определения лексем программа определяет ошибки во входном тексте, такие как ошибка в константе, в ключевом слове, в идентификаторе, в знаках операций и других символах. Ошибки выводятся в поле «Ошибки» и не заносятся ни в таблицы лексем, ни в дескрипторный и псевдокод.

1. Оценка сложности алгоритм

Для определения сложности алгоритма необходимо проверить на сложность каждый этап вычислений (табл.2).

Таблица 2

Оценка сложности этапов

|  |  |  |
| --- | --- | --- |
| Название этапа | Описание этапа | Оценка сложности |
| Очистка кода от лишних пробелов, комментариев и пустых строк | Программа проходит каждый символ исходного текста единожды проходя по состояниям автомата. Если символ удовлетворяет условиям автомата, то он добавляется конец строки, где находиться очищенный код | O(n), где n – длина исходного кода |
| Лексический анализ очищенного текста | Программа проходит каждый символ 1 раз проходя по состояниям автомата. Если нашлась лексема или ошибка, она записывается в соответствующую таблицу/поле  Количество символов = длине строки очищенного кода | Запись нового значения в список занимает O(1), значит сложность данного этапа O(c), где c – длина очищенного кода |

Общая оценка сложности всей программы равна самой сложной оценки всех этапов. Так как длина исходного кода всегда либо равна, либо больше длины очищенного кода, тогда общая оценка сложности программы равна O(n), где n – длина строки исходного кода.

# 5. Физическое проектирование

1. Выбор структур данных

Таблица 3

Структуры данных

|  |  |  |
| --- | --- | --- |
| Назначение | Обозначение | Тип данных |
| Входная строка | str | String^ |
| Номер строки | n | Int |
| Индекс по строке | i | Double |
| Длинна строки | len | Int |
| Состояние | s | Int |
| Строка числа | q | String^ |
| Строка слова | w | String^ |
| Счетчик ключевых слов | KW | Int |
| Счетчик констант | CT | Int |
| Счетчик идентификаторов | ID | Int |
| Счетчик знаков | ZO | Int |
| Счетчик символов | Cb | int |

1. Спецификация функция

Таблица 4

Спецификации функции

|  |  |  |  |
| --- | --- | --- | --- |
| Назначение | Обозначение | Тип | Параметры |
| Проверка символа для очищения кода | test | Int | char c |
| Очищение кода от пробелов и комментариев | clean | Void |  |
| Проверка символа для анализа кода | Test | Int | char c |
| Проверка символа на алфавите | cimbol | Int | char c |
| Проверка символа на знак | znack | Int | char c |
| Проверка символа на знак операции | operation | Int | char c |
| Анализ кода | Kw\_const | Void |  |
| Функция загрузки файла по нажатию на кнопку | upload\_file\_Click | Void | System::Object^ sender, System::EventArgs^ e |
| Функция запуска очищения кода и анализа по нажатию на кнопку | correct\_Click | Void | System::Object^ sender, System::EventArgs^ e |
| Функция очищения полей после нажатия на кнопку | button1\_Click | Void | System::Object^ sender, System::EventArgs^ e |

# 6. Проектирование интерфейса

Интерфейс — это набор инструментов, который позволяет пользователю взаимодействовать с программой. В более широком смысле термин обозначает любые инструменты для соприкосновения между разными системами и сущностями. Часто говорят о графическом интерфейсе — это внешний вид сайта, программы или приложения.

**Графический пользовательский интерфейс** (GUI) – разновидность пользовательского интерфейса, в котором элементы интерфейса (меню, кнопки, значки, списки и т. п.), представленные пользователю на дисплее. Именно этот тип интерфейса разрабатывался в ходе работы(рис. ).

Интерфейс приложения (рис. ) состоит из нескольких текстовых полей – «Исходный код», «Очищенный код», «Ошибки», «Псевдокод», «Дескрипторный код», также на экране расположено несколько кнопок для взаимодействия – «загрузить файл», «анализ», «сброс». Помимо этого, на экране располагаются таблицы, предназначенные для заполнения их лексемами из исходного кода – «ключевые слова», «константы», «знаки операций», «идентификаторы», «символы».

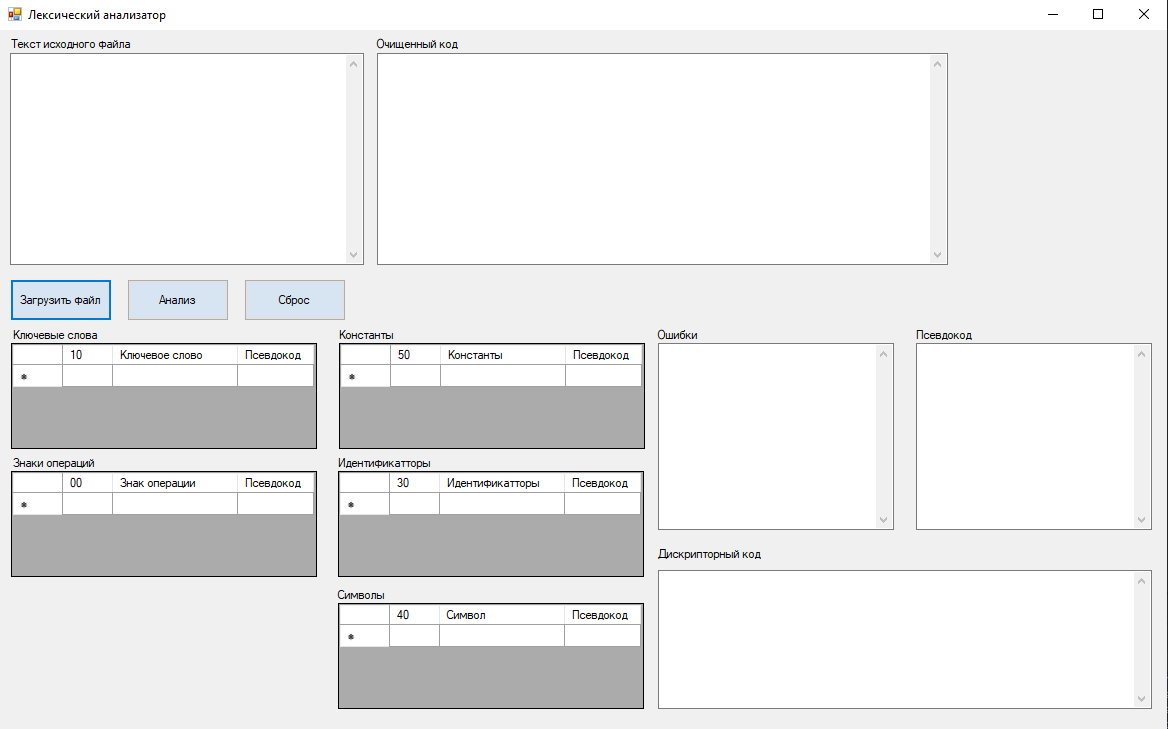


Рис.12. Интерфейс программы

# 7. Тестирование программного обеспечения

Во время тестирования была проанализирована работа программы при различных вариантах входных данных.

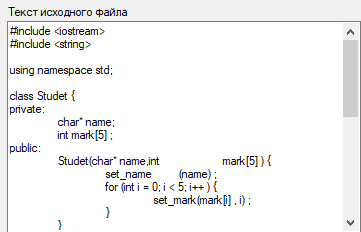


Рис.13. Считывание с файла

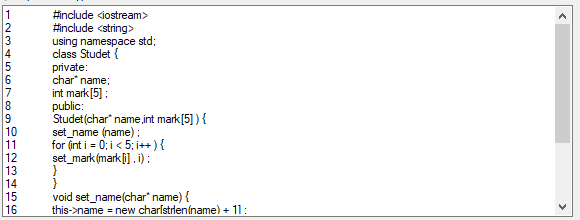


Рис.14. Удаление пробелов и комментариев

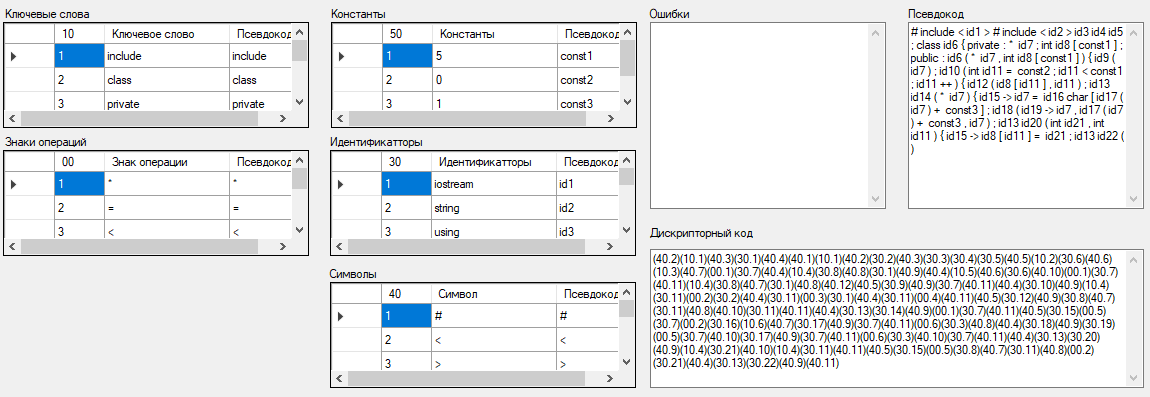


Рис.15. Лексический анализ

Таблица 6

Тестирование данных

|  |  |  |  |
| --- | --- | --- | --- |
| Дата тестирования | Тестовые данные | Описание теста | Результат |
| 13.04.2023 | Source.txt | Считывание текста с файла | Успех |
| 21.04.2023 | Source.txt | Очищение считанного кода от лишних пробелов и комментариев | Успех |
| 11.05.2023 | Source.txt | Определение ключевых слов | Успех |
| 13.05.2023 | Source.txt | Определение констант с плавающей точкой | Успех |
| 20.05.2023 | Source.txt | Определение идентификаторов | Успех |
| 25.05.2023 | Source.txt | Определение знаков операций из кода | Успех |
| 30.05.2023 | Source.txt | Определение прочих символов | Успех |
| 31.05.2023 | Source.txt | Определение ошибок | Успех |

Таблица 7

Тестирование программы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Дата тестирования | Модуль тестирования | Кто проводил тестирование | Описание теста | Результаты тестирования |
| 02.06.2023 | ТА1.exe | Еленников Г.М. | Проверка на правильность считывания информации с файла | Успех |
| 03.06.2023 | ТА1.exe | Иванова А.А. | Тест функции очищения кода | Успех |
| 03.06.2023 | ТА1.exe | Сарычева Е.В. | Тестирование ввода текста с клавиатуры | Успех |
| 04.06.2023 | ТА1.exe | Фомина К.С. | Тестирование функции лексического анализа | Ошибка при выявлении ключевых слов |
| 05.06.2023 | ТА1.exe | Иванова А.А. | Выявление ключевых слов | Ошибка в считывании по строкам |
| 07.06.2023 | ТА1.exe | Фомина К.С. | Лексический анализатор | Успех |

# 8. Заключение.

В ходе выполнения данной курсовой работы были освоены знания и навыки работы с автоматами и лексическимb анализаторомb. Было создано программное обеспечение выполняющая работу лексического анализатора на языке программирования C++ в соответствии с заданным вариантом(рис.2). Был смоделирован лексический анализатор компилятора на языке высокого уровня С++ с использованием Windows Forms. Данное приложение способно проводить обработку вводимого текста программы, удалять лишние пробелы, знаки табуляции, комментарии, излишние переходы на новую строку, выполнять подсчет количества строк в программе и выводить таблицу классов лексем, дескрипторный код и псевдокод.

Для программы созданы алгоритмы в виде блок-схем и словесной форме, подсчитана сложность алгоритма. Помимо этого, была разработана регулярная и КС-грамматики, построено дерево вывода и разобрано правостороннее порождение. Были построены таблицы переходов автомата и регулярная грамматика по нему.

Были получены знания в области построения автоматов, на основе которых строятся лексические анализаторы, компиляторы.

# 9. Список литературы.

1. Теория языков программирования и методы трансляции: учебное пособие / О. Г. Ганичева ; М-во образования и науки Российской Федерации, Федеральное агентство по образованию, ГОУ ВПО "Череповецкий гос. унт", Ин-т информ. технологий. - Череповец : ГОУ ВПО "Череповецкий гос. унт", 2011. – 185 с.
2. Е.В. Ершов, д-р техн. наук, проф.; Л.Н. Виноградова и др. «Методика и организация самостоятельной работы студентов» Коллектив авторов, 2012 ФГБОУ ВПО «Череповецкий государственный университет», 2012. – 208 с.
3. Теория автоматов. Ю.Г. Карпов – СПб.: Питер, 2003. – 208 с.
4. Проектирование лексического анализатора для интерактивного конструктора задачи многокритериальной оптимизации Охотниченко А.В., Сибилева Н.С. Математическое и программное обеспечение систем в промышленной и социальной сферах. 2018. Т. 6. № 1. С. 40.
5. Сайт Microsoft с документация по языку программирования C# [эл. ресурс] https://learn.microsoft.com/ru-ru/cpp/cpp/cpp-language-reference?view=msvc-170

Приложение 1. Техническое задание

МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное

образовательное учреждение высшего образования

ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Институт информационных технологий

Математического и программного обеспечения ЭВМ

Модуль «Информатика»

УТВЕРЖДАЮ

Зав. кафедрой МПО ЭВМ

д. т.н. \_\_\_\_\_\_\_\_\_\_ Ершов Е.В.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

Разработка алгоритмического обеспечения и построение лексического и синтаксического анализаторов

Техническое задание на курсовую работу

Листов 7

Руководители: Ганичева О. Г. и Пышницкий К.М

Исполнитель: студент гр. 1ПИб-01-1оп-21

Фомина К.С.

2023 год

Введение

Данная курсовая работа посвящена разработке алгоритмического обеспечения и построение лексического и синтаксического анализаторов, с помощью которых пользователь сможет удалять из поступающего на вход кода лишние пробелы, строки и комментарии, выделять лексемы и ошибки, подсчитывать количество строк программы.

1. Основания для разработки

Основанием для разработки является задание на курсовую работу по дисциплине Модуль «Информатика», выданное на кафедре МПО ЭВМ ИИТ ЧГУ.

Дата утверждения: март 2023 года

Наименование темы разработки: «Разработка алгоритмического обеспечения и построение лексического и синтаксического анализаторов».

2. Назначение разработки

Разработать программу, моделирующую работу лексического анализатора, которая решает следующие задачи: выделяет из текста входной программы все лексемы языка С++, удаляет лишние пробелы, строки и комментарии из входного кода, диагностирует и локализирует лексические ошибки, подсчитывает количество строк в программе.

3. Требования к программе

3.1. Требования к функциональным характеристикам

Модель лексического анализатора должна проводить обработку вводимого текста программы и удалять лишние пробелы, знаки табуляции, комментарии, лишние переходы на новую строку, выполнять подсчет количества строк в программе, диагностировать и локализировать лексические ошибки, выводить таблицу классов лексем, дескрипторный код и псевдокод.

Все конструкции должны предусматривать наличие следующих операторов: составного, присваивания, сравнения, ввода/вывода, разделителей операторов. А также арифметические и (или) логические операции.

Работа лексического анализатора должна моделироваться конечным автоматом, граф переходов которого должен быть представлен в расчетно-пояснительной записке.

Программа должна выводить ошибки: неопознанный символ, неправильный оператор, ошибка в числе, ошибка в ключевом слове.

3.2. Требования к надёжности

Для правильной работы программы необходимо:

1) количество вводимых пользователем символов кода в текстовом поле программы не должно превышать 65528;

2) вводимые лексемы должны удовлетворять возможным лексемам, которые определены с помощью состояний конечного автомата.

3.3. Условия эксплуатации

Условия эксплуатации ограниченны нормами работы в офисном помещении:

* относительная влажность воздуха не более 80%;
* запыленность воздуха не более 0,75 мг/м³;
* знание основ работы в операционной системе Windows и браузере.

3.4. Требования к составу и параметрам технических средств

* процессор с тактовой частотой 1000 MHz или более мощный;
* оперативная память 512 Мб или больше;
* свободное место на жёстком диске от 120 Мб;
* архитектура с разрядностью 32 бит или 64 бит;
* операционная система Windows, Android, Linux, iOS;
* наличие компьютерной мыши, клавиатуры и монитора (для ПК).

3.5. Требования к информационной и программной совместимости

Текст программы реализуется на языке высоко уровня – C++ в программе Visual Studio 2022. Для выполнения действий, связанных с редактированием или дописыванием кода, в том случае, если на персональном электронном устройстве установлена более ранняя версия Visual Studio, необходимо установить дополнительно набор инструментов платформы v142 и пакет SDK для Windows версией 10.0. Для выполнения всех действий, связанных с работой программы, подходят такие операционные системы как Windows 10.

3.6. Требования к маркировке и упаковке

Распространение через средства коммуникации (для некоммерческого использования).

3.7. Требования к транспортированию и хранению

Транспортировку и хранение рекомендуется выполнять на оптическом носителе типа USB Flash, в условиях подходящих для выбранного носителя данных, ограничение на срок хранения отсутствует.

3.8. Специальные требования

Минимальное умение пользования компьютером.

* 1. Требование к программной документации

Программная документация должна содержать:

4.1. Программная документация должна содержать расчётно-пояснительную записку с содержанием:

* Титульный лист
* Оглавление
* Введение
  + - * + Теоретическая часть
        + Практическая часть
* Заключение
* Список литературы
* Приложения

4.2. Требования к оформлению:

* Документ
* Печать на отдельных листах формата А4 (210х297 мм); оборотная сторона не заполняется; листы нумеруются. Печать возможна ч/б.
* Файлы предъявляются на компакт-диске: РПЗ с ТЗ; программный код.
* Листы и диск в конверте вложены в пластиковую папку скоросшивателя.
* Страницы
* Ориентация – книжная; отдельные страницы, при необходимости, альбомная.
* Поля: верхнее, нижнее – по 2 см, левое – 3 см , правое – 1 см.
* Абзацы
* Межстрочный интервал – 1,5, перед и после абзаца – 0.
* Шрифты
* Кегль – 14. В таблицах шрифт 12. Шрифт листинга – 10.
* Рисунки
* Подписывается под ним по центру: Рис.Х. Название

В приложениях: Рис.П1.3. Название

* Таблицы
* Подписывается: над таблицей, выравнивание по правому: «Таблица Х».
* В следующей строке по центру Название
* Надписи в шапке (имена столбцов, полей) – по центру.
* В теле таблицы (записи) текстовые значения – выровнены по левому, числа, даты – по правому.

5. Стадии и этапы разработки

В данном пункте представлены этапы разработки, которые будут выполняться при разработке программы, в соответствии с табл. П1.1.

Таблица П1.1

Этапы разработки

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование этапа разработки | Сроки разработки | Результат выполнения | Отметка о выполнении |
| Сбор необходимой информации для выполнения данной курсовой работы | Март- апрель 2023г. | Овладение необходимыми знаниями для написания кода программы | Выполнено |
| Составление алгоритма программы | Март 2023г. | Выявление и составление четкого алгоритма программы | Выполнено |
| Реализация алгоритма | Апрель-май 2023г. | Получение технически правильного кода программы | Выполнено |
| Тестирование | Май 2023г. | Исправление ошибок и сравнение результатов работы программы и самостоятельных подсчетов | Выполнено |

6. Порядок контроля и приемки

Контроль выполнения работы осуществляется преподавателем. В случае обнаружения недостатков, исполнитель работы обязан исправить их в срок, согласованный с преподавателем. Контроль разработки продукта осуществляется поэтапно, в соответствии с графиком, представленным в табл. П1.2.

Таблица П1.2

Порядок контроля и приемки

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование контрольного этапа выполнения курсовой работы | Сроки контроля | Результат выполнения | Отметка о приемке результата контрольного этапа |
| Сдача технического задания | 28 апреля 2023г. | Документ “Техническое задание” проверен |  |
| Демонстрация работы программы | Июнь 2023г. | Работа программы проверена |  |
| Защита курсовой работы | Июнь 2023г. | Оценка |  |

Приложение 2.

Текст программы

Файл MyForm.cpp

#include "MyForm.h"

using namespace System;

using namespace System::Windows::Forms;

using namespace System::Windows::Forms;

[STAThreadAttribute]

int main(array<String^>^ args)

{

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

TA1::MyForm form;

Application::Run(% form);

}

Файл MyForm.h

#include <iostream>

#include <String>

#pragma once

namespace TA1 {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

/// <summary>

/// Сводка для MyForm

/// </summary>

public ref class MyForm : public System::Windows::Forms::Form

{

public:

MyForm(void)

{

InitializeComponent();

//

//TODO: добавьте код конструктора

//

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~MyForm()

{

if (components)

{

delete components;

}

}

private: System::Windows::Forms::Button^ upload\_file;

private: System::Windows::Forms::TextBox^ source;

private: System::Windows::Forms::Label^ label1;

private: System::Windows::Forms::TextBox^ corrected\_code;

private: System::Windows::Forms::Label^ label2;

private: System::Windows::Forms::OpenFileDialog^ openFileDialog1;

private: System::Windows::Forms::Button^ correct;

private: System::Windows::Forms::Button^ button1;

private: System::Windows::Forms::Label^ label3;

private: System::Windows::Forms::Label^ label4;

private: System::Windows::Forms::TextBox^ Error;

private: System::Windows::Forms::Label^ label5;

private: System::Windows::Forms::DataGridView^ kwordGrid;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn4;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn5;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn6;

private: System::Windows::Forms::DataGridView^ ConstGrid;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn1;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn2;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn3;

private: System::Windows::Forms::DataGridView^ IdentGrid;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn7;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn8;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn9;

private: System::Windows::Forms::Label^ label6;

private: System::Windows::Forms::DataGridView^ SimbolGrid;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn13;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn14;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn15;

private: System::Windows::Forms::DataGridView^ OperGrid;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn16;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn17;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn18;

private: System::Windows::Forms::Label^ label8;

private: System::Windows::Forms::Label^ label9;

private: System::Windows::Forms::Label^ label10;

private: System::Windows::Forms::TextBox^ psevdo;

private: System::Windows::Forms::Label^ label11;

private: System::Windows::Forms::TextBox^ discript;

protected:

private:

/// <summary>

/// Обязательная переменная конструктора.

/// </summary>

System::ComponentModel::Container^ components;

#pragma region Windows Form Designer generated code

/// <summary>

/// Требуемый метод для поддержки конструктора — не изменяйте

/// содержимое этого метода с помощью редактора кода.

/// </summary>

void InitializeComponent(void)

{

this->upload\_file = (gcnew System::Windows::Forms::Button());

this->source = (gcnew System::Windows::Forms::TextBox());

this->label1 = (gcnew System::Windows::Forms::Label());

this->corrected\_code = (gcnew System::Windows::Forms::TextBox());

this->label2 = (gcnew System::Windows::Forms::Label());

this->openFileDialog1 = (gcnew System::Windows::Forms::OpenFileDialog());

this->correct = (gcnew System::Windows::Forms::Button());

this->button1 = (gcnew System::Windows::Forms::Button());

this->label3 = (gcnew System::Windows::Forms::Label());

this->label4 = (gcnew System::Windows::Forms::Label());

this->Error = (gcnew System::Windows::Forms::TextBox());

this->label5 = (gcnew System::Windows::Forms::Label());

this->kwordGrid = (gcnew System::Windows::Forms::DataGridView());

this->dataGridViewTextBoxColumn4 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->dataGridViewTextBoxColumn5 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->dataGridViewTextBoxColumn6 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->ConstGrid = (gcnew System::Windows::Forms::DataGridView());

this->dataGridViewTextBoxColumn1 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->dataGridViewTextBoxColumn2 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->dataGridViewTextBoxColumn3 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->IdentGrid = (gcnew System::Windows::Forms::DataGridView());

this->dataGridViewTextBoxColumn7 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->dataGridViewTextBoxColumn8 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->dataGridViewTextBoxColumn9 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->label6 = (gcnew System::Windows::Forms::Label());

this->SimbolGrid = (gcnew System::Windows::Forms::DataGridView());

this->dataGridViewTextBoxColumn13 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->dataGridViewTextBoxColumn14 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->dataGridViewTextBoxColumn15 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->OperGrid = (gcnew System::Windows::Forms::DataGridView());

this->dataGridViewTextBoxColumn16 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->dataGridViewTextBoxColumn17 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->dataGridViewTextBoxColumn18 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->label8 = (gcnew System::Windows::Forms::Label());

this->label9 = (gcnew System::Windows::Forms::Label());

this->label10 = (gcnew System::Windows::Forms::Label());

this->psevdo = (gcnew System::Windows::Forms::TextBox());

this->label11 = (gcnew System::Windows::Forms::Label());

this->discript = (gcnew System::Windows::Forms::TextBox());

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->kwordGrid))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->ConstGrid))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->IdentGrid))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->SimbolGrid))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->OperGrid))->BeginInit();

this->SuspendLayout();

//

// upload\_file

//

this->upload\_file->BackColor = System::Drawing::SystemColors::GradientInactiveCaption;

this->upload\_file->Location = System::Drawing::Point(11, 249);

this->upload\_file->Margin = System::Windows::Forms::Padding(2);

this->upload\_file->Name = L"upload\_file";

this->upload\_file->Size = System::Drawing::Size(102, 42);

this->upload\_file->TabIndex = 0;

this->upload\_file->Text = L"Загрузить файл";

this->upload\_file->UseVisualStyleBackColor = false;

this->upload\_file->Click += gcnew System::EventHandler(this, &MyForm::upload\_file\_Click);

//

// source

//

this->source->Location = System::Drawing::Point(11, 23);

this->source->Margin = System::Windows::Forms::Padding(2);

this->source->Multiline = true;

this->source->Name = L"source";

this->source->ScrollBars = System::Windows::Forms::ScrollBars::Vertical;

this->source->Size = System::Drawing::Size(354, 212);

this->source->TabIndex = 1;

//

// label1

//

this->label1->AutoSize = true;

this->label1->Location = System::Drawing::Point(9, 7);

this->label1->Margin = System::Windows::Forms::Padding(2, 0, 2, 0);

this->label1->Name = L"label1";

this->label1->Size = System::Drawing::Size(127, 13);

this->label1->TabIndex = 2;

this->label1->Text = L"Текст исходного файла";

//

// corrected\_code

//

this->corrected\_code->Location = System::Drawing::Point(378, 23);

this->corrected\_code->Margin = System::Windows::Forms::Padding(2);

this->corrected\_code->Multiline = true;

this->corrected\_code->Name = L"corrected\_code";

this->corrected\_code->ScrollBars = System::Windows::Forms::ScrollBars::Vertical;

this->corrected\_code->Size = System::Drawing::Size(571, 212);

this->corrected\_code->TabIndex = 3;

//

// label2

//

this->label2->AutoSize = true;

this->label2->Location = System::Drawing::Point(375, 7);

this->label2->Margin = System::Windows::Forms::Padding(2, 0, 2, 0);

this->label2->Name = L"label2";

this->label2->Size = System::Drawing::Size(88, 13);

this->label2->TabIndex = 4;

this->label2->Text = L"Очищенный код";

this->label2->Click += gcnew System::EventHandler(this, &MyForm::label2\_Click);

//

// correct

//

this->correct->BackColor = System::Drawing::SystemColors::GradientInactiveCaption;

this->correct->Location = System::Drawing::Point(128, 249);

this->correct->Margin = System::Windows::Forms::Padding(2);

this->correct->Name = L"correct";

this->correct->Size = System::Drawing::Size(102, 42);

this->correct->TabIndex = 5;

this->correct->Text = L"Очистить код";

this->correct->UseVisualStyleBackColor = false;

this->correct->Click += gcnew System::EventHandler(this, &MyForm::correct\_Click);

//

// button1

//

this->button1->BackColor = System::Drawing::SystemColors::GradientInactiveCaption;

this->button1->Location = System::Drawing::Point(245, 249);

this->button1->Margin = System::Windows::Forms::Padding(2);

this->button1->Name = L"button1";

this->button1->Size = System::Drawing::Size(102, 42);

this->button1->TabIndex = 6;

this->button1->Text = L"Сброс";

this->button1->UseVisualStyleBackColor = false;

this->button1->Click += gcnew System::EventHandler(this, &MyForm::button1\_Click);

//

// label3

//

this->label3->AutoSize = true;

this->label3->Location = System::Drawing::Point(11, 298);

this->label3->Margin = System::Windows::Forms::Padding(2, 0, 2, 0);

this->label3->Name = L"label3";

this->label3->Size = System::Drawing::Size(92, 13);

this->label3->TabIndex = 9;

this->label3->Text = L"Ключевые слова";

//

// label4

//

this->label4->AutoSize = true;

this->label4->Location = System::Drawing::Point(337, 298);

this->label4->Margin = System::Windows::Forms::Padding(2, 0, 2, 0);

this->label4->Name = L"label4";

this->label4->Size = System::Drawing::Size(62, 13);

this->label4->TabIndex = 10;

this->label4->Text = L"Константы";

//

// Error

//

this->Error->Location = System::Drawing::Point(659, 313);

this->Error->Margin = System::Windows::Forms::Padding(2);

this->Error->Multiline = true;

this->Error->Name = L"Error";

this->Error->ScrollBars = System::Windows::Forms::ScrollBars::Vertical;

this->Error->Size = System::Drawing::Size(236, 187);

this->Error->TabIndex = 11;

//

// label5

//

this->label5->AutoSize = true;

this->label5->Location = System::Drawing::Point(656, 298);

this->label5->Margin = System::Windows::Forms::Padding(2, 0, 2, 0);

this->label5->Name = L"label5";

this->label5->Size = System::Drawing::Size(47, 13);

this->label5->TabIndex = 12;

this->label5->Text = L"Ошибки";

//

// kwordGrid

//

this->kwordGrid->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->kwordGrid->Columns->AddRange(gcnew cli::array< System::Windows::Forms::DataGridViewColumn^ >(3) {

this->dataGridViewTextBoxColumn4,

this->dataGridViewTextBoxColumn5, this->dataGridViewTextBoxColumn6

});

this->kwordGrid->Location = System::Drawing::Point(12, 313);

this->kwordGrid->Margin = System::Windows::Forms::Padding(2);

this->kwordGrid->Name = L"kwordGrid";

this->kwordGrid->RowHeadersWidth = 51;

this->kwordGrid->RowTemplate->Height = 24;

this->kwordGrid->Size = System::Drawing::Size(306, 106);

this->kwordGrid->TabIndex = 13;

//

// dataGridViewTextBoxColumn4

//

this->dataGridViewTextBoxColumn4->HeaderText = L"10";

this->dataGridViewTextBoxColumn4->MinimumWidth = 6;

this->dataGridViewTextBoxColumn4->Name = L"dataGridViewTextBoxColumn4";

this->dataGridViewTextBoxColumn4->Width = 50;

//

// dataGridViewTextBoxColumn5

//

this->dataGridViewTextBoxColumn5->HeaderText = L"Ключевое слово";

this->dataGridViewTextBoxColumn5->MinimumWidth = 6;

this->dataGridViewTextBoxColumn5->Name = L"dataGridViewTextBoxColumn5";

this->dataGridViewTextBoxColumn5->Width = 125;

//

// dataGridViewTextBoxColumn6

//

this->dataGridViewTextBoxColumn6->HeaderText = L"Псевдокод";

this->dataGridViewTextBoxColumn6->MinimumWidth = 6;

this->dataGridViewTextBoxColumn6->Name = L"dataGridViewTextBoxColumn6";

this->dataGridViewTextBoxColumn6->Width = 76;

//

// ConstGrid

//

this->ConstGrid->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->ConstGrid->Columns->AddRange(gcnew cli::array< System::Windows::Forms::DataGridViewColumn^ >(3) {

this->dataGridViewTextBoxColumn1,

this->dataGridViewTextBoxColumn2, this->dataGridViewTextBoxColumn3

});

this->ConstGrid->Location = System::Drawing::Point(340, 313);

this->ConstGrid->Margin = System::Windows::Forms::Padding(2);

this->ConstGrid->Name = L"ConstGrid";

this->ConstGrid->RowHeadersWidth = 51;

this->ConstGrid->RowTemplate->Height = 24;

this->ConstGrid->Size = System::Drawing::Size(306, 106);

this->ConstGrid->TabIndex = 14;

//

// dataGridViewTextBoxColumn1

//

this->dataGridViewTextBoxColumn1->HeaderText = L"50";

this->dataGridViewTextBoxColumn1->MinimumWidth = 6;

this->dataGridViewTextBoxColumn1->Name = L"dataGridViewTextBoxColumn1";

this->dataGridViewTextBoxColumn1->Width = 50;

//

// dataGridViewTextBoxColumn2

//

this->dataGridViewTextBoxColumn2->HeaderText = L"Константы";

this->dataGridViewTextBoxColumn2->MinimumWidth = 6;

this->dataGridViewTextBoxColumn2->Name = L"dataGridViewTextBoxColumn2";

this->dataGridViewTextBoxColumn2->Width = 125;

//

// dataGridViewTextBoxColumn3

//

this->dataGridViewTextBoxColumn3->HeaderText = L"Псевдокод";

this->dataGridViewTextBoxColumn3->MinimumWidth = 6;

this->dataGridViewTextBoxColumn3->Name = L"dataGridViewTextBoxColumn3";

this->dataGridViewTextBoxColumn3->Width = 76;

//

// IdentGrid

//

this->IdentGrid->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->IdentGrid->Columns->AddRange(gcnew cli::array< System::Windows::Forms::DataGridViewColumn^ >(3) {

this->dataGridViewTextBoxColumn7,

this->dataGridViewTextBoxColumn8, this->dataGridViewTextBoxColumn9

});

this->IdentGrid->Location = System::Drawing::Point(339, 441);

this->IdentGrid->Margin = System::Windows::Forms::Padding(2);

this->IdentGrid->Name = L"IdentGrid";

this->IdentGrid->RowHeadersWidth = 51;

this->IdentGrid->RowTemplate->Height = 24;

this->IdentGrid->Size = System::Drawing::Size(306, 106);

this->IdentGrid->TabIndex = 18;

//

// dataGridViewTextBoxColumn7

//

this->dataGridViewTextBoxColumn7->HeaderText = L"30";

this->dataGridViewTextBoxColumn7->MinimumWidth = 6;

this->dataGridViewTextBoxColumn7->Name = L"dataGridViewTextBoxColumn7";

this->dataGridViewTextBoxColumn7->Width = 50;

//

// dataGridViewTextBoxColumn8

//

this->dataGridViewTextBoxColumn8->HeaderText = L"Идентификатторы";

this->dataGridViewTextBoxColumn8->MinimumWidth = 6;

this->dataGridViewTextBoxColumn8->Name = L"dataGridViewTextBoxColumn8";

this->dataGridViewTextBoxColumn8->Width = 125;

//

// dataGridViewTextBoxColumn9

//

this->dataGridViewTextBoxColumn9->HeaderText = L"Псевдокод";

this->dataGridViewTextBoxColumn9->MinimumWidth = 6;

this->dataGridViewTextBoxColumn9->Name = L"dataGridViewTextBoxColumn9";

this->dataGridViewTextBoxColumn9->Width = 76;

//

// label6

//

this->label6->AutoSize = true;

this->label6->Location = System::Drawing::Point(336, 426);

this->label6->Margin = System::Windows::Forms::Padding(2, 0, 2, 0);

this->label6->Name = L"label6";

this->label6->Size = System::Drawing::Size(100, 13);

this->label6->TabIndex = 16;

this->label6->Text = L"Идентификатторы";

//

// SimbolGrid

//

this->SimbolGrid->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->SimbolGrid->Columns->AddRange(gcnew cli::array< System::Windows::Forms::DataGridViewColumn^ >(3) {

this->dataGridViewTextBoxColumn13,

this->dataGridViewTextBoxColumn14, this->dataGridViewTextBoxColumn15

});

this->SimbolGrid->Location = System::Drawing::Point(339, 573);

this->SimbolGrid->Margin = System::Windows::Forms::Padding(2);

this->SimbolGrid->Name = L"SimbolGrid";

this->SimbolGrid->RowHeadersWidth = 51;

this->SimbolGrid->RowTemplate->Height = 24;

this->SimbolGrid->Size = System::Drawing::Size(306, 106);

this->SimbolGrid->TabIndex = 22;

//

// dataGridViewTextBoxColumn13

//

this->dataGridViewTextBoxColumn13->HeaderText = L"40";

this->dataGridViewTextBoxColumn13->MinimumWidth = 6;

this->dataGridViewTextBoxColumn13->Name = L"dataGridViewTextBoxColumn13";

this->dataGridViewTextBoxColumn13->Width = 50;

//

// dataGridViewTextBoxColumn14

//

this->dataGridViewTextBoxColumn14->HeaderText = L"Символ";

this->dataGridViewTextBoxColumn14->MinimumWidth = 6;

this->dataGridViewTextBoxColumn14->Name = L"dataGridViewTextBoxColumn14";

this->dataGridViewTextBoxColumn14->Width = 125;

//

// dataGridViewTextBoxColumn15

//

this->dataGridViewTextBoxColumn15->HeaderText = L"Псевдокод";

this->dataGridViewTextBoxColumn15->MinimumWidth = 6;

this->dataGridViewTextBoxColumn15->Name = L"dataGridViewTextBoxColumn15";

this->dataGridViewTextBoxColumn15->Width = 76;

//

// OperGrid

//

this->OperGrid->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->OperGrid->Columns->AddRange(gcnew cli::array< System::Windows::Forms::DataGridViewColumn^ >(3) {

this->dataGridViewTextBoxColumn16,

this->dataGridViewTextBoxColumn17, this->dataGridViewTextBoxColumn18

});

this->OperGrid->Location = System::Drawing::Point(12, 441);

this->OperGrid->Margin = System::Windows::Forms::Padding(2);

this->OperGrid->Name = L"OperGrid";

this->OperGrid->RowHeadersWidth = 51;

this->OperGrid->RowTemplate->Height = 24;

this->OperGrid->Size = System::Drawing::Size(306, 106);

this->OperGrid->TabIndex = 21;

//

// dataGridViewTextBoxColumn16

//

this->dataGridViewTextBoxColumn16->HeaderText = L"00";

this->dataGridViewTextBoxColumn16->MinimumWidth = 6;

this->dataGridViewTextBoxColumn16->Name = L"dataGridViewTextBoxColumn16";

this->dataGridViewTextBoxColumn16->Width = 50;

//

// dataGridViewTextBoxColumn17

//

this->dataGridViewTextBoxColumn17->HeaderText = L"Знак операции";

this->dataGridViewTextBoxColumn17->MinimumWidth = 6;

this->dataGridViewTextBoxColumn17->Name = L"dataGridViewTextBoxColumn17";

this->dataGridViewTextBoxColumn17->Width = 125;

//

// dataGridViewTextBoxColumn18

//

this->dataGridViewTextBoxColumn18->HeaderText = L"Псевдокод";

this->dataGridViewTextBoxColumn18->MinimumWidth = 6;

this->dataGridViewTextBoxColumn18->Name = L"dataGridViewTextBoxColumn18";

this->dataGridViewTextBoxColumn18->Width = 76;

//

// label8

//

this->label8->AutoSize = true;

this->label8->Location = System::Drawing::Point(336, 558);

this->label8->Margin = System::Windows::Forms::Padding(2, 0, 2, 0);

this->label8->Name = L"label8";

this->label8->Size = System::Drawing::Size(54, 13);

this->label8->TabIndex = 20;

this->label8->Text = L"Символы";

//

// label9

//

this->label9->AutoSize = true;

this->label9->Location = System::Drawing::Point(11, 426);

this->label9->Margin = System::Windows::Forms::Padding(2, 0, 2, 0);

this->label9->Name = L"label9";

this->label9->Size = System::Drawing::Size(89, 13);

this->label9->TabIndex = 19;

this->label9->Text = L"Знаки операций";

//

// label10

//

this->label10->AutoSize = true;

this->label10->Location = System::Drawing::Point(914, 298);

this->label10->Margin = System::Windows::Forms::Padding(2, 0, 2, 0);

this->label10->Name = L"label10";

this->label10->Size = System::Drawing::Size(63, 13);

this->label10->TabIndex = 24;

this->label10->Text = L"Псевдокод";

//

// psevdo

//

this->psevdo->Location = System::Drawing::Point(917, 313);

this->psevdo->Margin = System::Windows::Forms::Padding(2);

this->psevdo->Multiline = true;

this->psevdo->Name = L"psevdo";

this->psevdo->ScrollBars = System::Windows::Forms::ScrollBars::Vertical;

this->psevdo->Size = System::Drawing::Size(236, 187);

this->psevdo->TabIndex = 23;

//

// label11

//

this->label11->AutoSize = true;

this->label11->Location = System::Drawing::Point(656, 517);

this->label11->Margin = System::Windows::Forms::Padding(2, 0, 2, 0);

this->label11->Name = L"label11";

this->label11->Size = System::Drawing::Size(110, 13);

this->label11->TabIndex = 26;

this->label11->Text = L"Дискрипторный код";

//

// discript

//

this->discript->Location = System::Drawing::Point(659, 540);

this->discript->Margin = System::Windows::Forms::Padding(2);

this->discript->Multiline = true;

this->discript->Name = L"discript";

this->discript->ScrollBars = System::Windows::Forms::ScrollBars::Vertical;

this->discript->Size = System::Drawing::Size(494, 139);

this->discript->TabIndex = 25;

//

// MyForm

//

this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->ClientSize = System::Drawing::Size(1168, 700);

this->Controls->Add(this->label11);

this->Controls->Add(this->discript);

this->Controls->Add(this->label10);

this->Controls->Add(this->psevdo);

this->Controls->Add(this->SimbolGrid);

this->Controls->Add(this->OperGrid);

this->Controls->Add(this->label8);

this->Controls->Add(this->label9);

this->Controls->Add(this->IdentGrid);

this->Controls->Add(this->label6);

this->Controls->Add(this->ConstGrid);

this->Controls->Add(this->kwordGrid);

this->Controls->Add(this->label5);

this->Controls->Add(this->Error);

this->Controls->Add(this->label4);

this->Controls->Add(this->label3);

this->Controls->Add(this->button1);

this->Controls->Add(this->correct);

this->Controls->Add(this->label2);

this->Controls->Add(this->corrected\_code);

this->Controls->Add(this->label1);

this->Controls->Add(this->source);

this->Controls->Add(this->upload\_file);

this->Margin = System::Windows::Forms::Padding(2);

this->Name = L"MyForm";

this->Text = L"Лексический анализатор";

this->Load += gcnew System::EventHandler(this, &MyForm::MyForm\_Load);

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->kwordGrid))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->ConstGrid))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->IdentGrid))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->SimbolGrid))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->OperGrid))->EndInit();

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

private: System::Void MyForm\_Load(System::Object^ sender, System::EventArgs^ e) {

}

private:System::IO::StreamReader^ sr;

private:bool doneReading;

private:

bool probel(const char c) {

if (c == ' ') return true;

}

bool str\_comment(char c) {

if (c == '//') return true;

}

bool str\_perenos(char c) {

if (c == '\r') return true;

}

bool m\_str\_comment1(char c) {

if (c == '/\*') return true;

}

bool m\_str\_comment2(char c) {

if (c == '\*/') return true;

}

bool symbol(char c) {

if (c!= ' ' && c!= '//' && c!='\r' && c != '\n' && c!= '/\*' && c!= '\*/') return true;

}

bool f(char c) {

if (c == '\n') return true;

}

bool end(char c) {

if (c == EOF) return true;

}

int test(char c) {

if (c == ' ' || c=='\t') return 1;

else if (c == '/') return 3;

/\*if (c == '\*') return 4;\*/

else if (c == '\r') return 2;

else if (c != ' ' && c != '//' && c != '\r' && c != '\n' && c != '/\*' && c != '\*/' && c != '\t') return 6;

/\*if (c == EOF) return 7;\*/

}

void clean() {

String^ str = source->Text;

int n = 1;

array<String^>^ word = str->Split();

int i = 0;

str += "\n";

int len = str->Length-1;

int s = 0; // начальное состояние

while (i < len) {

switch (s)

{

case 0:

s = test(str[i]);

corrected\_code->Text += n++; corrected\_code->Text += "\t";

break;

case 1: //пробелы

i++;

s = test(str[i]);

if (s == 6) corrected\_code->Text += " ";

break;

case 2: //перенос строки

if (i + 2 <= len) i += 2;

else i++;

s = test(str[i]);

if (s != 3 && n!=1 && i < len) {

corrected\_code->Text += "\r\n";

corrected\_code->Text += n++; corrected\_code->Text += "\t";

};

if (s != 6 && s!=3) s = 8;

break;

case 8:

i++;

s = test(str[i]);

if (s != 6 && s != 3) s = 8;

break;

case 3: //строчный комментарий

i++;

s = test(str[i]);

if (s == 3) s = 7;

else if (str[i] == '\*') s = 5;

else if (s == 2 ||s == 1) s = 9;

break;

case 7: // строчный комментарий, удаление

i++;

s = test(str[i]);

if (s != 2) s = 7;

else {

if (i + 2 <= len) { i += 2; s = test(str[i]); }

}

break;

case 9:

i+=1;

s = test(str[i]);

if (s == 2) s = 9;

else if (s == 1) s = 9;

else if (s==3) {

if (i + 2 <= len) { i += 2; s = test(str[i]); }

}

break;

case 4: //многострочный комментарий

i++;

s = test(str[i]);

if (s != 3) s = 5;

break;

case 5:

i++;

s = test(str[i]);

if (str[i] != '\*') s = 5;

else s = 4;

break;

case 6: //Символ

corrected\_code->Text += str[i];

i++;

s = test(str[i]);

break;

}

}

}

int Test(char c) {

int s;

if (c == '0' || c == '1' || c == '2' || c == '3' || c == '4' || c == '5' || c == '6' || c == '7' || c == '8' || c == '9') s = 2;

else if (c == 'a') s = 10;

else if (c == 'b') s = 29;

else if (c == 'c') s = 44;

else if (c == 'd') s = 117;

else if (c == 'e') s = 118;

else if (c == 'f') s = 219;

else if (c == 'g') s = 120;

else if (c == 'i') s = 121;

else if (c == 'l') s = 122;

else if (c == 'm') s = 123;

else if (c == 'n') s = 124;

else if (c == 'o') s = 125;

else if (c == 'p') s = 126;

else if (c == 'r') s = 127;

else if (c == 's') s = 128;

else if (c == 't') s = 129;

else if (c == 'u') s = 130;

else if (c == 'v') s = 131;

else if (c == 'w') s = 132;

else if (c == 'x') s = 133;

else s = 217;

return s;

}

int cimbol(char c) {

int p = 0;

String^ l = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890\_";

for (int i = 0; i < l->Length; i++)

{

if (l[i] == c) p=1;

}

return p;

}

int znack(char c) {

int p = 0;

String^ l = "!@#$%^&(){}[];:<>,.'";

l += '"';

for (int i = 0; i < l->Length; i++)

{

if (l[i] == c) p=1;

}

return p;

}

int operation(char c) {

String^ l = "+-/\*<>=";

int p=0;

for (int i = 0; i < l->Length; i++)

{

if (l[i] == c) p = 1;

}

return p;

}

void Kw\_const() {

String^ str = corrected\_code->Text;

int n = 1;

String^ q;

array<String^>^ word = str->Split();

double i = 2; // неучитываем первые 2 символа строки (номер строки и символ табуляции)

/\* str += "\n";\*/

str += " ";

int len = str->Length;

int s = 0; // начальное состояние

String^ number;

String^ w;

int KW = 1, CT = 1, SP = 1, ID = 1, ZO = 1, Cb = 1;

while (i < len) {

switch (s)

{

case 0: // начальное состояние

/\*if (str[i] == '-') s = 1;\*/

s = Test(str[i]);

if (cimbol(str[i]) != 1 && operation(str[i] != 1)) s = 201;

else if (operation(str[i]) == 1) s = 203;

else if (znack(str[i]) == 1) {

s = 201; w += str[i++];

}

else if (str[i] == ' ') {

i++; s = 0;

}

else if (i < len - 1 && str[i] == '\n') { i += 3; n++; s = 0; }

else if (i < len - 1 && str[i] == '\r') { i += 4; n++; s = 0; }

else if (i < len - 1 && str[i] == '\t') { i += 2; n++; s = 0; }

break;

case 1: // отрицательное число (введен минус)

q += str[i];

if (Test(str[i]) == 2) s = 2;

else { s = 201; break; }

i++;

break;

case 2: // вводится цифра

q += str[i++];

if (Test(str[i]) == 2) s = 2;

else if (str[i] == 'e') s = 3;

else if (str[i] == 'l' || str[i] == 'f' || str[i] == 'L' || str[i] == 'F') s = 6;

else if (str[i] == '.') s = 7;

else if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i])==1) s = 119;

else if (cimbol(str[i]) == 1) s = 202; //ошибка индефикаттора

else if (znack(str[i]) == 1) s = 119;

else s = 9;

break;

case 3: // появляется буква е в числе

q += str[i++];

if (str[i] == '-') s = 4;

else if (Test(str[i]) == 2) s = 5;

else if (str[i] == 'e') s = 9;

else if (str[i] == '.') s = 9;

else if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 9;

break;

case 4: //введен минус после е

q += str[i++];

if (Test(str[i]) == 2) s = 5;

else if (str[i] == 'e') s = 9;

else if (str[i] == '-') s = 9;

else if (str[i] == '.') s = 9;

else if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 9;

break;

case 5: // после е или минуса после е введена цифра

q += str[i++];

if (Test(str[i]) == 2) s = 5;

else if (str[i] == 'e') s = 9;

else if (str[i] == '-') s = 9;

else if (str[i] == '.') s = 9;

else if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 119;

break;

case 6: // введено 'l' или 'f'

q += str[i++];

if (str[i] != ' ' && str[i] != '\n' && str[i] != '\r' && str[i] != ',' && znack(str[i]) != 1) s = 9;

else if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 119;

break;

case 7: // точка

q += str[i++];

if (Test(str[i]) == 2) s = 8;

else if (str[i] == '.') s = 9;

break;

case 8: // после точки введена цифра

q += str[i++];

if (Test(str[i]) == 2) s = 8;

else if (str[i] == 'e') s = 3;

else if (str[i] == 'l' || str[i] == 'f' || str[i] == 'L' || str[i] == 'F') s = 6;

else if (str[i] == '.') s = 9;

else if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 119;

break;

case 9: // ошибка константы

q += str[i++];

if (str[i] != ' ' && str[i] != '\n' && str[i] != '\r' && str[i] != ',' && znack(str[i]) != 1) s = 9;

else s = 218;

break;

case 10: // a

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'l') s = 11;

else if (str[i] == 'n') s = 12;

else if (str[i] == 's') s = 13;

else if (str[i] == 'u') s = 14;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 11: //al

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'i') s = 15;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 15: //ali

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'g') s = 16;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 16: //alig

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'n') s = 17;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 17: //align

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'a') s = 18;

else if (str[i] == 'o') s = 19;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 18: //aligna

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 's') s = 20;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 20: //alignas

if (cimbol(str[i]) == 1) {

i++;

if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 116;

else s = 230;

}

else s = 202; //ошибка идентификатора

break;

case 19: //aligno

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'f') s = 21;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 21: //alignof

if (cimbol(str[i]) == 1) {

i++;

if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 116;

else s = 230;

}

else s = 202; //ошибка идентификатора

break;

case 12: //an

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'd') s = 22;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 22: //and

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 116;

else s = 230;

}

else s = 202; //ошибка идентификатора

break;

case 23: //and\_

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'e') s = 24;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 24: //and\_e

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'q') s = 25;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 25: //and\_eq

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 116;

else s = 230;

}

else s = 202; //ошибка идентификатора

break;

case 13: //as

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'm') s = 26;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 26: //asm

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 116;

else s = 230;

}

else s = 202; //ошибка идентификатора

break;

case 14: //au

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 't') s = 27;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 27: //aut

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'o') s = 28;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 28: //auto

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 116;

else s = 230;

}

else s = 202; //ошибка идентификатора

break;

case 29: //b

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'i') s = 30;

else if (str[i] == 'o') s = 31;

else if (str[i] == 'r') s = 32;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 30: //bi

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 't') s = 33;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 33: //bit

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'a') s = 34;

else if (str[i] == 'o') s = 35;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 34: //bita

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'n') s = 36;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 36: //bitan

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'd') s = 37;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 37: //bitand

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 116;

else s = 230;

}

else s = 202; //ошибка идентификатора

break;

case 35: //bito

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'r') s = 38;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 38: //bitor

if (cimbol(str[i]) == 1) {

w += str[i++];;

if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 116;

else s = 230;

}

else s = 202; //ошибка идентификатора

break;

case 31: //bo

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'o') s = 39;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 39: //boo

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'l') s = 40;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 40: //bool

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 116;

else s = 230;

}

else s = 202; //ошибка идентификатора

break;

case 32: //br

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'e') s = 41;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 41: //bre

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'a') s = 42;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 42: //brea

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'k') s = 43;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 43: //break

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 116;

else s = 230;

}

else s = 202; //ошибка идентификатора

break;

case 44: //c

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'a') s = 45;

else if (str[i] == 'h') s = 46;

else if (str[i] == 'l') s = 47;

else if (str[i] == 'o') s = 48;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 45: //ca

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 's') s = 49;

else if (str[i] == 't') s = 50;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 49: //cas

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'e') s = 51;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 51: //case

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 116;

else s = 230;

}

else s = 202; //ошибка идентификатора

break;

case 50: //cat

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'c') s = 52;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 52: //catc

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'h') s = 53;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 53: //catch

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 116;

else s = 230;

}

else s = 202; //ошибка идентификатора

break;

case 46: //ch

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'a') s = 54;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 54: //cha

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'r') s = 55;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 55: //char

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 116;

else if (str[i] == '8') s = 56;

else if (str[i] == '1') s = 57;

else if (str[i] == '3') s = 58;

else s = 230;

}

else s = 202; //ошибка идентификатора

break;

case 56: //char8

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == '\_') s = 59;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 59: //char8\_

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 't') s = 60;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 60: //char8\_t

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 116;

else s = 230;

}

else s = 202; //ошибка идентификатора

break;

case 57: //char1

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == '6') s = 61;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 61: //char16

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == '\_') s = 62;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 62: //char16\_

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 't') s = 63;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 63: //char16\_t

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 116;

else s = 230;

}

else s = 202; //ошибка идентификатора

break;

case 58: //char3

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == '2') s = 64;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 64: //char32

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == '\_') s = 65;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 65: //char32\_

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 't') s = 66;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 66: //char32\_t

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 116;

else s = 230;

}

else s = 202; //ошибка идентификатора

break;

case 47: //cl

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'a') s = 67;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 67: //cla

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 's') s = 68;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 68: //clas

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 's') s = 69;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 69: //class

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 116;

else s = 230;

}

else s = 202; //ошибка идентификатора

break;

case 48: //co

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'm') s = 70;

else if (str[i] == 'n') s = 71;

else if (str[i] == '\_') s = 72;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 70: //com

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'p') s = 73;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 73: //comp

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'l') s = 74;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 74: //compl

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 116;

else s = 230;

}

else s = 202; //ошибка идентификатора

break;

case 71: //con

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'c') s = 75;

else if (str[i] == 's') s = 76;

else if (str[i] == 'n') s = 77;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 75: //conc

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'e') s = 78;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 78: //conce

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'p') s = 79;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 79: //concep

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 't') s = 80;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 80: //concept

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 116;

else s = 230;

}

else s = 202; //ошибка идентификатора

break;

case 76: //cons

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 't') s = 81;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 81: //const

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 116;

else if (str[i] == '\_') s = 82;

else if (str[i] == 'e') s = 83;

else if (str[i] == 'i') s = 84;

else s = 230;

}

else s = 202; //ошибка идентификатора

break;

case 82: //const\_

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'c') s = 85;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 85: //const\_c

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'a') s = 86;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 86: //const\_ca

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 's') s = 87;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 87: //const\_cas

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 't') s = 88;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 88: //const\_cast

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 116;

else s = 230;

}

else s = 202; //ошибка идентификатора

break;

case 83: //conste

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'v') s = 89;

else if (str[i] == 'x') s = 90;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 89: //constev

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'a') s = 91;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 91: //consteva

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'l') s = 92;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 92: //consteval

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 116;

else s = 230;

}

else s = 202; //ошибка идентификатора

break;

case 90: //constex

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'p') s = 93;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 93: //constexp

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'r') s = 94;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 94: //constexpr

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 116;

else s = 230;

}

else s = 202; //ошибка идентификатора

break;

case 84: //consti

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'n') s = 95;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 95: //constin

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'i') s = 96;

else if (str[i] == 'u') s = 97;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 96: //constini

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 't') s = 98;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 98: //constinit

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 116;

else s = 230;

}

else s = 202; //ошибка идентификатора

break;

case 97: //constinu

if (cimbol(str[i]) == 1) {

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'e') s = 99;

else s = 217;

}

}

else s = 202; //ошибка идентификатора

break;

case 99: //constinue

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 116;

else s = 230;

}

else s = 202; //ошибка идентификатора

break;

case 72: //co\_

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'a') s = 100;

else if (str[i] == 'r') s = 101;

else if (str[i] == 'y') s = 102;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 100: //co\_a

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'w') s = 103;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 103: //co\_aw

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'a') s = 104;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 104: //co\_awa

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'i') s = 105;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 105: //co\_awai

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 't') s = 106;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 106: //co\_await

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 116;

else s = 230;

}

else s = 202; //ошибка идентификатора

break;

case 101: //co\_r

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'e') s = 107;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 107: //co\_re

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 't') s = 108;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 108: //co\_ret

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'u') s = 109;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 109: //co\_retu

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'r') s = 110;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 110: //co\_retur

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'n') s = 111;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 111: //co\_return

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 116;

else s = 230;

}

else s = 202; //ошибка идентификатора

break;

case 102: //co\_y

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'i') s = 112;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 112: //co\_yi

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'e') s = 113;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 113: //co\_yie

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'l') s = 114;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 114: //co\_yiel

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'd') s = 115;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 115: //co\_yield

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 116;

else s = 230;

}

else s = 202; //ошибка идентификатора

break;

case 117: //d

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'e') s = 134;

else if (str[i] == 'o') s = 135;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 134: //de

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'c') s = 138;

else if (str[i] == 'f') s = 139;

else if (str[i] == 'l') s = 140;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 138: //dec

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'l') s = 141;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 141: //decl

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 't') s = 142;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 142: //declt

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'y') s = 143;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 143: //declty

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'p') s = 144;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 144: //decltyp

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'e') s = 145;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 145: //decltype

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 116;

else s = 230;

}

else s = 202; //ошибка идентификатора

break;

case 139: //def

if (cimbol(str[i]) == 1) {

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'a') s = 146;

else s = 217;

}

}

else s = 202; //ошибка идентификатора

break;

case 146: //defa

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'u') s = 147;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 147: //defau

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'l') s = 148;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 148: //defaul

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 't') s = 149;

else s = 116;

}

else s = 202; //ошибка идентификатора

break;

case 149: //default

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 116;

else s = 230;

}

else s = 202; //ошибка идентификатора

break;

case 140: //del

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'e') s = 150;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 150: //dele

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 't') s = 151;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 151: //delet

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'e') s = 152;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 152: //delete

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 116;

else s = 230;

}

else s = 202; //ошибка идентификатора

break;

case 135: //do

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 116;

else if (str[i] == 'u') s = 136;

else s = 230;

}

else s = 202; //ошибка идентификатора

break;

case 136: //dou

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'b') s = 153;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 153: //doub

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'l') s = 154;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 154: //doubl

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'e') s = 155;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 155: //double

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 116;

else s = 230;

}

else s = 202; //ошибка идентификатора

break;

case 118: //e

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'l') s = 156;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 156: //el

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 's') s = 157;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 157: //els

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'e') s = 158;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 158: //else

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 116;

else s = 230;

}

else s = 202; //ошибка идентификатора

break;

case 219: //f

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'a') s = 159;

else if (str[i] == 'l') s = 160;

else if (str[i] == 'o') s = 161;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 159: //fa

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'l') s = 162;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 162: //fal.

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 's') s = 163;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 163: //fals

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'e') s = 164;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 164: //false

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 116;

else s = 230;

}

else s = 202; //ошибка идентификатора

break;

case 160: //fl

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'o') s = 165;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 165: //flo

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'a') s = 166;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 166: //floa

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 't') s = 167;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 167: //float

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 116;

else s = 230;

}

else s = 202; //ошибка идентификатора

break;

case 161: //fo

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'r') s = 168;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 168: //for

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 116;

else s = 230;

}

else s = 202; //ошибка идентификатора

break;

case 120: //g

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'o') s = 169;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 169: //go

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 't') s = 170;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 170://got

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'o') s = 171;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 171: //goto

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 116;

else s = 230;

}

else s = 202; //ошибка идентификатора

break;

case 121://i

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'f') s = 172;

else if (str[i] == 'n') s = 173;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 172: //if

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 116;

else s = 230;

}

else s = 202; //ошибка идентификатора

break;

case 173: //in

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 't') s = 174;

else s = 116;

}

else s = 202; //ошибка идентификатора

break;

case 174: //int

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 116;

else s = 230;

}

else s = 202; //ошибка идентификатора

break;

case 122: //l

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'o') s = 175;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 175: //lo

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'n') s = 176;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 176: //lon

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'g') s = 177;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 177: //long

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 116;

else s = 230;

}

else s = 202; //ошибка идентификатора

break;

case 123: //n

s = 217;

case 124: //o

s = 217;

case 126: //p

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'r') s = 178;

else if (str[i] == 'u') s = 184;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 178: //pr

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'i') s = 179;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 179: //pri

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'v') s = 180;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 180: //priv

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'a') s = 181;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 181: //priva

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 't') s = 182;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 182: //privat

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'e') s = 183;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 183: //private

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 116;

else s = 230;

}

else s = 202; //ошибка идентификатора

break;

case 184: //pu

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'b') s = 185;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 185: //pub

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'l') s = 186;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 186: //publ

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'i') s = 187;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 187: //publi

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'c') s = 188;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 188: //public

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 116;

else s = 230;

}

else s = 202; //ошибка идентификатора

break;

case 127://r

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'u') s = 189;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 189://re

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 't') s = 190;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 190://ret

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'u') s = 191;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 191://retu

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'r') s = 192;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 192://retur

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'n') s = 193;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 193: //return

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 116;

else s = 230;

}

else s = 202; //ошибка идентификатора

break;

case 125:

s = 217;

break;

case 128:

s = 217;

break;

case 129:

s = 217;

break;

case 130:

s = 217;

break;

case 131: //v

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'o') s = 194;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 194://vo

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'i') s = 195;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 195://voi

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'd') s = 196;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 196: //void

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 116;

else s = 230;

}

else s = 202; //ошибка идентификатора

break;

case 132://w

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'h') s = 197;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 197://wh

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'i') s = 198;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 198://whi

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'l') s = 199;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 199://whil

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == 'e') s = 200;

else s = 217;

}

else s = 202; //ошибка идентификатора

break;

case 200: //while

if (cimbol(str[i]) == 1) {

w += str[i++];

if (str[i] == ' ' || str[i] == '\n' || str[i] == '\r' || str[i] == ',' || znack(str[i]) == 1) s = 116;

else s = 230;

}

else s = 202; //ошибка идентификатора

break;

case 201: //символ

/\*w += str[i++];\*/

break;

case 202: //ошибка в идентификатторе

break;

case 203:

if (str[i] == '+') s = 204;

else if (str[i] == '-') s = 205;

else if (str[i] == '=') s = 206;

else if (str[i] == '\*') s = 207;

else if (str[i] == '/') s = 208;

else if (str[i] == '<') s = 223;

else if (str[i] == '>') s = 224;

else s = 201;

w += str[i++];

break;

case 204:

if (str[i] == '+') { s = 210; } //+

else s = 209;

/\*else if (cimbol(str[i])==1)\*/

/\*w += str[i];\*/

break;

case 205: //-

if (str[i] == '-') s = 210;

else s = 209;

break;

case 206: //=

if (str[i] == '=') s = 210;

else s = 209;

break;

case 207: //\*

w += str[i];

i++;

if (cimbol(str[i]) != 1 && str[i] != ' ') s = 221;

else s = 222;

break;

case 208: // /

i++;

if (cimbol(str[i]) != 1 && str[i] != ' ') s = 221;

else s = 222;

break;

case 133:

s = 217;

break;

case 116: //разделитель ключевое слово

break;

case 217: //Идентификатnор

break;

case 218: // ошибка в константе

break;

case 119: // константа

break;

case 221: //error operation

break;

case 209:

/\*w += str[i];\*/

w += str[i++];

if (cimbol(str[i]) != 1) s = 221;

else s = 222;

break;

case 210:

w += str[i++];

if (str[i] != ' ' && str[i] != '\n' && str[i] != ';' && znack(str[i]) != 1) s = 221;

else s = 222;

case 222: //out

break;

case 223: //<

/\*w += str[i++];\*/

if (str[i] == ' ') s = 222;

else if (str[i] == '<') s = 201;

else s = 201;

break;

case 224: //>

/\*w += str[i++];\*/

if (str[i] == ' ') s = 201;

else if (str[i] == '>') s = 201;

else s = 201;

break;

}

if (s == 116) {

int ctest = -1;

while (str[i] != ' ' && str[i] != '\r' && str[i] != '\n' && cimbol(str[i]) == 1) {

w += str[i]; i++;

if (i == len) break;

}

//if (i != len && znack(str[i]) == 1) {

//

// //for (int j = 0; j < Cb-1; j++) if (SimbolGrid[1, j]->Value->ToString() == ""+str[i]) ctest = j + 1;

// //if (ctest == -1) {

// // SimbolGrid->Rows->Add(Cb, str[i], str[i]); // запись в таблицу

// // Cb++;

// // ctest = Cb;

// //}

// /\*s = 0;\*/

//

//}

if (i != len) s = 217;

int test = -1;

for (int j = 0; j < KW-1; j++) if (kwordGrid[1, j]->Value->ToString() == w) test = j + 1;

if (test == -1) {

kwordGrid->Rows->Add(KW, w, w); // запись в таблицу

psevdo->Text += w + " ";

discript->Text += "(10." + (KW++) + ")";

}

else {

psevdo->Text += w + " ";

discript->Text += "(10." + (test) + ")";

}

s = 0; w = "";

}

else if (s == 217) {

int ctest = -1;

while (str[i] != ' ' && str[i] != '\r' && str[i] != '\n' && cimbol(str[i]) == 1) {

w += str[i]; i ++ ;

if (i == len) break;

}

if ( i != len && znack(str[i]) == 1) {

}

int test = -1;

for (int j = 0; j < ID-1; j++)

{

if (IdentGrid[1, j]->Value->ToString() == ""+w) test = j + 1;

}

if (test == -1) {

IdentGrid->Rows->Add(ID, w, "id" +ID); // запись в таблицу

test = ID++;

}

psevdo->Text += "id" + test + " ";

discript->Text += "(30." + (test) + ")";

/\*if (i!=len && str[i] == '\r' && (i + 3) < len - 1) i += 4;\*/

s = 0; w = "";

}

else if (s == 218) {

Error->Text += n + " " + q + "- ошибка в константе " + "\r\n";

if (i != len && str[i] == '\r' && str[i] != '\n' && (i + 3) < len - 1) i += 4;

s = 0; q = ""; w = "";

}

else if (s == 119) {

int ctest = -1;

while (str[i] != ' ' && str[i] != '\r' && str[i] != '\n' && cimbol(str[i]) == 1) {

w += str[i]; i++;

if (i == len) break;

}

if (i != len && znack(str[i]) == 1) {

}

int test = -1;

for (int j = 0; j < CT - 1; j++)

{

if (ConstGrid[1, j]->Value->ToString() == q) test = j + 1;

}

if (test == -1) {

ConstGrid->Rows->Add(CT, q, "const" + CT); // запись в таблицу

test = CT++;

}

psevdo->Text += "const" + test + " ";

discript->Text += "(30." + (test)+")";

/\*if (i != len && str[i] == '\r' && str[i] != '\n' && (i + 3) < len - 1) i += 4;\*/

s = 0; w = ""; q = "";

}

else if (s == 201) {

/\*w += str[i++];\*/

int test = -1;

if (str[i] == ' ' || str[i] == '\n' || cimbol(str[i])==1 || str[i] == '\r') {

for (int j = 0; j < Cb - 1; j++) if (SimbolGrid[1, j]->Value->ToString() == "" + w) test = j + 1;

if (test == -1) {

SimbolGrid->Rows->Add(Cb, w, w); // запись в таблицу

Cb++;

test = Cb;

}

psevdo->Text += w + " ";

discript->Text += "(40." + (test)+")";

/\*if (i != len && str[i] == '\r' && (i + 3) < len - 1) i += 4;\*/

s = 0; w = ""; q = "";

}

else {

w += str[i];

while (w[0] == str[i] && i<len) {

w += str[i]; i++;

}

Error->Text += n + " " + w + "Ошибка в знаках" + "\r\n";

/\*if (i != len && str[i] == '\r' && (i + 3) < len - 1) i += 4;\*/

}

s = 0; w = "";

}

else if (s == 202) {

while (str[i] != ' ' && str[i] != '\r' && i != len && str[i] != '\n') {

w += str[i]; i++;

if (i == len) break;

}

Error->Text += n + " " + w + "Ошибка в идентификаторе"+"\r\n";

/\*if (i != len && str[i] == '\r' && (i + 3) < len - 1) i += 4;\*/

s = 0; w = "";

}

else if (s == 230) {

if (operation(str[i]) == 1) s = 0;

else {

while (str[i] != ' ' && str[i] != '\r' && i != len && str[i] != '\n' && znack(str[i])!=1) {

w += str[i]; i++;

if (i == len) break;

}

Error->Text += n + " " + w + "Ошибка в константе" + "\r\n";

/\*if (i != len && str[i] == '\r' && (i + 3) < len - 1) i += 4;\*/

}

s = 0; w = "";

}

else if (s == 221) {

while (operation(str[i]) == 1 && w[0]==str[i]) {

w += str[i]; i++;

}

if (w->Length != 1) {

Error->Text += n + " " + w + "Ошибка в операторе" + "\r\n";

/\*if (i != len && str[i] == '\r' && str[i] != '\n' && (i + 3) < len - 1) i += 4;\*/

s = 0; w = "";

}

/\*else s = ;\*/

}

else if (s == 222) {

if (cimbol(str[i]) == 1 || str[i] == ' ' || str[i] == '\n' || znack(str[i])==1) {

int test = -1;

for (int j = 0; j < ZO-1; j++)

{

if (OperGrid[1, j]->Value->ToString() ==w) test = j + 1;

}

if (test == -1) {

OperGrid->Rows->Add(ZO, w, w); // запись в таблицу

test = ZO++;

}

psevdo->Text += w + " ";

discript->Text += "(00." + (test)+")";

/\*if (i != len && str[i] == '\r' && str[i] != '\n' && (i + 3) < len - 1) i += 4;\*/

s = 0; w = "";

}

else if (w == "-" && str[i] == '>') {

int test = -1;

for (int j = 0; j < Cb - 1; j++) if (SimbolGrid[1, j]->Value->ToString() == "" + w) test = j + 1;

if (test == -1) {

SimbolGrid->Rows->Add(Cb, w, w); // запись в таблицу

Cb++;

test = Cb;

}

psevdo->Text += str[i] + " ";

discript->Text += "(40." + (test)+")";

if (i != len && str[i] == '\r' && str[i] != '\n' && (i + 3) < len - 1) i += 4;

s = 0; q = "";

test = -1;

for (int j = 0; j < Cb - 1; j++) if (SimbolGrid[1, j]->Value->ToString() == "" + str[i]) test = j + 1;

if (test == -1) {

SimbolGrid->Rows->Add(Cb, str[i], str[i]); // запись в таблицу

Cb++;

test = Cb;

}

psevdo->Text += str[i] + " ";

discript->Text += "(40." + (test)+")";

/\*if (i != len && str[i] == '\r' && str[i] != '\n' && (i + 3) < len - 1) i += 4;\*/

s = 0; q = "";

}

}

}}

private: System::Void upload\_file\_Click(System::Object^ sender, System::EventArgs^ e) {

source->Text = "";

OpenFileDialog^ file = gcnew OpenFileDialog();

file->Filter = "Text Files|\*.txt|All Files|\*.\*";

if (this->openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK)

{

this->source->Text += this->openFileDialog1->FileName;

this->source->Text = System::IO::File::ReadAllText(this->source->Text);

}

}

private: System::Void correct\_Click(System::Object^ sender, System::EventArgs^ e) {

corrected\_code->Text = "";

Error->Text = "";

psevdo->Text = "";

IdentGrid->Rows->Clear();

OperGrid->Rows->Clear();

kwordGrid->Rows->Clear();

SimbolGrid->Rows->Clear();

ConstGrid->Rows->Clear();

discript->Text = "";

clean();

Kw\_const();

}

private: System::Void label2\_Click(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) {

source->Text = "";

corrected\_code->Text = "";

Error->Text = "";

psevdo->Text = "";

IdentGrid->Rows->Clear();

OperGrid->Rows->Clear();

kwordGrid->Rows->Clear();

SimbolGrid->Rows->Clear();

ConstGrid->Rows->Clear();

discript->Text = "";

}

};}

Приложение 3.

Руководство пользователя

1. Общие сведения о программе

Исполнительный файл: ЛексическийАнализатор.exe. Данная программа предназначена для анализа, введенного пользователем кода.

1. Описание установки

Установка программы не требуется.

1. Описание запуска

Двойным нажатием левой кнопки мыши запустить файл смотрим.exe

1. Инструкция по работе

4.1 Описание возможностей программы

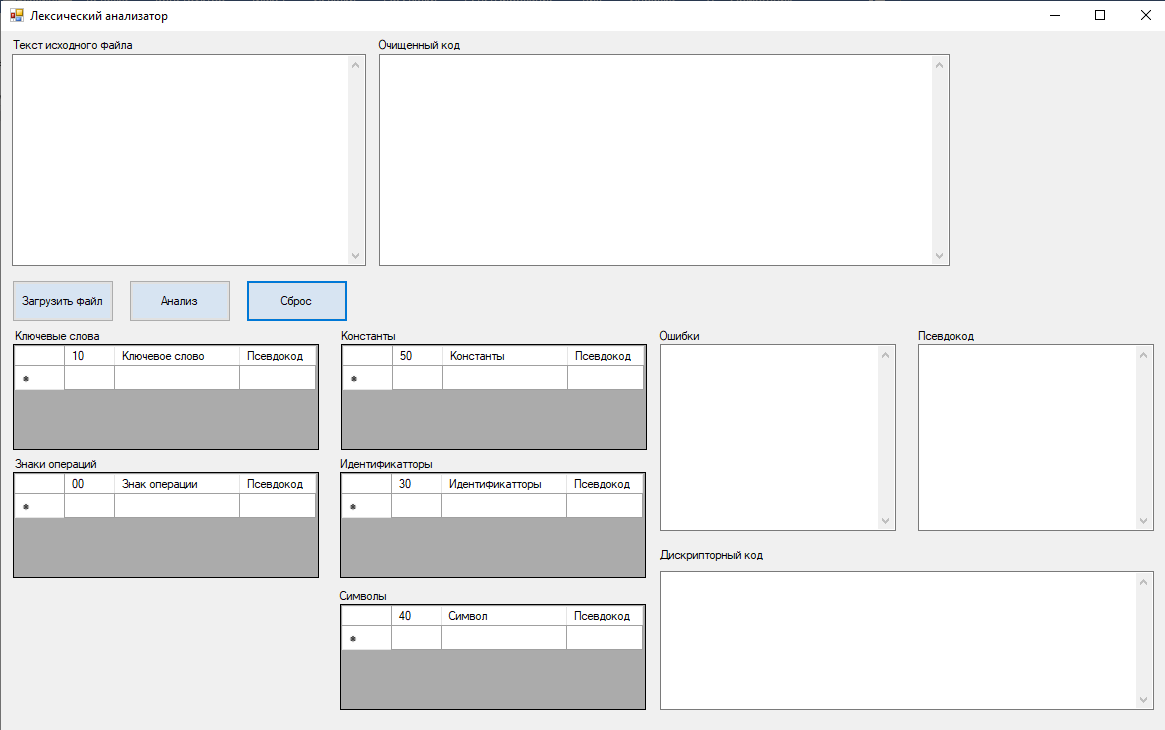


Рис. П3.1. Запуск приложения

Для того, чтобы выгрузить информацию из файла, необходимо нажать на кнопку «Загрузить файл». Вместо того, чтобы загружать текст с файла, пользователь может ввести его в поле «исходный текс» с клавиатуры.

Затем необходимо нажать на кнопку «Анализ». Программа проанализирует введенный кода (рис. П3.2)

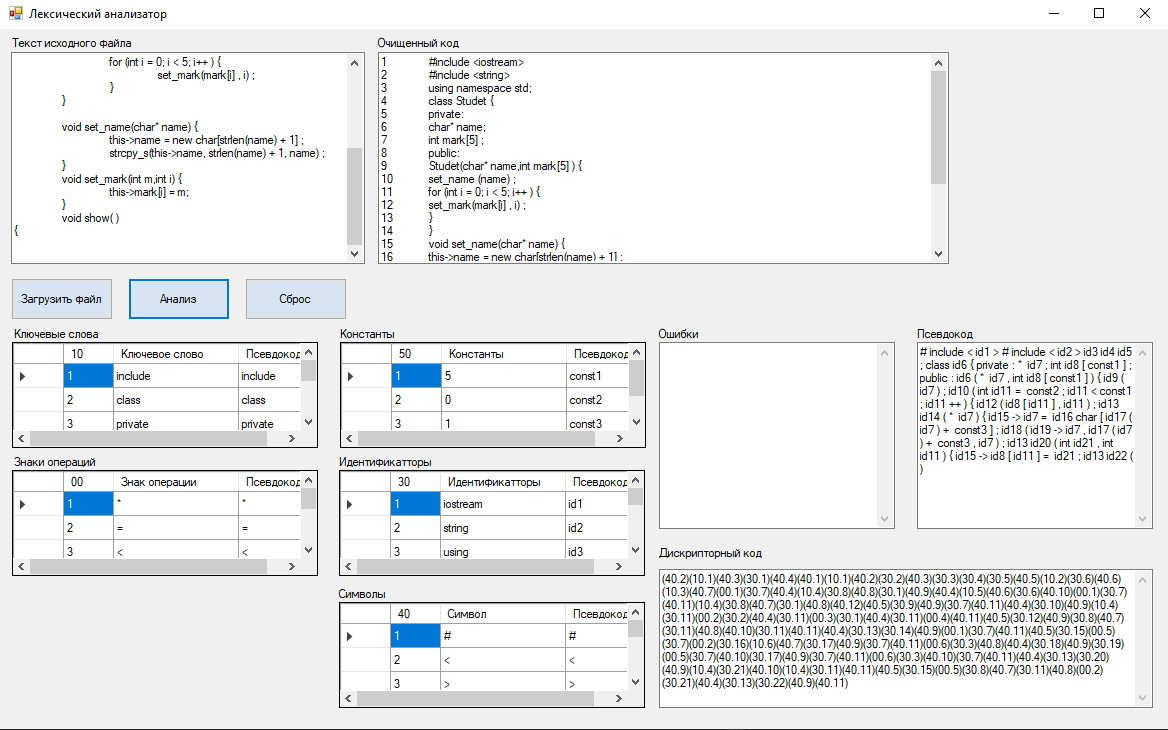


Рис. П3.2. Анализ

Чтобы очистить все поля пользователю необходимо кликнуть на кнопку «сброс», после работу можно производить заново. Чтобы прекратить работу программы нужно закрыть рабочее окно.

# Приложение 4.

Таблица переходов

Таблица П4.1

Таблица переходов

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Входной символ/  состояние | a | b | c | d | e | f | g | i | l |
| S0 | S10 | S29 | S44 | S117 | S118 | S219 | S120 | S121 | S122 |
| S1 | - | - | - | - | - | - | - | - | - |
| S2 | - | - | - | - | - | - | - | - | - |
| S3 | - | - | - | - | - | - | - | - | - |
| S4 | - | - | - | - | - | - | - | - | - |
| S5 | - | - | - | - | - | - | - | - | - |
| S6 | - | - | - | - | - | - | - | - | - |
| S7 | - | - | - | - | - | - | - | - | - |
| S8 | - | - | - | - | - | - | - | - | - |
| S10 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | S11 |
| S11 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | S15 | **S217** |
| S15 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | S16 | **S217** | **S217** |
| S16 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S17 | S18 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S18 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S20 | - | - | - | - | - | - | - | - | - |
| S19 | **S217** | **S217** | **S217** | **S217** | **S217** | S21 | **S217** | **S217** | **S217** |
| S21 | - | - | - | - | - | - | - | - | - |
| S12 | **S217** | **S217** | **S217** | S22 | **S217** | **S217** | **S217** | **S217** | **S217** |
| S22 | - | - | - | - | - | - | - | - | - |
| S23 | **S217** | **S217** | **S217** | **S217** | S24 | **S217** | **S217** | **S217** | **S217** |
| S24 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S25 | - | - | - | - | - | - | - | - | - |
| S13 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S26 | - | - | - | - | - | - | - | - | - |
| S14 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S27 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S28 | - | - | - | - | - | - | - | - | - |
| S29 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | S30 | **S217** |
| S30 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S33 | S34 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S34 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S36 | **S217** | **S217** | **S217** | S37 | **S217** | **S217** | **S217** | **S217** | **S217** |
| S37 | - | - | - | - | - | - | - | - | - |
| S35 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S38 | - | - | - | - | - | - | - | - | - |
| S39 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | S40 |
| S40 | - | - | - | - | - | - | - | - | - |
| S32 | **S217** | **S217** | **S217** | **S217** | S41 | **S217** | **S217** | **S217** | **S217** |
| S41 | S42 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |

Продолжение таблица П4.1

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Входной символ/  состояние | a | b | c | d | e | f | g | i | l |
| S42 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S43 | - | - | - | - | - | - | - | - | - |
| S44 | S45 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | S47 |
| S45 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S49 | **S217** | **S217** | **S217** | **S217** | S51 | **S217** | **S217** | **S217** | **S217** |
| S51 | - | - | - | - | - | - | - | - | - |
| S50 | **S217** | **S217** | S52 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S52 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S53 | - | - | - | - | - | - | - | - | - |
| S46 | S54 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S54 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S55 | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** |
| S56 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S59 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S60 | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** |
| S47 | S67 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S67 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S68 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S69 | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** |
| S48 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S70 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S73 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | S74 |
| S71 | **S217** | **S217** | S75 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S75 | **S217** | **S217** | **S217** | **S217** | S78 | **S217** | **S217** | **S217** | **S217** |
| S78 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S79 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S80 | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** |
| S76 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S81 | **-** | **-** | **-** | **-** | S83 | **-** | **-** | S84 | **-** |
| S82 | **S217** | **S217** | S85 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S85 | S86 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S86 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S87 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S88 | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** |
| S83 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S89 | S91 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S91 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | S92 |
| S90 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S93 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S94 | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** |
| S84 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S95 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | S96 | **S217** |
| S96 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S98 | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** |
| S97 | **S217** | **S217** | **S217** | **S217** | S99 | **S217** | **S217** | **S217** | **S217** |

Продолжение таблица П4.1

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Входной символ/  состояние | a | b | c | d | e | f | g | i | l |
| S99 | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** |
| S72 | S100 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S100 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S103 | S104 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S104 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | S105 | **S217** |
| S105 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S106 | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** |
| S101 | **S217** | **S217** | **S217** | **S217** | S107 | **S217** | **S217** | **S217** | **S217** |
| S107 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S108 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S109 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S110 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S111 | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** |
| S102 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | S112 | **S217** |
| S112 | **S217** | **S217** | **S217** | **S217** | S113 | **S217** | **S217** | **S217** | **S217** |
| S113 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | S114 |
| S114 | **S217** | **S217** | **S217** | S115 | **S217** | **S217** | **S217** | **S217** | **S217** |
| S115 | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** |
| S117 | **S217** | **S217** | **S217** | **S217** | S134 | **S217** | **S217** | **S217** | **S217** |
| S134 | **S217** | **S217** | S138 | **S217** | **S217** | S139 | **S217** | **S217** | S140 |
| S138 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | S141 |
| S141 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S142 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S143 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S144 | **S217** | **S217** | **S217** | **S217** | S145 | **S217** | **S217** | **S217** | **S217** |
| S145 | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** |
| S146 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S147 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | S148 |
| S148 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S149 | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** |
| S140 | **S217** | **S217** | **S217** | **S217** | S150 | **S217** | **S217** | **S217** | **S217** |
| S150 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S151 | **S217** | **S217** | **S217** | **S217** | S152 | **S217** | **S217** | **S217** | **S217** |
| S152 | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** |
| S135 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S136 | **S217** | S153 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S153 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | S154 |
| S154 | **S217** | **S217** | **S217** | **S217** | S155 | **S217** | **S217** | **S217** | **S217** |
| S155 | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** |
| S118 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | S156 |
| S156 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S157 | **S217** | **S217** | **S217** | **S217** | S158 | **S217** | **S217** | **S217** | **S217** |
| S158 | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** |
| S219 | S159 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | S160 |
| S159 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | S162 |

Продолжение таблица П4.1

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Входной символ/  состояние | a | b | c | d | e | f | g | i | l |
| S162 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S163 | **S217** | **S217** | **S217** | **S217** | S164 | **S217** | **S217** | **S217** | **S217** |
| S164 | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** |
| S160 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S165 | S166 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S166 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S167 | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** |
| S161 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S168 | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** |
| S169 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S170 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S171 | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** |
| S121 | **S217** | **S217** | **S217** | **S217** | **S217** | S172 | **S217** | **S217** | **S217** |
| S172 | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** |
| S173 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S174 | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** |
| S122 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S175 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S176 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | S177 | **S217** | **S217** |
| S177 | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** |
| S123 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S124 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S126 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S178 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | S179 | **S217** |
| S179 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S180 | S181 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S182 | **S217** | **S217** | **S217** | **S217** | S183 | **S217** | **S217** | **S217** | **S217** |
| S183 | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** |
| S184 | **S217** | S185 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S185 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | S186 |
| S186 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | S187 |  |
| S187 | **S217** | **S217** | S188 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S188 | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** |
| S127 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S189 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S190 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S191 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S192 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S193 | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** |
| S125 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S128 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S129 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S130 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S131 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S194 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | S195 | **S217** |

Продолжение таблица П4.1

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Входной символ/  состояние | a | b | c | d | e | f | g | i | l |
| S195 | **S217** | **S217** | **S217** | S196 | **S217** | **S217** | **S217** | **S217** | **S217** |
| S196 | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** |
| S132 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S197 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | S198 | **S217** |
| S198 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | S199 |
| S199 | **S217** | **S217** | **S217** | **S217** | S200 | **S217** | **S217** | **S217** | **S217** |
| S200 | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** |
| S203 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S204 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S205 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S206 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S207 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S208 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S133 | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** | **S217** |
| S209 | **S222** | **S222** | **S222** | **S222** | **S222** | **S222** | **S222** | **S222** | **S222** |
| S210 | **S222** | **S222** | **S222** | **S222** | **S222** | **S222** | **S222** | **S222** | **S222** |
| S223 | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** |
| S224 | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** |

# Приложение 5.

# Граф лексического анализатора

