МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное

образовательное учреждение высшего образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт (факультет) \_\_\_\_\_\_\_\_\_Институт информационных технологий\_\_\_\_\_\_\_\_\_

Кафедра \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_МПО ЭВМ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КУРСОВАЯ РАБОТА

по модулю Информатика

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

на тему Разработка алгоритмического обеспечения и построение лексического

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

анализатора компилятора

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Выполнил студент группы

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1ПИб-02-1оп-22

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

направления подготовки (специальности)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Программная инженерия

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*шифр, наименование*

\_\_\_\_Микуцких Григорий Андреевич\_\_\_\_

*фамилия, имя, отчество*

Руководители

Ганичева Оксана Георгиевна

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*фамилия, имя, отчество*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

доцент

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*должность*

Пышницкий Константин Михайлович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*фамилия, имя, отчество*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

старший преподаватель

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*должность*

Дата представления работы

«\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20 \_\_\_ г.

Заключение о допуске к защите

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

количество баллов

Подпись преподавателя\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Череповец, \_2024\_

*год*

Аннотация

Курсовая работа посвящена изучению этапов лексического анализа компиляторов.

В ходе работы построен конечный автомат (КА), моделирующий работу этапа лексического анализа компилятора, по КА построена соответствующая ему регулярная грамматика, разработана программа, моделирующая работу лексического анализатора, построена КС-грамматика, описывающая синтаксис инструкций объединения и структуры («union» и «struct») в языке C++.

Оглавление

[Введение 4](#_Toc169028519)

[1. Описание предметной области 5](#_Toc169028520)

[2. Основная часть 11](#_Toc169028521)

[2.1 Постановка задачи 11](#_Toc169028522)

[2.2 Построение конечного автомата 12](#_Toc169028523)

[2.3 Построение регулярной грамматики по конечному автомата 23](#_Toc169028524)

[2.4 Построение КС-грамматики 44](#_Toc169028525)

[3. Логическое проектирование 48](#_Toc169028526)

[3.1 Блок-схемы алгоритмов 48](#_Toc169028527)

[3.2 Словесное описание алгоритмов 52](#_Toc169028528)

[3.3 Оценка сложности алгоритма 54](#_Toc169028529)

[4. Физическое проектирование 55](#_Toc169028530)

[5. Проектирование интерфейса 58](#_Toc169028531)

[6. Тестирование программного обеспечения 59](#_Toc169028532)

[Заключение 61](#_Toc169028533)

[Список литературы 62](#_Toc169028534)

[Приложение 1. Техническое задание 63](#_Toc169028535)

[Приложение 2. Руководство пользователя 71](#_Toc169028536)

[Приложение 3. Текст программы 73](#_Toc169028537)

[Приложение 4. Граф конечного автомата 88](#_Toc169028538)

# Введение

Для того чтобы эффективно использовать и реализовывать языки программирования, необходимо хорошо знать фундаментальные понятия, лежащие в основе их построения.

Построение трансляторов является областью информатики, глубокие теоретические результаты которой привели к решению важных практических проблем. Эта область объединяет идеи, методы и алгоритмы синтаксического и семантического анализа искусственных языков, а также огромный накопленный практический опыт техники и технологии разработки трансляторов для различных языков программирования высокого уровня, архитектур процессоров и операционных систем.

Концепция синтаксически-ориентированной обработки предложений входного языка говорит о том, что выход транслятора формируется на основе структуры входного предложения, а её понимание помогает при осознании задач и алгоритмов каждого функционального блока современных трансляторов. В число основных идей, на которых базируется эта концепция, входят понятия и идеи формальных грамматик и формальных языков, а также идея семантических вычислений – связи «структуры со смыслом» для символьных цепочек, приводящая к пониманию процедуры трансляции как отображению символьных цепочек на их значения на основе структуры, приписанной этим цепочкам.

Цель: построение лексического анализатора компилятора.

Задача: написать программу для моделирования работы лексического анализатора.

План решения: построить конечный автомат для моделирования работы этапа лексического анализа компилятора при работе со структурами и объединениями языка С++, по конечному автомату построить соответствующую ему регулярную грамматику, построить КС-грамматику, смоделировать работу лексического анализатора в визуальном приложении Windows Forms в среде CLR, используя язык программирования C++.

# Описание предметной области

У каждого языка программирования, как и у любого естественного языка, есть свои синтаксис и семантика.

Синтаксис — совокупность правил, определяющих формирование его элементов. Синтаксис задается с помощью правил, которые описывают понятия некоторого языка. Примерами понятий являются: переменная, выражение, оператор, процедура. Последовательность понятий и их допустимое использование в правилах определяет синтаксически правильные структуры, образующие программы. Именно иерархия объектов, а не то, как они взаимодействуют между собой, определяются через синтаксис. Например, оператор может встречаться только в процедуре, а выражение в операторе, переменная может состоять из имени и необязательных индексов и т.д. Синтаксис не связан с такими явлениями в программе, как «несоответствие типов» или «переменная с данным именем не определена». Этим занимается семантика.

Семантика — правила и условия, определяющие соотношения между элементами языка и их смысловыми значениями, а также интерпретацию содержательного значения синтаксических конструкций языка. Объекты языка программирования не только размещаются в тексте в соответствии с некоторой иерархией, но и дополнительно связаны между собой посредством других понятий, образующих разнообразные ассоциации [4].

Семантика языков программирования изменяется в очень широких пределах. Цепочки, имеющие одинаковую синтаксическую структуру в различных языках программирования, могут различаться по семантике (что, например, наблюдается в C++, Pascal, Basic для приведенного выше фрагмента арифметического выражения). Язык C++ допускает описание переменных в любой точке программы перед первым ее использованием, а в Паскале переменные должны быть определены в специальной области описания. Знание семантики языка позволяет отделить ее от его синтаксиса и использовать для преобразования в другой язык (осуществить генерацию кода). Описание семантики и распознавание ее корректности обычно является самой трудоемкой и объемной частью транслятора, так как необходимо осуществить перебор и анализ множества вариантов допустимых комбинаций операций и операндов. В зависимости от принятого решения, транслятор может анализировать программу за один или несколько проходов, что влияет на скорость трансляции [4].

Транслятор — это программа, которая переводит входной текст в некоторый выход. Например, входом транслятора может быть программа, написанная на языке программирования высокого уровня, таком как Паскаль или С, или каком-нибудь специализированном языке. Выходом транслятора может быть целевая программа на языке ассемблера, промежуточном языке или машинном языке, либо просто последовательность некоторых действий, предписанных входным предложением (рис. 1).

В традиционных областях применения трансляторов принята своя терминология. Основной такой областью является трансляция входной программы, написанной на языке программирования высокого уровня в эквивалентную программу на другом языке. В этом случае входной язык называется исходным (source language), а входная программа на этом языке – исходным кодом (source code). Язык, на который осуществляется трансляция, называется объектным языком, результат трансляции – объектным кодом. Компьютер, накотором впоследствии оттранслированная программа будет запущена, называется целевой машиной (target machine), он не обязательно совпадает с типом машины, на которой работает транслятор [1].

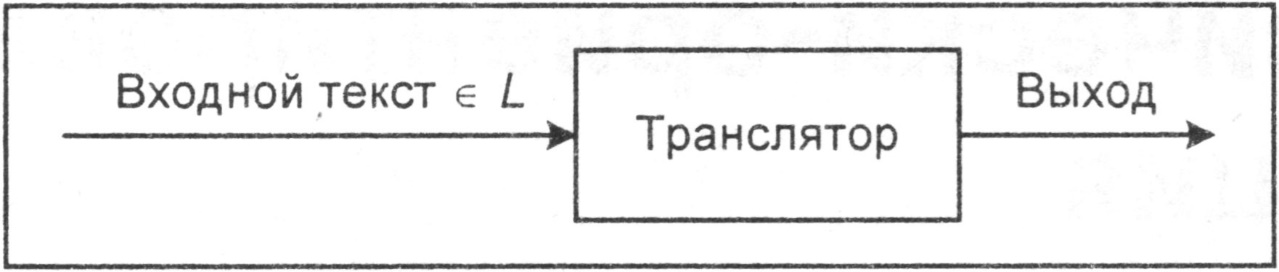


Рис. 1. Схема транслятора

При трансляции языков программирования от транслятора может требоваться выполнение разных задач. Если задачей является трансляция в язык низкого уровня, например, машинный язык или язык ассемблера, то такой транслятор называется компилятором. Другая возможная задача при трансляции программы на алгоритмическом языке – непосредственное выполнение транслятором операторов исходного кода. В этом случае транслятор называется интерпретатором.

В реальных трансляторах первой фазой является не синтаксический анализ входной программы, а так называемый лексический анализ, т.е. предварительная обработка входного текста с выделением в нем структурно значимых единиц – лексем (рис. 2). Лексема — это та минимальная единица языка, которая имеет смысл с точки зрения синтаксиса. В естественном языке такими единицами являются не буквы, а слова (словоформы), в языке программирования – никак не отдельные символы, но имена, отдельные служебные слова и т.д. [1].

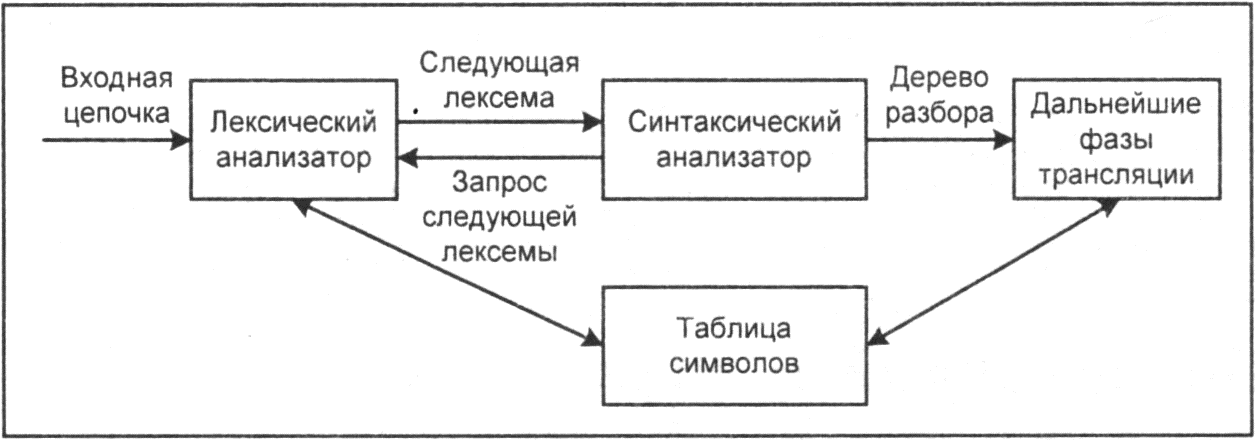


Рис. 2. Лексический и синтаксический анализаторы в структуре трансляции

Лексический анализатор должен выделить типы лексем и позаботиться об идентификации конкретных лексем, например, конкретных арифметических операций или идентификаторов. Обычным приёмом для сохранения информации об идентификаторах является построение в процессе лексического анализа таблицы имён, которая будет хранить имя переменной и другую необходимую информацию, а лексема id будет иметь индекс, указывающий на вход, соответствующий конкретному идентификатору в таблице имён.

Лексический анализатор может выполнять и другие функции, например, выбрасывать пробелы, комментарии, находить начало и конец символьных цепочек — констант и т.д. При этом все символьные последовательности, при распознавании которых лексический анализатор должен выполнить семантические операции, обычно задаются автоматной грамматикой или другим эквивалентным способом (например, регулярными выражениями) [1].

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 [3].

Синтаксический анализатор (синтаксический разбор) — это часть компилятора, которая отвечает за выявление основных синтаксических конструкций входного языка. В задачу синтаксического анализа входит: найти и выделить основные синтаксические конструкции в тексте входной программы, установить тип и проверить правильность каждой синтаксической конструкции, наконец, 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. Чаще всего регулярные грамматики применимы к языкам ассемблера, а языки высокого уровня построены на основе синтаксиса КС-языков.

Распознаватель дает ответ на вопрос о том, принадлежит или нет цепочка входных символов заданному языку – это основная задача синтаксического анализатора. Синтаксический анализатор воспринимает выход лексического анализатора и разбирает его в соответствии с грамматикой входного языка [3].

Конечным автоматом (КА) является простейший распознаватель, у которого отсутствует вспомогательная память. Он распознаёт автоматные (регулярные) языки, то есть с его помощью можно определить, принадлежит или нет заданная входная цепочка языку, допускаемому этим автоматом.

Детерминированным конечным автоматом (ДКА) называется пятерка объектов A = (X, S, s0, F, δ), где X – конечный алфавит входных символов, S – конечное множество состояний устройства управления, s0 ∊ S – начальное состояние устройства управления, F ⊆ S – множество выделенных заключительных состояний, δ – функция переходов, которая задается отображением δ: s × x → s', в котором множество δ(s, x) содержит не более одного состояния для любых s ∊ S и x ∊ X. Если δ(s, x) для всех s ∊ S и x ∊ X содержит одно состояние, то автомат A называется полностью определённым КА (рис. 3).

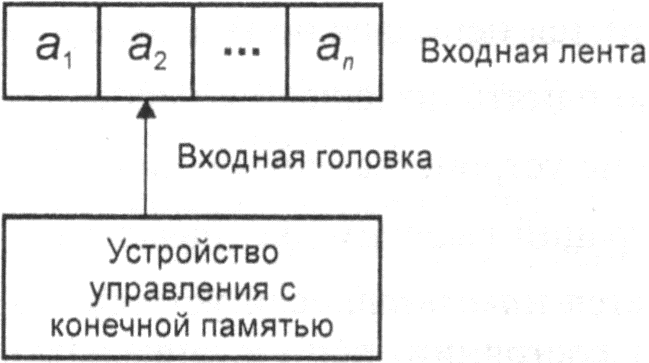


Рис. 3. Модель конечного автомата

Формальные грамматики — наиболее широко используемый математический аппарат для описания синтаксиса языков программирования. Это связано с простотой и естественностью описания языка и его структуры, с одной стороны, и наличием автоматических средств, позволяющих преобразовать грамматику в программу синтаксического анализа, с другой.

Контекстно-свободная грамматика (КС-грамматика, грамматика типа 2) — это грамматика G = (N, T, P, S), у которой правила вывода имеют вид А → В, где А ∊ N, а в V\*. Языки, порождаемые КС-грамматиками, называются контекстно-свободными языками (КС-языками) [2].

N – конечное непустое множество нетерминальных символов, которые представляют собой переменные, обозначающие множество строк и помогающие в определении языка, порождающего грамматикой; T – конечное непустое множество терминальных символов, которые представляют собой базовые символы, из которых строятся строки; P – множество продукций (правил), которые определяют способы, которыми терминалы и нетерминалы объединяются друг с другом для создания строк; S – стартовый символ грамматики (нетерминал).

Нетерминалу из левой части правила вывода КС-грамматики соответствует металингвистическая переменная из левой части металингвистической формулы. Отношению (→) – металингвистическая связка (::=), а цепочке терминалов и нетерминалов из правой части правила вывода – цепочка металингвистических переменных и основных символов языка из правой части металингвистической формулы.

Пример КС-грамматики выглядит следующим образом: G = ({E, Т, P}, {{i, +, \*, (, )}, P, E), где P = {E → E + T, E → T, T → T\* P, T → P, P → i, P → (E)}.

Грамматика типа 3 или автоматная (регулярная) грамматика — грамматика G = (N, T, P, S), у которой правила вывода имеют вид A → aВ или А → а, где А, N, а ∪ {ε}. Языки, порождаемые автоматными грамматиками, являются подмножеством КС-языков и называются автоматными (регулярными).

Примером автоматной грамматики является грамматика G = ({A, S}, {0, 1}, P, S), где P = {S → 0A, A → 1A, A → 0A, A → 0, A → 1} [2].

# Основная часть

## Постановка задачи

Для разработки программы был составлен тестовый код, содержащий конструкции объединения (union) и структуры (struct) языка C++ (рис. 4).



Рис. 4. Тестовая входная программа

Тестовый код содержит пустые строки, табуляции, многострочный и однострочные комментарии, задание структуры и объединения, структуры ввода данных с клавиатуры и вывода на консоль.

Программа должна:

* очищать входной код от пустых строк и табуляций/лишних пробелов;
* выделять из кода лексемы (ключевых слов, идентификаторов, констант, операторов сравнения, знаков операций и разделителей), заносить их в таблицу;
* формировать дескрипторный код программы;
* формировать псевдокод на основе дескрипторного;
* вести обнаружение и локализацию ошибки в процессе лексического анализа (недопустимый символ, неправильное задание константы и прочее).

## Построение конечного автомата

Определение, математическое, табличное, графическое

A = (X, S, s0, F, δ)

X = { a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z, A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ‘\_’, ‘,’, ‘.’, ‘^’, ‘:’, ‘;’, ‘"’, +, –, \*, /, =, %, &, ‘|’, !, ?, #, (, ), [, ], {, }, <, > }

S = { S0, S1, S2, S3, S4, S5, S6, S7, S8, S9, S10, S11, S12, S13, S14, S15, S16, S17, S18, S19, S20, S21, S22, S23, S24, S25, S26, S27, S28, S29, S30, S31, S32, S33, S34, S35, S36, S37, S38, S39, S40, S41, S42, S43, S44, S45, S46, S47, S48, S49, S50, S51, S52, S53, S54, S55, S56, S57, S58, S59, S60, S61, S62, S63, S64, S65, S66, S67, S68, S69, S70, S71, S72, S73, S74, S75, S76, S77, S78, S79, S80, S81, S82, S83, S84, S85, S86, S87, S88, S89, S90, S91, S92, S93, S94, S95, S96, S97, S98, S99, S100, S101, S102, S103, S104, S105, S106, S107, S108, S109, S110, S111, S112, S113, S114, S115, S116, S117, S118, S119, S120, S121, S122, S123, S124, S125, S126, S128, S129, S130, S131, S132, S135, S136, S137, S138, S139, S140, S141, S142, S143, S144, S145, S146, S147, S148, S149, S150, S151, S152, S153, S154, S155, S156, S157, S158, S159, S160, S161, S162, S163, S164, S165, S166, S167, S168, S169, S170, S171, S174, S175, S176, S177,S178,S179,S180,S181,S182,S183, S184 }

s0 = { S0 }

F = { S0, S2, S5, S6, S135, S137, S139, S140, S151, S152, S156 }

Граф конечного автомата представлен на рис. 5.

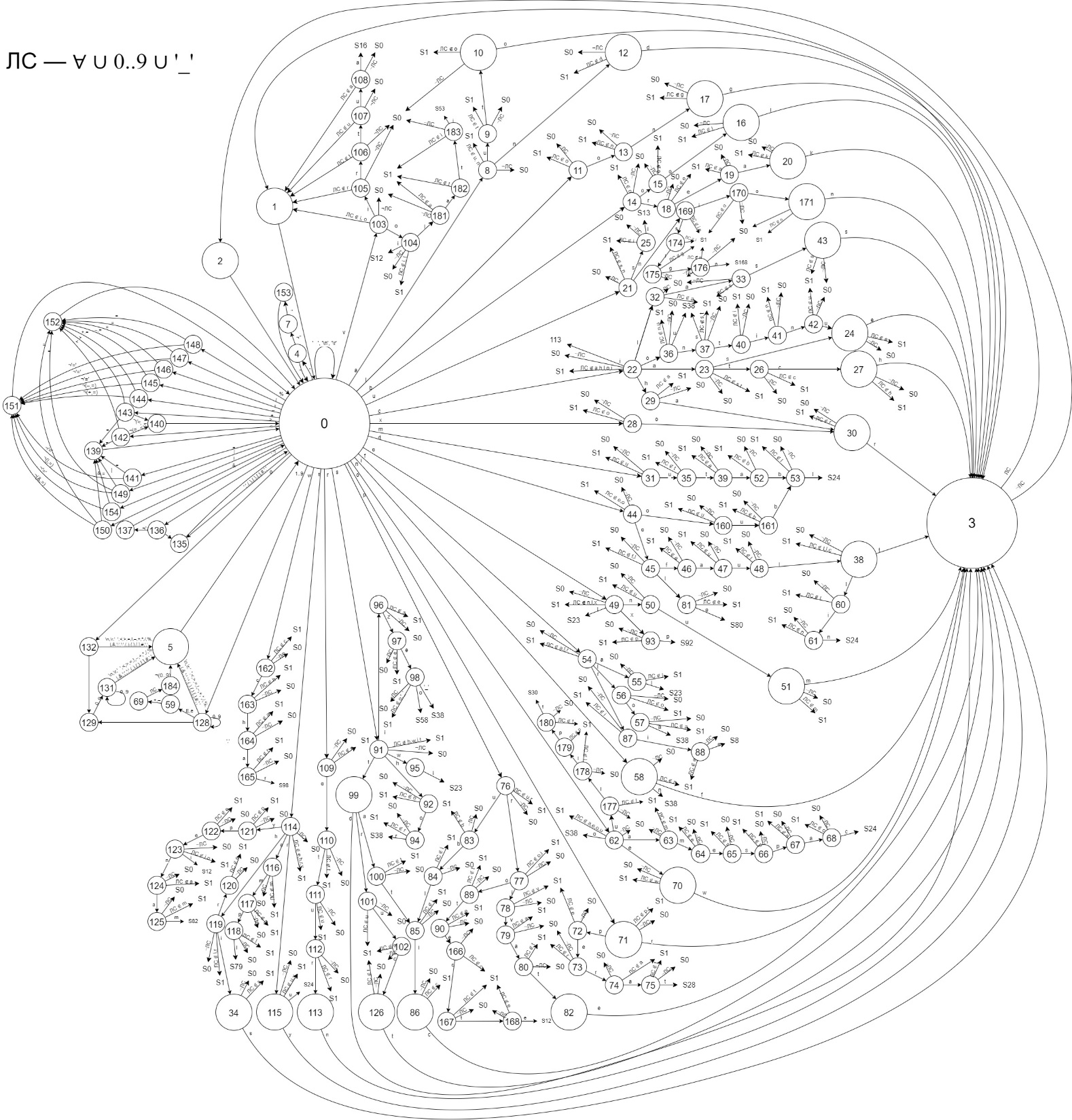


Рис. 5. Граф конечного автомата

Увеличенный вариант графа КА представлен на рис. П4.1.

Таблица переходов КА представлена в табл. 1.

Таблица 1

Таблица переходов по конечному автомату

|  |  |  |  |
| --- | --- | --- | --- |
| Переход по КА | Переход по КА | Переход по КА | Переход по КА |
| (S0, ' ') → S0 | (S34, ∀ ∪ 0..9 ∪ '\_' ∉ s) → S1 | (S78, v) → S79 | (S122, e) → S123 |
| (S0, \n) → S0 | (S34, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S78, ∀ ∪ 0..9 ∪ '\_' ∉ v) → S1 | (S122, ∀ ∪ 0..9 ∪ '\_' ∉ e) → S1 |
| (S0, \r) → S0 | (S35, t) → S39 | (S78, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S122, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 |
| (S0, " ) → S4 | (S35, ∀ ∪ 0..9 ∪ '\_' ∉ t) → S1 | (S79, a) → S80 | (S123, n) → S124 |
| (S0, a) → S8 | (S35, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S79, ∀ ∪ 0..9 ∪ '\_' ∉ a) → S1 | (S123, i) → S12 |
| (S0, l) → S11 | (S36, n) → S37 | (S79, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S123, ∀ ∪ 0..9 ∪ '\_' ∉ n, i) → S1 |
| (S0, b) → S14 | (S36, u) → S38 | (S80, t) → S82 | (S123, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 |
| (S0, u) → S21 | (S36, ∀ ∪ 0..9 ∪ '\_' ∉ n, u) → S1 | (S80, ∀ ∪ 0..9 ∪ '\_' ∉ t) → S1 | (S124, a) → S125 |
| (S0, c) → S22 | (S36, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S80, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S124, ∀ ∪ 0..9 ∪ '\_' ∉ a) → S1 |
| (S0, x) → S28 | (S37, s) → S38 | (S81, e) → S80 | (S124, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 |
| (S0, m) → S31 | (S37, t) → S40 | (S81, ∀ ∪ 0..9 ∪ '\_' ∉ e) → S1 | (S125, m) → S82 |
| (S0, d) → S44 | (S37, ∀ ∪ 0..9 ∪ '\_' ∉ s, t) → S1 | (S81, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S125, ∀ ∪ 0..9 ∪ '\_' ∉ m) → S1 |
| (S0, e) → S49 | (S37, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S82, e) → S3 | (S125, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 |
| (S0, f) → S54 | (S38, t) → S3 | (S82, ∀ ∪ 0..9 ∪ '\_' ∉ e) → S1 | (S126, t) → S3 |
| (S0, i) → S58 | (S38, l) → S60 | (S82, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S126, ∀ ∪ 0..9 ∪ '\_' ∉ t) → S1 |
| (S0, n) → S62 | (S38, c) → S157 | (S83, b) → S84 | (S126, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 |

Продолжение табл. 1

|  |  |  |  |
| --- | --- | --- | --- |
| (S0, o) → S71 | (S38, ∀ ∪ 0..9 ∪ '\_' ∉ t, l, c) → S1 | (S83, ∀ ∪ 0..9 ∪ '\_' ∉ b) → S1 | (S128, 0..9) → S128 |
| (S0, p) → S76 | (S38, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S83, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S128, '.') → S129 |
| (S0, s) → S91 | (S39, a) → S52 | (S84, l) → S85 | (S128, e, E) → S59 |
| (S0, v) → S103 | (S39, ∀ ∪ 0..9 ∪ '\_' ∉ a) → S1 | (S84, ∀ ∪ 0..9 ∪ '\_' ∉ l) → S1 | (S128, '\n', '\r', ' ', <, >, =, !, –, +, \*, /, %, |, &, ';', ':', (, ), {, }, [, ], #, ',') → S5 |
| (S0, r) → S109 | (S39, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S84, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S129, 0..9) → S131 |
| (S0, t) → S114 | (S40, i) → S41 | (S85, i) → S86 | (S131, '\n', '\r', ' ', <, >, =, !, –, +, \*, /, %, |, &, ';', ':', (, ), {, }, [, ], #, ',', '.') → S5 |
| (S0, w) → S162 | (S40, ∀ ∪ 0..9 ∪ '\_' ∉ i) → S1 | (S85, ∀ ∪ 0..9 ∪ '\_' ∉ i) → S1 | (S131, 0..9) → S131 |
| (S0, 1..9) → S128 | (S40, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S85, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S131, e, E) → S59 |
| (S0, 0) → S132 | (S41, n) → S42 | (S86, c) → S3 | (S132, '\n', '\r', ' ', <, >, =, !, –, +, \*, /, %, |, &, ';', ':', (, ), {, }, [, ], #, ',') → S5 |
| (S0, . ) → S135 | (S41, ∀ ∪ 0..9 ∪ '\_' ∉ n) → S1 | (S86, ∀ ∪ 0..9 ∪ '\_' ∉ c) → S1 | (S132, '.') → S129 |
| (S0, ',') → S135 | (S41, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S86, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S135 → S0 |
| (S0, {) → S135 | (S42, u) → S24 | (S87, i) → S88 | (S136, ':') → S135 |
| (S0, }) → S135 | (S42, ∀ ∪ 0..9 ∪ '\_' ∉ u) → S1 | (S87, ∀ ∪ 0..9 ∪ '\_' ∉ i) → S1 | (S136, ¬':') → S137 |
| (S0, [) → S135 | (S42, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S87, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S137 → S0 |
| (S0, ]) → S135 | (S43, s) → S3 | (S88, e) → S8 | (S138, '=') → S139 |
| (S0, '(') → S135 | (S43, ∀ ∪ 0..9 ∪ '\_' ∉ s) → S1 | (S88, ∀ ∪ 0..9 ∪ '\_' ∉ e) → S1 | (S138, ¬'=') → S140 |

Продолжение табл. 1

|  |  |  |  |
| --- | --- | --- | --- |
| (S0, ')') → S135 | (S43, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S88, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S139 → S0 |
| (S0, ';') → S135 | (S44, e) → S45 | (S89, t) → S90 | S140 → S0 |
| (S0, #) → S135 | (S44, o) → S160 | (S89, ∀ ∪ 0..9 ∪ '\_' ∉ t) → S1 | (S141, '=') → S139 |
| (S0, : ) → S136 | (S44, ∀ ∪ 0..9 ∪ '\_' ∉ e, o) → S1 | (S89, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S141, ¬'=') → S151 |
| (S0, ?) → S136 | (S44, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S90, e) → S166 | (S142, '=') → S139 |
| (S0, =) → S141 | (S45, f) → S46 | (S90, ∀ ∪ 0..9 ∪ '\_' ∉ e) → S1 | (S142, <) → S152 |
| (S0, <) → S142 | (S45, l) → S81 | (S90, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S142, ¬(=, <)) → S140 |
| (S0, >) → S143 | (S45, ∀ ∪ 0..9 ∪ '\_' ∉ f, l) → S1 | (S91, h) → S92 | (S143, '=') → S139 |
| (S0, +) → S144 | (S45, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S91, w) → S95 | (S143, >) → S152 |
| (S0, –) → S145 | (S46, a) → S47 | (S91, i) → S96 | (S143, ¬(=, >)) → S140 |
| (S0, \*) → S146 | (S46, ∀ ∪ 0..9 ∪ '\_' ∉ a) → S1 | (S91, t) → S99 | (S144, +, =) → S152 |
| (S0, /) → S147 | (S46, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S91, ∀ ∪ 0..9 ∪ '\_' ∉ h, w, I, t) → S1 | (S144, ¬(+, =)) → S151 |
| (S0, %) → S148 | (S47, u) → S48 | (S91, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S145, –, =) → S152 |
| (S0, |) → S149 | (S47, ∀ ∪ 0..9 ∪ '\_' ∉ u) → S1 | (S92, o) → S94 | (S145, ¬(–, =)) → S151 |
| (S0, &) → S150 | (S47, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S92, ∀ ∪ 0..9 ∪ '\_' ∉ o) → S1 | (S146, =) → S152 |
| (S0, !) → S154 | (S48, l) → S38 | (S92, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S146, ¬'=') → S151 |
| (S0, '\_' ∪ ∀ ∉ a, b, c, d, e, f, I, l, m, n, o, p, r, s, t, u, v, w, x) → S1 | (S48, ∀ ∪ 0..9 ∪ '\_' ∉ l) → S1 | (S93, p) → S92 | (S147, =) → S152 |

Продолжение табл. 1

|  |  |  |  |
| --- | --- | --- | --- |
| (S1, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S48, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S93, ∀ ∪ 0..9 ∪ '\_' ∉ p) → S1 | (S147, ¬'=') → S151 |
| (S1, ∀ ∪ 0..9 ∪ '\_') → S1 | (S49, l) → S23 | (S93, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S148, =) → S152 |
| S2 → S0 | (S49, n) → S50 | (S94, r) → S38 | (S148, ¬'=') → S151 |
| (S3, ¬(∀ ∪ 0..9 ∪ '\_')) → S2 | (S49, x) → S93 | (S94, ∀ ∪ 0..9 ∪ '\_' ∉ r) → S1 | (S149, '|') → S139 |
| (S3, ∀ ∪ 0..9 ∪ '\_') → S1 | (S49, ∀ ∪ 0..9 ∪ '\_' ∉ l, n, x) → S1 | (S94, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S149, =) → S152 |
| (S4, " ) → S0 | (S49, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S95, i) → S23 | (S149, ¬(|, =)) → S151 |
| (S4, ¬" ) → S7 | (S50, u) → S51 | (S95, ∀ ∪ 0..9 ∪ '\_' ∉ i) → S1 | (S150, &) → S139 |
| S5 → S0 | (S50, d) → S16 | (S95, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S150, =) → S152 |
| S6 → S0 | (S50, ∀ ∪ 0..9 ∪ '\_' ∉ u, d) → S1 | (S96, z) → S97 | (S149, ¬(&, =)) → S151 |
| (S7, " ) → S153 | (S50, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S96, ∀ ∪ 0..9 ∪ '\_' ∉ z) → S1 | S151 → S0 |
| (S8, u) → S9 | (S51, m) → S3 | (S96, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S152 → S0 |
| (S8, n) → S12 | (S51, ∀ ∪ 0..9 ∪ '\_' ∉ m) → S1 | (S97, e) → S98 | S153 → S0 |
| (S8, ∀ ∪ 0..9 ∪ '\_' ∉ u, n) → S1 | (S51, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S97, ∀ ∪ 0..9 ∪ '\_' ∉ e) → S1 | (S154, =) → S139 |
| (S8, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S52, b) → S53 | (S97, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S154, ¬'=') → S151 |
| (S9, t) → S10 | (S52, ∀ ∪ 0..9 ∪ '\_' ∉ b) → S1 | (S98, o) → S58 | (S155, '\n', '\r', ' ', <, >, =, !, –, +, \*, /, %, |, &, ';', ':', (, ), {, }, [, ], #, ',', '.') → S156 |
| (S9, ∀ ∪ 0..9 ∪ '\_' ∉ t) → S1 | (S52, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S98, '\_') → S38 | S156 → S0 |

Продолжение табл. 1

|  |  |  |  |
| --- | --- | --- | --- |
| (S9, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S53, l) → S24 | (S98, ∀ ∪ 0..9 ∪ '\_' ∉ o, '\_') → S1 | (S157, l) → S158 |
| (S10, o) → S3 | (S53, ∀ ∪ 0..9 ∪ '\_' ∉ l) → S1 | (S98, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S157, ∀ ∪ 0..9 ∪ '\_' ∉ l) → S1 |
| (S10, ∀ ∪ 0..9 ∪ '\_' ∉ o) → S1 | (S53, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S99, a) → S100 | (S157, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 |
| (S10, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S54, a) → S55 | (S99, r) → S101 | (S158, u) → S159 |
| (S11, o) → S13 | (S54, l) → S56 | (S99, d) → S3 | (S158, ∀ ∪ 0..9 ∪ '\_' ∉ u) → S1 |
| (S11, ∀ ∪ 0..9 ∪ '\_' ∉ o) → S1 | (S54, o) → S30 | (S99, ∀ ∪ 0..9 ∪ '\_' ∉ a, r, d) → S1 | (S158, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 |
| (S11, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S54, r) → S87 | (S99, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S159, d) → S24 |
| (S12, d) → S3 | (S54, ∀ ∪ 0..9 ∪ '\_' ∉ l, o, r) → S1 | (S100, t) → S85 | (S159, ∀ ∪ 0..9 ∪ '\_' ∉ d) → S1 |
| (S12, ∀ ∪ 0..9 ∪ '\_' ∉ d) → S1 | (S54, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S100, ∀ ∪ 0..9 ∪ '\_' ∉ t) → S1 | (S159, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 |
| (S12, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S55, l) → S23 | (S100, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S160, u) → S161 |
| (S13, n) → S17 | (S55, ∀ ∪ 0..9 ∪ '\_' ∉ l) → S1 | (S101, i) → S13 | (S160, ∀ ∪ 0..9 ∪ '\_' ∉ u) → S1 |
| (S13, ∀ ∪ 0..9 ∪ '\_' ∉ n) → S1 | (S55, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S101, u) → S102 | (S160, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 |
| (S13, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S56, o) → S57 | (S101, ∀ ∪ 0..9 ∪ '\_' ∉ i, u) → S1 | (S161, b) → S53 |
| (S14, o) → S15 | (S56, ∀ ∪ 0..9 ∪ '\_' ∉ o) → S1 | (S101, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S161, ∀ ∪ 0..9 ∪ '\_' ∉ b) → S1 |
| (S14, r) → S18 | (S56, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S102, c) → S126 | (S161, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 |
| (S14, ∀ ∪ 0..9 ∪ '\_' ∉ o, r) → S1 | (S57, a) → S38 | (S102, ∀ ∪ 0..9 ∪ '\_' ∉ c) → S1 | (S162, c) → S163 |

Продолжение табл. 1

|  |  |  |  |
| --- | --- | --- | --- |
| (S14, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S57, ∀ ∪ 0..9 ∪ '\_' ∉ a) → S1 | (S102, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S162, ∀ ∪ 0..9 ∪ '\_' ∉ c) → S1 |
| (S15, o) → S16 | (S57, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S103, i) → S105 | (S162, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 |
| (S15, ∀ ∪ 0..9 ∪ '\_' ∉ o) → S1 | (S58, n) → S38 | (S103, o) → S104 | (S163, h) → S164 |
| (S15, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S58, f) → S3 | (S103, ∀ ∪ 0..9 ∪ '\_' ∉ i, o) → S1 | (S163, ∀ ∪ 0..9 ∪ '\_' ∉ h) → S1 |
| (S16, l) → S3 | (S58, ∀ ∪ 0..9 ∪ '\_' ∉ n, f) → S1 | (S103, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S163, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 |
| (S16, ∀ ∪ 0..9 ∪ '\_' ∉ l) → S1 | (S58, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S104, i) → S12 | (S164, a) → S165 |
| (S16, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S59, + | –) → S69 | (S104, l) → S181 | (S164, ∀ ∪ 0..9 ∪ '\_' ∉ a) → S1 |
| (S17, g) → S3 | (S60, i) → S61 | (S104, ∀ ∪ 0..9 ∪ '\_' ∉ i, l) → S1 | (S164, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 |
| (S17, ∀ ∪ 0..9 ∪ '\_' ∉ g) → S1 | (S60, ∀ ∪ 0..9 ∪ '\_' ∉ i) → S1 | (S104, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S165, r) → S98 |
| (S17, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S60, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S105, r) → S106 | (S165, ∀ ∪ 0..9 ∪ '\_' ∉ r) → S1 |
| (S18, e) → S19 | (S61, n) → S24 | (S105, ∀ ∪ 0..9 ∪ '\_' ∉ r) → S1 | (S165, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 |
| (S18, ∀ ∪ 0..9 ∪ '\_' ∉ e) → S1 | (S61, ∀ ∪ 0..9 ∪ '\_' ∉ n) → S1 | (S105, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S166, c) → S167 |
| (S18, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S61, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S106, t) → S107 | (S166, ∀ ∪ 0..9 ∪ '\_' ∉ c) → S1 |
| (S19, a) → S20 | (S62, a) → S63 | (S106, ∀ ∪ 0..9 ∪ '\_' ∉ t) → S1 | (S166, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 |
| (S19, ∀ ∪ 0..9 ∪ '\_' ∉ a) → S1 | (S62, e) → S70 | (S106, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S167, t) → S168 |
| (S19, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S62, o) → S38 | (S107, u) → S108 | (S167, ∀ ∪ 0..9 ∪ '\_' ∉ t) → S1 |

Продолжение табл. 1

|  |  |  |  |
| --- | --- | --- | --- |
| (S20, k) → S3 | (S62, u) → S177 | (S107, ∀ ∪ 0..9 ∪ '\_' ∉ u) → S1 | (S167, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 |
| (S20, ∀ ∪ 0..9 ∪ '\_' ∉ k) → S1 | (S62, ∀ ∪ 0..9 ∪ '\_' ∉ a, e, o, u) → S1 | (S107, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S168, e) → S12 |
| (S20, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S62, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S108, a) → S16 | (S168, ∀ ∪ 0..9 ∪ '\_' ∉ e) → S1 |
| (S21, s) → S25 | (S63, m) → S64 | (S108, ∀ ∪ 0..9 ∪ '\_' ∉ a) → S1 | (S168, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 |
| (S21, n) → S169 | (S63, ∀ ∪ 0..9 ∪ '\_' ∉ m) → S1 | (S108, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S169, i) → S170 |
| (S21, ∀ ∪ 0..9 ∪ '\_' ∉ s, h) → S1 | (S63, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S109, e) → S110 | (S169, s) → S174 |
| (S21, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S64, e) → S65 | (S109, ∀ ∪ 0..9 ∪ '\_' ∉ e) → S1 | (S169, ∀ ∪ 0..9 ∪ '\_' ∉ i, s) → S1 |
| (S22, a) → S23 | (S64, ∀ ∪ 0..9 ∪ '\_' ∉ e) → S1 | (S109, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S169, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 |
| (S22, h) → S29 | (S64, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S110, t) → S111 | (S170, o) → S171 |
| (S22, l) → S32 | (S65, s) → S66 | (S110, ∀ ∪ 0..9 ∪ '\_' ∉ t) → S1 | (S170, ∀ ∪ 0..9 ∪ '\_' ∉ o) → S1 |
| (S22, o) → S36 | (S65, ∀ ∪ 0..9 ∪ '\_' ∉ s) → S1 | (S110, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S170, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 |
| (S22, i) → S113 | (S65, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S111, u) → S112 | (S171, n) → S3 |
| (S22, ∀ ∪ 0..9 ∪ '\_' ∉ a, h, l, o, i) → S1 | (S66, p) → S67 | (S111, ∀ ∪ 0..9 ∪ '\_' ∉ u) → S1 | (S171, ∀ ∪ 0..9 ∪ '\_' ∉ n) → S1 |
| (S22, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S66, ∀ ∪ 0..9 ∪ '\_' ∉ p) → S1 | (S111, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S171, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 |
| (S23, s) → S24 | (S66, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S112, r) → S113 | (S174, i) → S175 |
| (S23, t) → S26 | (S67, a) → S68 | (S112, ∀ ∪ 0..9 ∪ '\_' ∉ r) → S1 | (S174, ∀ ∪ 0..9 ∪ '\_' ∉ i) → S1 |
| (S23, ∀ ∪ 0..9 ∪ '\_' ∉ s, t) → S1 | (S67, ∀ ∪ 0..9 ∪ '\_' ∉ a) → S1 | (S112, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S174, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 |

Продолжение табл. 1

|  |  |  |  |
| --- | --- | --- | --- |
| (S23, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S67, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S113, n) → S3 | (S175, g) → S176 |
| (S24, e) → S3 | (S68, c) → S24 | (S113, ∀ ∪ 0..9 ∪ '\_' ∉ n) → S1 | (S175, ∀ ∪ 0..9 ∪ '\_' ∉ g) → S1 |
| (S24, ∀ ∪ 0..9 ∪ '\_' ∉ e) → S1 | (S68, ∀ ∪ 0..9 ∪ '\_' ∉ c) → S1 | (S113, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S175, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 |
| (S24, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S68, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S114, e) → S116 | (S176, n) → S168 |
| (S25, i) → S13 | (S69, ∀ ∪ 0..9 ∪ '\_') → S184 | (S114, h) → S119 | (S176, ∀ ∪ 0..9 ∪ '\_' ∉ n) → S1 |
| (S25, ∀ ∪ 0..9 ∪ '\_' ∉ i) → S1 | (S70, w) → S3 | (S114, r) → S115 | (S176, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 |
| (S25, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S70, ∀ ∪ 0..9 ∪ '\_' ∉ w) → S1 | (S114, y) → S121 | (S177, l) → S178 |
| (S26, c) → S27 | (S70, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S114, ∀ ∪ 0..9 ∪ '\_' ∉ e, h, r, y) → S1 | (S177, ∀ ∪ 0..9 ∪ '\_' ∉ l) → S1 |
| (S26, ∀ ∪ 0..9 ∪ '\_' ∉ c) → S1 | (S71, p) → S72 | (S114, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S177, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 |
| (S26, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S71, r) → S3 | (S115, u) → S24 | (S178, l) → S179 |
| (S27, h) → S3 | (S71, ∀ ∪ 0..9 ∪ '\_' ∉ p, r) → S1 | (S115, y) → S3 | (S178, ∀ ∪ 0..9 ∪ '\_' ∉ l) → S1 |
| (S27, ∀ ∪ 0..9 ∪ '\_' ∉ h) → S1 | (S71, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S115, ∀ ∪ 0..9 ∪ '\_' ∉ u, y) → S1 | (S178, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 |
| (S27, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S72, e) → S73 | (S115, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S179, p) → S180 |
| (S28, o) → S30 | (S72, ∀ ∪ 0..9 ∪ '\_' ∉ e) → S1 | (S116, m) → S117 | (S179, ∀ ∪ 0..9 ∪ '\_' ∉ p) → S1 |
| (S28, ∀ ∪ 0..9 ∪ '\_' ∉ o) → S1 | (S72, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S116, ∀ ∪ 0..9 ∪ '\_' ∉ m) → S1 | (S179, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 |
| (S28, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S73, r) → S74 | (S116, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S180, t) → S30 |

Продолжение табл. 1

|  |  |  |  |
| --- | --- | --- | --- |
| (S29, a) → S30 | (S73, ∀ ∪ 0..9 ∪ '\_' ∉ r) → S1 | (S117, p) → S118 | (S180, ∀ ∪ 0..9 ∪ '\_' ∉ t) → S1 |
| (S29, ∀ ∪ 0..9 ∪ '\_' ∉ a) → S1 | (S73, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S117, ∀ ∪ 0..9 ∪ '\_' ∉ p) → S1 | (S180, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 |
| (S29, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S74, a) → S75 | (S117, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S181, a) → S182 |
| (S30, r) → S3 | (S74, ∀ ∪ 0..9 ∪ '\_' ∉ a) → S1 | (S118, l) → S79 | (S181, ∀ ∪ 0..9 ∪ '\_' ∉ a) → S1 |
| (S30, ∀ ∪ 0..9 ∪ '\_' ∉ r) → S1 | (S74, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S118, ∀ ∪ 0..9 ∪ '\_' ∉ l) → S1 | (S181, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 |
| (S30, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S75, t) → S28 | (S118, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S182, t) → S183 |
| (S31, u) → S35 | (S75, ∀ ∪ 0..9 ∪ '\_' ∉ t) → S1 | (S119, i) → S34 | (S182, ∀ ∪ 0..9 ∪ '\_' ∉ t) → S1 |
| (S31, ∀ ∪ 0..9 ∪ '\_' ∉ u) → S1 | (S75, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S119, r) → S120 | (S182, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 |
| (S31, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S76, r) → S77 | (S119, ∀ ∪ 0..9 ∪ '\_' ∉ i, r) → S1 | (S183, i) → S53 |
| (S32, a) → S33 | (S76, u) → S83 | (S119, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S183, ∀ ∪ 0..9 ∪ '\_' ∉ i) → S1 |
| (S32, ∀ ∪ 0..9 ∪ '\_' ∉ a) → S1 | (S76, ∀ ∪ 0..9 ∪ '\_' ∉ r, u) → S1 | (S120, o) → S170 | (S183, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 |
| (S32, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S76, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S120, ∀ ∪ 0..9 ∪ '\_' ∉ o) → S1 | (S184, ¬(0..9)) → S5 |
| (S33, s) → S43 | (S77, i) → S78 | (S120, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S184, '\n', '\r', ' ', <, >, =, !, –, +, \*, /, %, |, &, ';', ':', (, ), {, }, [, ], #, ',', '.') → S130 |
| (S33, ∀ ∪ 0..9 ∪ '\_' ∉ s) → S1 | (S77, o) → S89 | (S121, p) → S122 |  |
| (S33, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S77, ∀ ∪ 0..9 ∪ '\_' ∉ i, o) → S1 | (S121, ∀ ∪ 0..9 ∪ '\_' ∉ p) → S1 |  |
| (S34, s) → S3 | (S77, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | (S121, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 |  |

## Построение регулярной грамматики по конечному автомата

G = (T, N, P, S)

T = X = { a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z, A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ‘\_’, ‘,’, ‘.’, ‘^’, ‘:’, ‘;’, ‘"’, +, –, \*, /, =, %, &, ‘|’, !, ?, #, (, ), [, ], {, }, <, > }

N = { S0, S1, S2, S3, S4, S5, S6, S7, S8, S9, S10, S11, S12, S13, S14, S15, S16, S17, S18, S19, S20, S21, S22, S23, S24, S25, S26, S27, S28, S29, S30, S31, S32, S33, S34, S35, S36, S37, S38, S39, S40, S41, S42, S43, S44, S45, S46, S47, S48, S49, S50, S51, S52, S53, S54, S55, S56, S57, S58, S59, S60, S61, S62, S63, S64, S65, S66, S67, S68, S69, S70, S71, S72, S73, S74, S75, S76, S77, S78, S79, S80, S81, S82, S83, S84, S85, S86, S87, S88, S89, S90, S91, S92, S93, S94, S95, S96, S97, S98, S99, S100, S101, S102, S103, S104, S105, S106, S107, S108, S109, S110, S111, S112, S113, S114, S115, S116, S117, S118, S119, S120, S121, S122, S123, S124, S125, S126, S128, S129, S130, S131, S132, S135, S136, S137, S138, S139, S140, S141, S142, S143, S144, S145, S146, S147, S148, S149, S150, S151, S152, S153, S154, S155, S156, S157, S158, S159, S160, S161, S162, S163, S164, S165, S166, S167, S168, S169, S170, S171, S174, S175, S176, S177,S178,S179,S180,S181,S182,S183, S184 }

S = S0

Совмещённая таблица переходов по конечному автомату и регулярной грамматике представлена в табл. 2.

Таблица 2

Совмещённая таблица переходов

|  |  |
| --- | --- |
| Переход по КА | Переход по регулярной грамматике |
| 1 | 2 |
| (S0, ' ') → S0 | S0 → ' ', S0 S0 → ' ' |
| (S0, \n) → S0 | S0 → \n, S0 S0 → \n |
| (S0, \r) → S0 | S0 → \r, S0 S0 → \r |
| (S0, " ) → S4 | S0 → ", S4 |
| (S0, a) → S8 | S0 → a, S8 |

Продолжение табл. 2

|  |  |
| --- | --- |
| 1 | 2 |
| (S0, l) → S11 | S0 → l, S11 |
| (S0, b) → S14 | S0 → b, S14 |
| (S0, u) → S21 | S0 → u, S21 |
| (S0, c) → S22 | S0 → c, S22 |
| (S0, x) → S28 | S0 → x, S28 |
| (S0, m) → S31 | S0 → m, S31 |
| (S0, d) → S44 | S0 → d, S44 |
| (S0, e) → S49 | S0 → e, S49 |
| (S0, f) → S54 | S0 → f, S54 |
| (S0, i) → S58 | S0 → i, S58 |
| (S0, n) → S62 | S0 → n, S62 |
| (S0, o) → S71 | S0 → o, S71 |
| (S0, p) → S76 | S0 → p, S76 |
| (S0, s) → S91 | S0 → s, S91 |
| (S0, v) → S103 | S0 → v, S103 |
| (S0, r) → S109 | S0 → r, S109 |
| (S0, t) → S114 | S0 → t, S114 |
| (S0, w) → S162 | S0 → w, S162 |
| (S0, 1..9) → S128 | S0 → 1..9, S128 |
| (S0, 0) → S132 | S0 → 0, S132 |
| (S0, . ) → S135 | S0 → '.', S135 S0 → '.' |
| (S0, ',') → S135 | S0 → ',', S135 S0 → ',' |
| (S0, {) → S135 | S0 → {, S135 S0 → { |
| (S0, }) → S135 | S0 → {, S135 S0 → { |
| (S0, [) → S135 | S0 → [, S135 S0 → [ |
| (S0, ]) → S135 | S0 → ], S135 S0 → ] |
| (S0, '(') → S135 | S0 → (, S135 S0 → ( |

Продолжение табл. 2

|  |  |
| --- | --- |
| 1 | 2 |
| (S0, ')') → S135 | S0 → ), S135 S0 → ) |
| (S0, ';') → S135 | S0 → ';', S135 S0 → ';' |
| (S0, #) → S135 | S0 → #, S135 S0 → # |
| (S0, : ) → S136 | S0 → ':', S136 |
| (S0, ?) → S136 | S0 → ?, S136 |
| (S0, =) → S141 | S0 → =, S141 |
| (S0, <) → S142 | S0 → <, S142 |
| (S0, >) → S143 | S0 → <, S143 |
| (S0, +) → S144 | S0 → +, S144 |
| (S0, –) → S145 | S0 → –, S145 |
| (S0, \*) → S146 | S0 → \*, S146 |
| (S0, /) → S147 | S0 → /, S147 |
| (S0, %) → S148 | S0 → %, S148 |
| (S0, |) → S149 | S0 → |, S149 |
| (S0, &) → S150 | S0 → &, S150 |
| (S0, !) → S154 | S0 → !, S154 |
| (S0, '\_' ∪ ∀ ∉ a, b, c, d, e, f, I, l, m, n, o, p, r, s, t, u, v, w, x) → S1 | S0 → '\_' ∪ ∀ ∉ (a, b, c, d, e, f, I, l, m, n, o, p, r, s, t, u, v, w, x), S1 |
| (S1, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S1→¬(∀ ∪ 0..9 ∪ '\_'), S0 S1→¬(∀ ∪ 0..9 ∪ '\_') |
| (S1, ∀ ∪ 0..9 ∪ '\_') → S1 | S1 → ∀ ∪ 0..9 ∪ '\_', S1 |
| S2 → S0 | S2 → S0 |
| (S3, ¬(∀ ∪ 0..9 ∪ '\_')) → S2 | S3→¬(∀ ∪ 0..9 ∪ '\_'), S2 S3→¬(∀ ∪ 0..9 ∪ '\_') |
| (S3, ∀ ∪ 0..9 ∪ '\_') → S1 | S3 → ∀ ∪ 0..9 ∪ '\_', S1 |
| (S4, " ) → S0 | S4 → ", S0 S4 → " |
| (S4, ¬" ) → S7 | S4 → ¬", S7 |
| S5 → S0 | S5 → S0 |
| S6 → S0 | S6 → S0 |

Продолжение табл. 2

|  |  |
| --- | --- |
| 1 | 2 |
| (S7, " ) → S153 | S7 → ", S153 |
| (S8, u) → S9 | S8 → u, S9 |
| (S8, n) → S12 | S8 → n, S12 |
| (S8, ∀ ∪ 0..9 ∪ '\_' ∉ u, n) → S1 | S8 → ∀ ∪ 0..9 ∪ '\_' ∉ (u, n), S1 |
| (S8, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S8→¬(∀ ∪ 0..9 ∪ '\_'), S0 S8→¬(∀ ∪ 0..9 ∪ '\_') |
| (S9, t) → S10 | S9 → t, S10 |
| (S9, ∀ ∪ 0..9 ∪ '\_' ∉ t) → S1 | S9 → ∀ ∪ 0..9 ∪ '\_' ∉ t, S1 |
| (S9, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S9→¬(∀ ∪ 0..9 ∪ '\_'), S0 S9→¬(∀ ∪ 0..9 ∪ '\_') |
| (S10, o) → S3 | S10 → o, S3 |
| (S10, ∀ ∪ 0..9 ∪ '\_' ∉ o) → S1 | S10 → ∀ ∪ 0..9 ∪ '\_' ∉ o, S1 |
| (S10, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S10 → ¬(∀ ∪ 0..9∪'\_'), S0 S10→¬(∀∪0..9∪ '\_') |
| (S11, o) → S13 | S11 → o, S13 |
| (S11, ∀ ∪ 0..9 ∪ '\_' ∉ o) → S1 | S11 → ∀ ∪ 0..9 ∪ '\_' ∉ o, S1 |
| (S11, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S11→¬(∀∪0..9∪'\_'), S0 S11→¬(∀ ∪ 0..9 ∪ '\_') |
| (S12, d) → S3 | S12 → d, S3 |
| (S12, ∀ ∪ 0..9 ∪ '\_' ∉ d) → S1 | S12 → ∀ ∪ 0..9 ∪ '\_' ∉ d, S1 |
| (S12, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S12→¬(∀∪0..9∪'\_'), S0 S12→¬(∀ ∪ 0..9 ∪ '\_') |
| (S13, n) → S17 | S13 → n, S17 |
| (S13, ∀ ∪ 0..9 ∪ '\_' ∉ n) → S1 | S13→∀∪0..9∪'\_' ∉ n, S1 S13→∀∪0..9∪'\_'∉n |
| (S13, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S13→¬(∀ ∪ 0..9 ∪ '\_'), S0 S13→¬(∀∪0..9∪'\_') |
| (S14, o) → S15 | S14 → o, S15 |
| (S14, r) → S18 | S14 → r, S18 |
| (S14, ∀ ∪ 0..9 ∪ '\_' ∉ o, r) → S1 | S14 → ∀ ∪ 0..9 ∪ '\_' ∉ (o, r), S1 |
| (S14, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S14→¬(∀ ∪ 0..9 ∪ '\_'), S0 S14→¬(∀∪0..9∪ '\_') |
| (S15, o) → S16 | S15 → o, S16 |
| (S15, ∀ ∪ 0..9 ∪ '\_' ∉ o) → S1 | S15 → ∀ ∪ 0..9 ∪ '\_' ∉ o, S1 |
| (S15, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S15→¬(∀ ∪ 0..9∪'\_'), S0 S15→¬(∀∪0..9∪ '\_') |

Продолжение табл. 2

|  |  |
| --- | --- |
| 1 | 2 |
| (S16, l) → S3 | S16 → l, S3 |
| (S16, ∀ ∪ 0..9 ∪ '\_' ∉ l) → S1 | S16 → ∀ ∪ 0..9 ∪ '\_' ∉ l, S1 |
| (S16, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S16→¬(∀∪ 0..9 ∪ '\_'), S0 S16→¬(∀∪0..9∪'\_') |
| (S17, g) → S3 | S17 → g, S3 |
| (S17, ∀ ∪ 0..9 ∪ '\_' ∉ g) → S1 | S17 → ∀ ∪ 0..9 ∪ '\_' ∉ g, S1 |
| (S17, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S17→¬(∀∪0..9∪'\_'), S0 S17 →¬(∀ ∪ 0..9 ∪ '\_') |
| (S18, e) → S19 | S18 → e, S19 |
| (S18, ∀ ∪ 0..9 ∪ '\_' ∉ e) → S1 | S18 → ∀ ∪ 0..9 ∪ '\_' ∉ e, S1 |
| (S18, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S18→¬(∀∪0..9∪'\_'), S0 S18→ ¬(∀ ∪ 0..9 ∪ '\_') |
| (S19, a) → S20 | S19 → a, S20 |
| (S19, ∀ ∪ 0..9 ∪ '\_' ∉ a) → S1 | S19 → ∀ ∪ 0..9 ∪ '\_' ∉ a, S1 |
| (S19, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S19→¬(∀∪0..9∪'\_'), S0 S19→¬(∀∪0..9∪'\_') |
| (S20, k) → S3 | S20 → k, S3 |
| (S20, ∀ ∪ 0..9 ∪ '\_' ∉ k) → S1 | S20 → ∀ ∪ 0..9 ∪ '\_' ∉ k, S1 |
| (S20, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S20→¬(∀∪0..9∪'\_'), S0 S20→¬(∀∪0..9∪'\_') |
| (S21, s) → S25 | S21 → s, S25 |
| (S21, n) → S169 | S21 → n, S169 |
| (S21, ∀ ∪ 0..9 ∪ '\_' ∉ s, h) → S1 | S21 → ∀ ∪ 0..9 ∪ '\_' ∉ (s, h), S1 |
| (S21, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S21→¬(∀∪0..9∪'\_'), S0 S21→¬(∀∪0..9∪'\_') |
| (S22, a) → S23 | S22 → a, S23 |
| (S22, h) → S29 | S22 → h, S29 |
| (S22, l) → S32 | S22 → l, S32 |
| (S22, o) → S36 | S22 → o, S36 |
| (S22, i) → S113 | S22 → i, S113 |
| (S22,∀∪0..9∪'\_'∉a,h,l,o,i)→S1 | S22 → ∀ ∪ 0..9 ∪ '\_' ∉ (a, h, l, o, i), S1 |
| (S22, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S22→¬(∀∪0..9∪'\_'), S0 S22→¬(∀∪0..9∪'\_') |

Продолжение табл. 2

|  |  |
| --- | --- |
| 1 | 2 |
| (S23, s) → S24 | S23 → s, S24 |
| (S23, t) → S26 | S23 → t, S26 |
| (S23, ∀ ∪ 0..9 ∪ '\_' ∉ s, t) → S1 | S23 → ∀ ∪ 0..9 ∪ '\_' ∉ (s, t), S1 |
| (S23, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S23→¬(∀∪0..9∪'\_'), S0 S23→¬(∀∪0..9∪'\_') |
| (S24, e) → S3 | S24 → e, S3 |
| (S24, ∀ ∪ 0..9 ∪ '\_' ∉ e) → S1 | S24 → ∀ ∪ 0..9 ∪ '\_' ∉ e, S1 |
| (S24, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S24→¬(∀∪0..9∪'\_'), S0 S24→¬(∀∪0..9∪'\_') |
| (S25, i) → S13 | S25 → i, S13 |
| (S25, ∀ ∪ 0..9 ∪ '\_' ∉ i) → S1 | S25 → ∀ ∪ 0..9 ∪ '\_' ∉ i, S1 |
| (S25, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S25→¬(∀∪0..9∪'\_'), S0 S25→¬(∀∪0..9∪'\_') |
| (S26, c) → S27 | S26 → c, S27 |
| (S26, ∀ ∪ 0..9 ∪ '\_' ∉ c) → S1 | S26 → ∀ ∪ 0..9 ∪ '\_' ∉ c, S1 |
| (S26, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S26→¬(∀∪0..9∪'\_'), S0 S26→¬(∀∪0..9∪'\_') |
| (S27, h) → S3 | S27 → h, S3 |
| (S27, ∀ ∪ 0..9 ∪ '\_' ∉ h) → S1 | S27 → ∀ ∪ 0..9 ∪ '\_' ∉ h, S1 |
| (S27, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S27→¬(∀∪0..9∪'\_'), S0 S27→¬(∀∪0..9∪'\_') |
| (S28, o) → S30 | S28 → o, S30 |
| (S28, ∀ ∪ 0..9 ∪ '\_' ∉ o) → S1 | S28 → ∀ ∪ 0..9 ∪ '\_' ∉ o, S1 |
| (S28, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S28→¬(∀∪0..9∪'\_'), S0 S28→¬(∀∪0..9∪'\_') |
| (S29, a) → S30 | S29 → a, S30 |
| (S29, ∀ ∪ 0..9 ∪ '\_' ∉ a) → S1 | S29 → ∀ ∪ 0..9 ∪ '\_' ∉ a, S1 |
| (S29, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S29→¬(∀∪0..9∪'\_'), S0 S29→¬(∀∪0..9∪'\_') |
| (S30, r) → S3 | S30 → r, S3 |
| (S30, ∀ ∪ 0..9 ∪ '\_' ∉ r) → S1 | S30 → ∀ ∪ 0..9 ∪ '\_' ∉ r, S1 |
| (S30, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S30→¬(∀∪0..9∪'\_'), S0 S30→¬(∀∪0..9∪'\_') |
| (S31, u) → S35 | S31 → u, S35 |

Продолжение табл. 2

|  |  |
| --- | --- |
| 1 | 2 |
| (S31, ∀ ∪ 0..9 ∪ '\_' ∉ u) → S1 | S31 → ∀ ∪ 0..9 ∪ '\_' ∉ u, S1 |
| (S31, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S31→¬(∀∪0..9∪'\_'), S0 S31→¬(∀∪0..9∪'\_') |
| (S32, a) → S33 | S32 → a, S33 |
| (S32, ∀ ∪ 0..9 ∪ '\_' ∉ a) → S1 | S32 → ∀ ∪ 0..9 ∪ '\_' ∉ a, S1 |
| (S32, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S32→¬(∀∪0..9∪'\_'), S0 S32→¬(∀∪0..9∪'\_') |
| (S33, s) → S43 | S33 → s, S43 |
| (S33, ∀ ∪ 0..9 ∪ '\_' ∉ s) → S1 | S33 → ∀ ∪ 0..9 ∪ '\_' ∉ s, S1 |
| (S33, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S33→¬(∀∪0..9∪'\_'), S0 S33→¬(∀∪0..9∪'\_') |
| (S34, s) → S3 | S34 → s, S3 |
| (S34, ∀ ∪ 0..9 ∪ '\_' ∉ s) → S1 | S34 → ∀ ∪ 0..9 ∪ '\_' ∉ s, S1 |
| (S34, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S34→¬(∀∪0..9∪'\_'), S0 S34→¬(∀∪0..9∪'\_') |
| (S35, t) → S39 | S35 → t, S39 |
| (S35, ∀ ∪ 0..9 ∪ '\_' ∉ t) → S1 | S35 → ∀ ∪ 0..9 ∪ '\_' ∉ t, S1 |
| (S35, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S35→¬(∀∪0..9∪'\_'), S0 S35→¬(∀∪0..9∪'\_') |
| (S36, n) → S37 | S36 → n, S37 |
| (S36, u) → S38 | S36 → u, S38 |
| (S36, ∀ ∪ 0..9 ∪ '\_' ∉ n, u) → S1 | S36 → ∀ ∪ 0..9 ∪ '\_' ∉ (n, u), S1 |
| (S36, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S37→¬(∀∪0..9∪'\_'), S0 S37→¬(∀∪0..9∪'\_') |
| (S37, s) → S38 | S37 → s, S38 |
| (S37, t) → S40 | S37 → t, S40 |
| (S37, ∀ ∪ 0..9 ∪ '\_' ∉ s, t) → S1 | S37 → ∀ ∪ 0..9 ∪ '\_' ∉ (s, t), S1 |
| (S37, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S37→¬(∀∪0..9∪'\_'), S0 S37→¬(∀∪0..9∪'\_') |
| (S38, t) → S3 | S38 → t, S3 |
| (S38, l) → S60 | S38 → l, S60 |
| (S38, c) → S157 | S38 → c, S157 |
| (S38, ∀ ∪ 0..9 ∪ '\_' ∉ t, l, c) → S1 | S38 → ∀ ∪ 0..9 ∪ '\_' ∉ (t, l, c), S1 |

Продолжение табл. 2

|  |  |
| --- | --- |
| 1 | 2 |
| (S38, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S38→¬(∀∪0..9∪'\_'), S0 S38→¬(∀∪0..9∪'\_') |
| (S39, a) → S52 | S39 → a, S52 |
| (S39, ∀ ∪ 0..9 ∪ '\_' ∉ a) → S1 | S39 → ∀ ∪ 0..9 ∪ '\_' ∉ a, S1 |
| (S39, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S39→¬(∀∪0..9∪'\_'), S0 S39→¬(∀∪0..9∪'\_') |
| (S40, i) → S41 | S40 → i, S41 |
| (S40, ∀ ∪ 0..9 ∪ '\_' ∉ i) → S1 | S40 → ∀ ∪ 0..9 ∪ '\_' ∉ i, S1 |
| (S40, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S40→¬(∀∪0..9∪'\_'), S0 S40→¬(∀∪0..9∪'\_') |
| (S41, n) → S42 | S41 → n, S42 |
| (S41, ∀ ∪ 0..9 ∪ '\_' ∉ n) → S1 | S41 → ∀ ∪ 0..9 ∪ '\_' ∉ n, S1 |
| (S41, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S41→¬(∀∪0..9∪'\_'), S0 S41→¬(∀∪0..9∪'\_') |
| (S42, u) → S24 | S42 → u, S24 |
| (S42, ∀ ∪ 0..9 ∪ '\_' ∉ u) → S1 | S42 → ∀ ∪ 0..9 ∪ '\_' ∉ u, S1 |
| (S42, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S42→¬(∀∪0..9∪'\_'), S0 S42→¬(∀∪0..9∪'\_') |
| (S43, s) → S3 | S43 → s, S3 |
| (S43, ∀ ∪ 0..9 ∪ '\_' ∉ s) → S1 | S43 → ∀ ∪ 0..9 ∪ '\_' ∉ s, S1 |
| (S43, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S43→¬(∀∪0..9∪'\_'), S0 S43→¬(∀∪0..9∪'\_') |
| (S44, e) → S45 | S44 → e, S45 |
| (S44, o) → S160 | S44 → o, S160 |
| (S44, ∀ ∪ 0..9 ∪ '\_' ∉ e, o) → S1 | S44 → ∀ ∪ 0..9 ∪ '\_' ∉ (e, o), S1 |
| (S44, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S44→¬(∀∪0..9∪'\_'), S0 S44→¬(∀∪0..9∪'\_') |
| (S45, f) → S46 | S45 → f, S46 |
| (S45, l) → S81 | S45 → l, S81 |
| (S45, ∀ ∪ 0..9 ∪ '\_' ∉ f, l) → S1 | S45 → ∀ ∪ 0..9 ∪ '\_' ∉ (f, l), S1 |
| (S45, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S45→¬(∀∪0..9∪'\_'), S0 S45→¬(∀∪0..9∪'\_') |
| (S46, a) → S47 | S46 → a, S47 |
| (S46, ∀ ∪ 0..9 ∪ '\_' ∉ a) → S1 | S46 → ∀ ∪ 0..9 ∪ '\_' ∉ a, S1 |

Продолжение табл. 2

|  |  |
| --- | --- |
| 1 | 2 |
| (S46, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S46→¬(∀∪0..9∪'\_'), S0 S46→¬(∀∪0..9∪'\_') |
| (S47, u) → S48 | S47 → u, S48 |
| (S47, ∀ ∪ 0..9 ∪ '\_' ∉ u) → S1 | S47 → ∀ ∪ 0..9 ∪ '\_' ∉ u, S1 |
| (S47, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S47→¬(∀∪0..9∪'\_'), S0 S47→¬(∀∪0..9∪'\_') |
| (S48, l) → S38 | S48 → l, S38 |
| (S48, ∀ ∪ 0..9 ∪ '\_' ∉ l) → S1 | S48 → ∀ ∪ 0..9 ∪ '\_' ∉ l, S1 |
| (S48, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S48→¬(∀∪0..9∪'\_'), S0 S48→¬(∀∪0..9∪'\_') |
| (S49, l) → S23 | S49 → l, S23 |
| (S49, n) → S50 | S49 → n, S50 |
| (S49, x) → S93 | S49 → x, S93 |
| (S49, ∀ ∪ 0..9 ∪ '\_' ∉ l, n, x) → S1 | S49 → ∀ ∪ 0..9 ∪ '\_' ∉ (l, n, x), S1 |
| (S49, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S49→¬(∀∪0..9∪'\_'), S0 S49→¬(∀∪0..9∪'\_') |
| (S50, u) → S51 | S50 → u, S51 |
| (S50, d) → S16 | S50 → d, S16 |
| (S50, ∀ ∪ 0..9 ∪ '\_' ∉ u, d) → S1 | S50 → ∀ ∪ 0..9 ∪ '\_' ∉ (u, d), S1 |
| (S50, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S50→¬(∀∪0..9∪'\_'), S0 S50→¬(∀∪0..9∪'\_') |
| (S51, m) → S3 | S51 → m, S3 |
| (S51, ∀ ∪ 0..9 ∪ '\_' ∉ m) → S1 | S51 → ∀ ∪ 0..9 ∪ '\_' ∉ m, S1 |
| (S51, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S51→¬(∀∪0..9∪'\_'), S0 S51→¬(∀∪0..9∪'\_') |
| (S52, b) → S53 | S52 → b, S53 |
| (S52, ∀ ∪ 0..9 ∪ '\_' ∉ b) → S1 | S52 → ∀ ∪ 0..9 ∪ '\_' ∉ b, S1 |
| (S52, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S52→¬(∀∪0..9∪'\_'), S0 S52→¬(∀∪0..9∪'\_') |
| (S53, l) → S24 | S53 → l, S24 |
| (S53, ∀ ∪ 0..9 ∪ '\_' ∉ l) → S1 | S53 → ∀ ∪ 0..9 ∪ '\_' ∉ l, S1 |
| (S53, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S53→¬(∀∪0..9∪'\_'), S0 S53→¬(∀∪0..9∪'\_') |
| (S54, a) → S55 | S54 → a, S55 |

Продолжение табл. 2

|  |  |
| --- | --- |
| 1 | 2 |
| (S54, l) → S56 | S54 → l, S56 |
| (S54, o) → S30 | S54 → o, S30 |
| (S54, r) → S87 | S54 → r, S87 |
| (S54, ∀ ∪ 0..9 ∪ '\_' ∉ l, o, r) → S1 | S54 → ∀ ∪ 0..9 ∪ '\_' ∉ (l, o, r), S1 |
| (S54, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S54→¬(∀∪0..9∪'\_'), S0 S54→¬(∀∪0..9∪'\_') |
| (S55, l) → S23 | S55 → l, S23 |
| (S55, ∀ ∪ 0..9 ∪ '\_' ∉ l) → S1 | S55 → ∀ ∪ 0..9 ∪ '\_' ∉ l, S1 |
| (S55, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S55→¬(∀∪0..9∪'\_'), S0 S55→¬(∀∪0..9∪'\_') |
| (S56, o) → S57 | S56 → o, S57 |
| (S56, ∀ ∪ 0..9 ∪ '\_' ∉ o) → S1 | S56 → ∀ ∪ 0..9 ∪ '\_' ∉ o, S1 |
| (S56, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S56→¬(∀∪0..9∪'\_'), S0 S56→¬(∀∪0..9∪'\_') |
| (S57, a) → S38 | S57 → a, S38 |
| (S57, ∀ ∪ 0..9 ∪ '\_' ∉ a) → S1 | S57 → ∀ ∪ 0..9 ∪ '\_' ∉ a, S1 |
| (S57, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S57→¬(∀∪0..9∪'\_'), S0 S57→¬(∀∪0..9∪'\_') |
| (S58, n) → S38 | S58 → n, S38 |
| (S58, f) → S3 | S58 → f, S3 |
| (S58, ∀ ∪ 0..9 ∪ '\_' ∉ n, f) → S1 | S58 → ∀ ∪ 0..9 ∪ '\_' ∉ (n, f), S1 |
| (S58, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S58→¬(∀∪0..9∪'\_'), S0 S58→¬(∀∪0..9∪'\_') |
| (S59, + | –) → S69 | S59 → + | –, S69 |
| (S60, i) → S61 | S60 → i, S61 |
| (S60, ∀ ∪ 0..9 ∪ '\_' ∉ i) → S1 | S60 → ∀ ∪ 0..9 ∪ '\_' ∉ i, S1 |
| (S60, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S60→¬(∀∪0..9∪'\_'), S0 S60→¬(∀∪0..9∪'\_') |
| (S61, n) → S24 | S61 → n, S24 |
| (S61, ∀ ∪ 0..9 ∪ '\_' ∉ n) → S1 | S61 → ∀ ∪ 0..9 ∪ '\_' ∉ n, S1 |
| (S61, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S61→¬(∀∪0..9∪'\_'), S0 S61→¬(∀∪0..9∪'\_') |
| (S62, a) → S63 | S62 → a, S63 |

Продолжение табл. 2

|  |  |
| --- | --- |
| 1 | 2 |
| (S62, e) → S70 | S62 → e, S70 |
| (S62, o) → S38 | S62 → o, S38 |
| (S62, u) → S177 | S62 → u, S177 |
| (S62, ∀∪0..9∪'\_'∉a, e, o, u) → S1 | S62 → ∀ ∪ 0..9 ∪ '\_' ∉ (a, e, o, u), S1 |
| (S62, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S62→¬(∀∪0..9∪'\_'), S0 S62→¬(∀∪0..9∪'\_') |
| (S63, m) → S64 | S63 → m, S64 |
| (S63, ∀ ∪ 0..9 ∪ '\_' ∉ m) → S1 | S63 → ∀ ∪ 0..9 ∪ '\_' ∉ m, S1 |
| (S63, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S63→¬(∀∪0..9∪'\_'), S0 S63→¬(∀∪0..9∪'\_') |
| (S64, e) → S65 | S64 → e, S65 |
| (S64, ∀ ∪ 0..9 ∪ '\_' ∉ e) → S1 | S64 → ∀ ∪ 0..9 ∪ '\_' ∉ e, S1 |
| (S64, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S64→¬(∀∪0..9∪'\_'), S0 S64→¬(∀∪0..9∪'\_') |
| (S65, s) → S66 | S65 → s, S66 |
| (S65, ∀ ∪ 0..9 ∪ '\_' ∉ s) → S1 | S65 → ∀ ∪ 0..9 ∪ '\_' ∉ s, S1 |
| (S65, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S65→¬(∀∪0..9∪'\_'), S0 S65→¬(∀∪0..9∪'\_') |
| (S66, p) → S67 | S66 → p, S67 |
| (S66, ∀ ∪ 0..9 ∪ '\_' ∉ p) → S1 | S66 → ∀ ∪ 0..9 ∪ '\_' ∉ p, S1 |
| (S66, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S66→¬(∀∪0..9∪'\_'), S0 S66→¬(∀∪0..9∪'\_') |
| (S67, a) → S68 | S67 → a, S68 |
| (S67, ∀ ∪ 0..9 ∪ '\_' ∉ a) → S1 | S67 → ∀ ∪ 0..9 ∪ '\_' ∉ a, S1 |
| (S67, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S67→¬(∀∪0..9∪'\_'), S0 S67→¬(∀∪0..9∪'\_') |
| (S68, c) → S24 | S68 → c, S24 |
| (S68, ∀ ∪ 0..9 ∪ '\_' ∉ c) → S1 | S68 → ∀ ∪ 0..9 ∪ '\_' ∉ c, S1 |
| (S68, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S68→¬(∀∪0..9∪'\_'), S0 S68→¬(∀∪0..9∪'\_') |
| (S69, ∀ ∪ 0..9 ∪ '\_') → S184 | S69 → ∀ ∪ 0..9 ∪ '\_', S184 |
| (S70, w) → S3 | S70 → w, S3 |
| (S70, ∀ ∪ 0..9 ∪ '\_' ∉ w) → S1 | S70 → ∀ ∪ 0..9 ∪ '\_' ∉ w, S1 |

Продолжение табл. 2

|  |  |
| --- | --- |
| 1 | 2 |
| (S70, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S70→¬(∀∪0..9∪'\_'), S0 S70→¬(∀∪0..9∪'\_') |
| (S71, p) → S72 | S71 → p, S72 |
| (S71, r) → S3 | S71 → r, S3 |
| (S71, ∀ ∪ 0..9 ∪ '\_' ∉ p, r) → S1 | S71 → ∀ ∪ 0..9 ∪ '\_' ∉ (p, r), S1 |
| (S71, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S71→¬(∀∪0..9∪'\_'), S0 S71→¬(∀∪0..9∪'\_') |
| (S72, e) → S73 | S72 → e, S73 |
| (S72, ∀ ∪ 0..9 ∪ '\_' ∉ e) → S1 | S72 → ∀ ∪ 0..9 ∪ '\_' ∉ e, S1 |
| (S72, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S72→¬(∀∪0..9∪'\_'), S0 S72→¬(∀∪0..9∪'\_') |
| (S73, r) → S74 | S73 → r, S74 |
| (S73, ∀ ∪ 0..9 ∪ '\_' ∉ r) → S1 | S73 → ∀ ∪ 0..9 ∪ '\_' ∉ r, S1 |
| (S73, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S73→¬(∀∪0..9∪'\_'), S0 S73→¬(∀∪0..9∪'\_') |
| (S74, a) → S75 | S74 → a, S75 |
| (S74, ∀ ∪ 0..9 ∪ '\_' ∉ a) → S1 | S74 → ∀ ∪ 0..9 ∪ '\_' ∉ a, S1 |
| (S74, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S74→¬(∀∪0..9∪'\_'), S0 S74→¬(∀∪0..9∪'\_') |
| (S75, t) → S28 | S75 → t, S28 |
| (S75, ∀ ∪ 0..9 ∪ '\_' ∉ t) → S1 | S75 → ∀ ∪ 0..9 ∪ '\_' ∉ t, S1 |
| (S75, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S75→¬(∀∪0..9∪'\_'), S0 S75→¬(∀∪0..9∪'\_') |
| (S76, r) → S77 | S76 → r, S77 |
| (S76, u) → S83 | S76 → u, S83 |
| (S76, ∀ ∪ 0..9 ∪ '\_' ∉ r, u) → S1 | S76 → ∀ ∪ 0..9 ∪ '\_' ∉ (r, u), S1 |
| (S76, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S76→¬(∀∪0..9∪'\_'), S0 S76→¬(∀∪0..9∪'\_') |
| (S77, i) → S78 | S77 → i, S78 |
| (S77, o) → S89 | S77 → o, S89 |
| (S77, ∀ ∪ 0..9 ∪ '\_' ∉ i, o) → S1 | S77 → ∀ ∪ 0..9 ∪ '\_' ∉ (i, o), S1 |
| (S77, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S77→¬(∀∪0..9∪'\_'), S0 S77→¬(∀∪0..9∪'\_') |
| (S78, v) → S79 | S78 → v, S79 |
| (S78, ∀ ∪ 0..9 ∪ '\_' ∉ v) → S1 | S78 → ∀ ∪ 0..9 ∪ '\_' ∉ v, S1 |
| (S78, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S78→¬(∀∪0..9∪'\_'), S0 S78→¬(∀∪0..9∪'\_') |
| (S79, a) → S80 | S79 → a, S80 |
| (S79, ∀ ∪ 0..9 ∪ '\_' ∉ a) → S1 | S79 → ∀ ∪ 0..9 ∪ '\_' ∉ a, S1 |
| (S79, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S79→¬(∀∪0..9∪'\_'), S0 S79→¬(∀∪0..9∪'\_') |

Продолжение табл. 2

|  |  |
| --- | --- |
| 1 | 2 |
| (S80, t) → S82 | S80 → t, S82 |
| (S80, ∀ ∪ 0..9 ∪ '\_' ∉ t) → S1 | S80 → ∀ ∪ 0..9 ∪ '\_' ∉ t, S1 |
| (S80, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S80→¬(∀∪0..9∪'\_'), S0 S80→¬(∀∪0..9∪'\_') |
| (S81, e) → S80 | S81 → e, S80 |
| (S81, ∀ ∪ 0..9 ∪ '\_' ∉ e) → S1 | S81 → ∀ ∪ 0..9 ∪ '\_' ∉ e, S1 |
| (S81, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S81→¬(∀∪0..9∪'\_'), S0 S81→¬(∀∪0..9∪'\_') |
| (S82, e) → S3 | S82 → e, S3 |
| (S82, ∀ ∪ 0..9 ∪ '\_' ∉ e) → S1 | S82 → ∀ ∪ 0..9 ∪ '\_' ∉ e, S1 |
| (S82, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S82→¬(∀∪0..9∪'\_'), S0 S82→¬(∀∪0..9∪'\_') |
| (S83, b) → S84 | S83 → b, S84 |
| (S83, ∀ ∪ 0..9 ∪ '\_' ∉ b) → S1 | S83 → ∀ ∪ 0..9 ∪ '\_' ∉ b, S1 |
| (S83, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S83→¬(∀∪0..9∪'\_'), S0 S83→¬(∀∪0..9∪'\_') |
| (S84, l) → S85 | S84 → l, S85 |
| (S84, ∀ ∪ 0..9 ∪ '\_' ∉ l) → S1 | S84 → ∀ ∪ 0..9 ∪ '\_' ∉ l, S1 |
| (S84, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S84→¬(∀∪0..9∪'\_'), S0 S84→¬(∀∪0..9∪'\_') |
| (S85, i) → S86 | S85 → i, S86 |
| (S85, ∀ ∪ 0..9 ∪ '\_' ∉ i) → S1 | S85 → ∀ ∪ 0..9 ∪ '\_' ∉ i, S1 |
| (S85, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S85→¬(∀∪0..9∪'\_'), S0 S85→¬(∀∪0..9∪'\_') |
| (S86, c) → S3 | S86 → c, S3 |
| (S86, ∀ ∪ 0..9 ∪ '\_' ∉ c) → S1 | S86 → ∀ ∪ 0..9 ∪ '\_' ∉ c, S1 |
| (S86, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S86→¬(∀∪0..9∪'\_'), S0 S86→¬(∀∪0..9∪'\_') |
| (S87, i) → S88 | S87 → i, S88 |
| (S87, ∀ ∪ 0..9 ∪ '\_' ∉ i) → S1 | S87 → ∀ ∪ 0..9 ∪ '\_' ∉ i, S1 |
| (S87, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S87→¬(∀∪0..9∪'\_'), S0 S87→¬(∀∪0..9∪'\_') |
| (S88, e) → S8 | S88 → e, S8 |
| (S88, ∀ ∪ 0..9 ∪ '\_' ∉ e) → S1 | S88 → ∀ ∪ 0..9 ∪ '\_' ∉ e, S1 |
| (S88, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S88→¬(∀∪0..9∪'\_'), S0 S88→¬(∀∪0..9∪'\_') |
| (S89, t) → S90 | S89 → t, S90 |
| (S89, ∀ ∪ 0..9 ∪ '\_' ∉ t) → S1 | S89 → ∀ ∪ 0..9 ∪ '\_' ∉ t, S1 |
| (S89, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S89→¬(∀∪0..9∪'\_'), S0 S89→¬(∀∪0..9∪'\_') |
| (S90, e) → S166 | S90 → e, S166 |

Продолжение табл. 2

|  |  |
| --- | --- |
| 1 | 2 |
| (S90, ∀ ∪ 0..9 ∪ '\_' ∉ e) → S1 | S90 → ∀ ∪ 0..9 ∪ '\_' ∉ e, S1 |
| (S90, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S90→¬(∀∪0..9∪'\_'), S0 S90→¬(∀∪0..9∪'\_') |
| (S91, h) → S92 | S91 → h, S92 |
| (S91, w) → S95 | S91 → w, S95 |
| (S91, i) → S96 | S91 → i, S96 |
| (S91, t) → S99 | S91 → t, S99 |
| (S91, ∀ ∪ 0..9 ∪ '\_' ∉ h, w, I, t) → S1 | S91 → ∀ ∪ 0..9 ∪ '\_' ∉ (h, w, I, t), S1 |
| (S91, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S91→¬(∀∪0..9∪'\_'), S0 S91→¬(∀∪0..9∪'\_') |
| (S92, o) → S94 | S92 → o, S94 |
| (S92, ∀ ∪ 0..9 ∪ '\_' ∉ o) → S1 | S92 → ∀ ∪ 0..9 ∪ '\_' ∉ o, S1 |
| (S92, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S92→¬(∀∪0..9∪'\_'), S0 S92→¬(∀∪0..9∪'\_') |
| (S93, p) → S92 | S93 → p, S92 |
| (S93, ∀ ∪ 0..9 ∪ '\_' ∉ p) → S1 | S93 → ∀ ∪ 0..9 ∪ '\_' ∉ p, S1 |
| (S93, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S93→¬(∀∪0..9∪'\_'), S0 S93→¬(∀∪0..9∪'\_') |
| (S94, r) → S38 | S94 → r, S38 |
| (S94, ∀ ∪ 0..9 ∪ '\_' ∉ r) → S1 | S94 → ∀ ∪ 0..9 ∪ '\_' ∉ r, S1 |
| (S94, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S94→¬(∀∪0..9∪'\_'), S0 S94→¬(∀∪0..9∪'\_') |
| (S95, i) → S23 | S95 → i, S23 |
| (S95, ∀ ∪ 0..9 ∪ '\_' ∉ i) → S1 | S95 → ∀ ∪ 0..9 ∪ '\_' ∉ i, S1 |
| (S95, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S95→¬(∀∪0..9∪'\_'), S0 S95→¬(∀∪0..9∪'\_') |
| (S96, z) → S97 | S96 → z, S97 |
| (S96, ∀ ∪ 0..9 ∪ '\_' ∉ z) → S1 | S96 → ∀ ∪ 0..9 ∪ '\_' ∉ z, S1 |
| (S96, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S96→¬(∀∪0..9∪'\_'), S0 S96→¬(∀∪0..9∪'\_') |
| (S97, e) → S98 | S97 → e, S98 |
| (S97, ∀ ∪ 0..9 ∪ '\_' ∉ e) → S1 | S97 → ∀ ∪ 0..9 ∪ '\_' ∉ e, S1 |
| (S97, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S97→¬(∀∪0..9∪'\_'), S0 S97→¬(∀∪0..9∪'\_') |
| (S98, o) → S58 | S98 → o, S58 |
| (S98, '\_') → S38 | S98 → '\_', S38 |
| (S98, ∀ ∪ 0..9 ∪ '\_' ∉ o, '\_') → S1 | S98 → ∀ ∪ 0..9 ∪ '\_' ∉ (o, '\_'), S1 |

Продолжение табл. 2

|  |  |
| --- | --- |
| 1 | 2 |
| (S98, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S98→¬(∀∪0..9∪'\_'), S0 S98→¬(∀∪0..9∪'\_') |
| (S99, a) → S100 | S99 → a, S100 |
| (S99, r) → S101 | S99 → r, S101 |
| (S99, d) → S3 | S99 → d, S3 |
| (S99, ∀ ∪ 0..9 ∪ '\_' ∉ a, r, d) → S1 | S99 → ∀ ∪ 0..9 ∪ '\_' ∉ (a, r, d), S1 |
| (S99, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S99→¬(∀∪0..9∪'\_'), S0 S99→¬(∀∪0..9∪'\_') |
| (S100, t) → S85 | S100 → t, S85 |
| (S100, ∀ ∪ 0..9 ∪ '\_' ∉ t) → S1 | S100 → ∀ ∪ 0..9 ∪ '\_' ∉ t, S1 |
| (S100, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S100→¬(∀∪0..9∪'\_'), S0 S100→¬(∀∪0..9∪'\_') |
| (S101, i) → S13 | S101 → i, S13 |
| (S101, u) → S102 | S101 → u, S102 |
| (S101, ∀ ∪ 0..9 ∪ '\_' ∉ i, u) → S1 | S101 → ∀ ∪ 0..9 ∪ '\_' ∉ (i, u), S1 |
| (S101, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S101→¬(∀∪0..9∪'\_'), S0 S101→¬(∀∪0..9∪'\_') |
| (S102, c) → S126 | S102 → c, S126 |
| (S102, ∀ ∪ 0..9 ∪ '\_' ∉ c) → S1 | S102 → ∀ ∪ 0..9 ∪ '\_' ∉ c, S1 |
| (S102, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S102→¬(∀∪0..9∪'\_'), S0 S102→¬(∀∪0..9∪'\_') |
| (S103, i) → S105 | S103 → i, S105 |
| (S103, o) → S104 | S103 → o, S104 |
| (S103, ∀ ∪ 0..9 ∪ '\_' ∉ i, o) → S1 | S103 → ∀ ∪ 0..9 ∪ '\_' ∉ (i, o), S1 |
| (S103, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S103→¬(∀∪0..9∪'\_'), S0 S103→¬(∀∪0..9∪'\_') |
| (S104, i) → S12 | S104 → i, S12 |
| (S104, l) → S181 | S104 → l, S181 |
| (S104, ∀ ∪ 0..9 ∪ '\_' ∉ i, l) → S1 | S104 → ∀ ∪ 0..9 ∪ '\_' ∉ (i, l), S1 |
| (S104, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S104→¬(∀∪0..9∪'\_'), S0 S104→¬(∀∪0..9∪'\_') |
| (S105, r) → S106 | S105 → r, S106 |
| (S105, ∀ ∪ 0..9 ∪ '\_' ∉ r) → S1 | S105 → ∀ ∪ 0..9 ∪ '\_' ∉ r, S1 |
| (S105, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S105→¬(∀∪0..9∪'\_'), S0 S105→¬(∀∪0..9∪'\_') |
| (S106, t) → S107 | S106 → t, S107 |
| (S106, ∀ ∪ 0..9 ∪ '\_' ∉ t) → S1 | S106 → ∀ ∪ 0..9 ∪ '\_' ∉ t, S1 |
| (S106, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S106→¬(∀∪0..9∪'\_'), S0 S106→¬(∀∪0..9∪'\_') |
| (S107, u) → S108 | S107 → u, S108 |

Продолжение табл. 2

|  |  |
| --- | --- |
| 1 | 2 |
| (S107, ∀ ∪ 0..9 ∪ '\_' ∉ u) → S1 | S107 → ∀ ∪ 0..9 ∪ '\_' ∉ u, S1 |
| (S107, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S107→¬(∀∪0..9∪'\_'), S0 S107→¬(∀∪0..9∪'\_') |
| (S108, a) → S16 | S108 → a, S16 |
| (S108, ∀ ∪ 0..9 ∪ '\_' ∉ a) → S1 | S108 → ∀ ∪ 0..9 ∪ '\_' ∉ a, S1 |
| (S108, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S108→¬(∀∪0..9∪'\_'), S0 S108→¬(∀∪0..9∪'\_') |
| (S109, e) → S110 | S109 → e, S110 |
| (S109, ∀ ∪ 0..9 ∪ '\_' ∉ e) → S1 | S109 → ∀ ∪ 0..9 ∪ '\_' ∉ e, S1 |
| (S109, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S109→¬(∀∪0..9∪'\_'), S0 S109→¬(∀∪0..9∪'\_') |
| (S110, t) → S111 | S110 → t, S111 |
| (S110, ∀ ∪ 0..9 ∪ '\_' ∉ t) → S1 | S110 → ∀ ∪ 0..9 ∪ '\_' ∉ t, S1 |
| (S110, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S110→¬(∀∪0..9∪'\_'), S0 S110→¬(∀∪0..9∪'\_') |
| (S111, u) → S112 | S111 → u, S112 |
| (S111, ∀ ∪ 0..9 ∪ '\_' ∉ u) → S1 | S111 → ∀ ∪ 0..9 ∪ '\_' ∉ u, S1 |
| (S111, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S111→¬(∀∪0..9∪'\_'), S0 S111→¬(∀∪0..9∪'\_') |
| (S112, r) → S113 | S112 → r, S113 |
| (S112, ∀ ∪ 0..9 ∪ '\_' ∉ r) → S1 | S112 → ∀ ∪ 0..9 ∪ '\_' ∉ r, S1 |
| (S112, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S112→¬(∀∪0..9∪'\_'), S0 S112→¬(∀∪0..9∪'\_') |
| (S113, n) → S3 | S113 → n, S3 |
| (S113, ∀ ∪ 0..9 ∪ '\_' ∉ n) → S1 | S113 → ∀ ∪ 0..9 ∪ '\_' ∉ n, S1 |
| (S113, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S113→¬(∀∪0..9∪'\_'), S0 S113→¬(∀∪0..9∪'\_') |
| (S114, e) → S116 | S114 → e, S116 |
| (S114, h) → S119 | S114 → h, S119 |
| (S114, r) → S115 | S114 → r, S115 |
| (S114, y) → S121 | S114 → y, S121 |
| (S114, ∀ ∪ 0..9 ∪ '\_' ∉ e, h, r, y) → S1 | S114 → ∀ ∪ 0..9 ∪ '\_' ∉ (e, h, r, y), S1 |
| (S114, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S114→¬(∀∪0..9∪'\_'), S0 S114→¬(∀∪0..9∪'\_') |
| (S115, u) → S24 | S115 → u, S24 |
| (S115, y) → S3 | S115 → y, S3 |
| (S115, ∀ ∪ 0..9 ∪ '\_' ∉ u, y) → S1 | S115 → ∀ ∪ 0..9 ∪ '\_' ∉ (u, y), S1 |
| (S115, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S115→¬(∀∪0..9∪'\_'), S0 S115→¬(∀∪0..9∪'\_') |
| (S116, m) → S117 | S116 → m, S117 |

Продолжение табл. 2

|  |  |
| --- | --- |
| 1 | 2 |
| (S116, ∀ ∪ 0..9 ∪ '\_' ∉ m) → S1 | S116 → ∀ ∪ 0..9 ∪ '\_' ∉ m, S1 |
| (S116, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S116→¬(∀∪0..9∪'\_'), S0 S116→¬(∀∪0..9∪'\_') |
| (S117, p) → S118 | S117 → p, S118 |
| (S117, ∀ ∪ 0..9 ∪ '\_' ∉ p) → S1 | S117 → ∀ ∪ 0..9 ∪ '\_' ∉ p, S1 |
| (S117, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S117→¬(∀∪0..9∪'\_'), S0 S117→¬(∀∪0..9∪'\_') |
| (S118, l) → S79 | S118 → l, S79 |
| (S118, ∀ ∪ 0..9 ∪ '\_' ∉ l) → S1 | S118 → ∀ ∪ 0..9 ∪ '\_' ∉ l, S1 |
| (S118, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S118→¬(∀∪0..9∪'\_'), S0 S118→¬(∀∪0..9∪'\_') |
| (S119, i) → S34 | S119 → i, S34 |
| (S119, r) → S120 | S119 → r, S120 |
| (S119, ∀ ∪ 0..9 ∪ '\_' ∉ i, r) → S1 | S119 → ∀ ∪ 0..9 ∪ '\_' ∉ (i, r), S1 |
| (S119, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S119→¬(∀∪0..9∪'\_'), S0 S119→¬(∀∪0..9∪'\_') |
| (S120, o) → S170 | S120 → o, S170 |
| (S120, ∀ ∪ 0..9 ∪ '\_' ∉ o) → S1 | S120 → ∀ ∪ 0..9 ∪ '\_' ∉ o, S1 |
| (S120, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S120→¬(∀∪0..9∪'\_'), S0 S120→¬(∀∪0..9∪'\_') |
| (S121, p) → S122 | S121 → p, S122 |
| (S121, ∀ ∪ 0..9 ∪ '\_' ∉ p) → S1 | S121 → ∀ ∪ 0..9 ∪ '\_' ∉ p, S1 |
| (S121, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S121→¬(∀∪0..9∪'\_'), S0 S121→¬(∀∪0..9∪'\_') |
| (S122, e) → S123 | S122 → e, S123 |
| (S122, ∀ ∪ 0..9 ∪ '\_' ∉ e) → S1 | S122 → ∀ ∪ 0..9 ∪ '\_' ∉ e, S1 |
| (S122, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S122→¬(∀∪0..9∪'\_'), S0 S122→¬(∀∪0..9∪'\_') |
| (S123, n) → S124 | S123 → n, S124 |
| (S123, i) → S12 | S123 → i, S12 |
| (S123, ∀ ∪ 0..9 ∪ '\_' ∉ n, i) → S1 | S123 → ∀ ∪ 0..9 ∪ '\_' ∉ (n, i), S1 |
| (S123, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S123→¬(∀∪0..9∪'\_'), S0 S123→¬(∀∪0..9∪'\_') |
| (S124, a) → S125 | S124 → a, S125 |
| (S124, ∀ ∪ 0..9 ∪ '\_' ∉ a) → S1 | S124 → ∀ ∪ 0..9 ∪ '\_' ∉ a, S1 |
| (S124, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S124→¬(∀∪0..9∪'\_'), S0 S124→¬(∀∪0..9∪'\_') |
| (S125, m) → S82 | S125 → m, S82 |
| (S125, ∀ ∪ 0..9 ∪ '\_' ∉ m) → S1 | S125 → ∀ ∪ 0..9 ∪ '\_' ∉ m, S1 |
| (S125, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S125→¬(∀∪0..9∪'\_'), S0 S125→¬(∀∪0..9∪'\_') |

Продолжение табл. 2

|  |  |
| --- | --- |
| 1 | 2 |
| (S126, t) → S3 | S126 → t, S3 |
| (S126, ∀ ∪ 0..9 ∪ '\_' ∉ t) → S1 | S126 → ∀ ∪ 0..9 ∪ '\_' ∉ t, S1 |
| (S126, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S126→¬(∀∪0..9∪'\_'), S0 S126→¬(∀∪0..9∪'\_') |
| (S128, 0..9) → S128 | S128 → 0..9, S128 |
| (S128, '.') → S129 | S128 → '.', S129 |
| (S128, e, E) → S59 | S128 → (e, E), S59 |
| (S128, '\n', '\r', ' ', <, >, =, !, –, +, \*, /, %, |, &, ';', ':', (, ), {, }, [, ], #, ',') → S5 | S128 → ('\n', '\r', ' ', <, >, =, !, –, +, \*, /, %, |, &, ';', ':', (, ), {, }, [, ], #, ','), S5 |
| (S129, 0..9) → S131 | S129 → 0..9, S131 |
| (S131, '\n', '\r', ' ', <, >, =, !, –, +, \*, /, %, |, &, ';', ':', (, ), {, }, [, ], #, ',', '.') → S5 | S131 → ('\n', '\r', ' ', <, >, =, !, –, +, \*, /, %, |, &, ';', ':', (, ), {, }, [, ], #, ',', '.'), S5 |
| (S131, 0..9) → S131 | S131 → 0..9, S131 |
| (S131, e, E) → S59 | S131 → (e, E), S59 |
| (S132, '\n', '\r', ' ', <, >, =, !, –, +, \*, /, %, |, &, ';', ':', (, ), {, }, [, ], #, ',') → S5 | S132 → ('\n', '\r', ' ', <, >, =, !, –, +, \*, /, %, |, &, ';', ':', (, ), {, }, [, ], #, ','), S5 |
| (S132, '.') → S129 | S132 → '.', S129 |
| S135 → S0 | S135 → S0 |
| (S136, ':') → S135 | S136 → ':', S135 |
| (S136, ¬':') → S137 | S136) → ¬':', S137 |
| S137 → S0 | S137 → S0 |
| S139 → S0 | S139 → S0 |
| S140 → S0 | S140 → S0 |
| (S141, '=') → S139 | S141 → '=', S139 |
| (S141, ¬'=') → S151 | S141 → ¬'=', S151 |
| (S142, '=') → S139 | S142 → '=', S139 |
| (S142, <) → S152 | S142 → <, S152 |
| (S142, ¬(=, <)) → S140 | S142 → ¬(=, <), S140 |
| (S143, '=') → S139 | S143 → '=', S139 |
| (S143, >) → S152 | S143 → >, S152 |

Продолжение табл. 2

|  |  |
| --- | --- |
| 1 | 2 |
| (S143, ¬(=, >)) → S140 | S143 → ¬(=, >), S140 |
| (S144, +, =) → S152 | S144 → (+, =), S152 |
| (S144, ¬(+, =)) → S151 | S144 → ¬(+, =), S151 |
| (S145, –, =) → S152 | S145 → (–, =), S152 |
| (S145, ¬(–, =)) → S151 | S145 → ¬(–, =), S151 |
| (S146, =) → S152 | S146 → =, S152 |
| (S146, ¬'=') → S151 | S146 → ¬'=', S151 |
| (S147, =) → S152 | S147 → =, S152 |
| (S147, ¬'=') → S151 | S147 → ¬'=', S151 |
| (S148, =) → S152 | S148 → =, S152 |
| (S148, ¬'=') → S151 | S148) → ¬'=', S151 |
| (S149, '|') → S139 | S149 → '|', S139 |
| (S149, =) → S152 | S149 → =, S152 |
| (S149, ¬(|, =)) → S151 | S149 → ¬(|, =), S151 |
| (S150, &) → S139 | S150 → &, S139 |
| (S150, =) → S152 | S150 → =, S152 |
| (S150, ¬(&, =)) → S151 | S150 → ¬(&, =), S151 |
| S151 → S0 | S151 → S0 |
| S152 → S0 | S152 → S0 |
| S153 → S0 | S153 → S0 |
| (S154, =) → S139 | S154 → =, S139 |
| (S154, ¬'=') → S151 | S154 → ¬'=', S151 |
| (S155, '\n', '\r', ' ', <, >, =, !, –, +, \*, /, %, |, &, ';', ':', (, ), {, }, [, ], #, ',', '.') → S156 | S155 → ('\n', '\r', ' ', <, >, =, !, –, +, \*, /, %, |, &, ';', ':', (, ), {, }, [, ], #, ',', '.'), S156 |
| S156 → S0 | S156 → S0 |
| (S157, l) → S158 | S157 → l, S158 |
| (S157, ∀ ∪ 0..9 ∪ '\_' ∉ l) → S1 | S157 → ∀ ∪ 0..9 ∪ '\_' ∉ l, S1 |
| (S157, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S157→¬(∀∪0..9∪'\_'), S0 S157→¬(∀∪0..9∪'\_') |
| (S158, u) → S159 | S158 → u, S159 |
| (S158, ∀ ∪ 0..9 ∪ '\_' ∉ u) → S1 | S158 → ∀ ∪ 0..9 ∪ '\_' ∉ u, S1 |
| (S158, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S158→¬(∀∪0..9∪'\_'), S0 S158→¬(∀∪0..9∪'\_') |

Продолжение табл. 2

|  |  |
| --- | --- |
| 1 | 2 |
| (S159, d) → S24 | S159 → d, S24 |
| (S159, ∀ ∪ 0..9 ∪ '\_' ∉ d) → S1 | S159 → ∀ ∪ 0..9 ∪ '\_' ∉ d, S1 |
| (S159, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S159→¬(∀∪0..9∪'\_'), S0 S159→¬(∀∪0..9∪'\_') |
| (S160, u) → S161 | S160 → u, S161 |
| (S160, ∀ ∪ 0..9 ∪ '\_' ∉ u) → S1 | S160 → ∀ ∪ 0..9 ∪ '\_' ∉ u, S1 |
| (S160, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S160→¬(∀∪0..9∪'\_'), S0 S160→¬(∀∪0..9∪'\_') |
| (S161, b) → S53 | S161 → b, S53 |
| (S161, ∀ ∪ 0..9 ∪ '\_' ∉ b) → S1 | S161 → ∀ ∪ 0..9 ∪ '\_' ∉ b, S1 |
| (S161, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S161→¬(∀∪0..9∪'\_'), S0 S161→¬(∀∪0..9∪'\_') |
| (S162, c) → S163 | S162 → c, S163 |
| (S162, ∀ ∪ 0..9 ∪ '\_' ∉ c) → S1 | S162 → ∀ ∪ 0..9 ∪ '\_' ∉ c, S1 |
| (S162, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S162→¬(∀∪0..9∪'\_'), S0 S162→¬(∀∪0..9∪'\_') |
| (S163, h) → S164 | S163 → h, S164 |
| (S163, ∀ ∪ 0..9 ∪ '\_' ∉ h) → S1 | S163 → ∀ ∪ 0..9 ∪ '\_' ∉ h, S1 |
| (S163, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S163→¬(∀∪0..9∪'\_'), S0 S163→¬(∀∪0..9∪'\_') |
| (S164, a) → S165 | S164 → a, S165 |
| (S164, ∀ ∪ 0..9 ∪ '\_' ∉ a) → S1 | S164 → ∀ ∪ 0..9 ∪ '\_' ∉ a, S1 |
| (S164, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S164→¬(∀∪0..9∪'\_'), S0 S164→¬(∀∪0..9∪'\_') |
| (S165, r) → S98 | S165 → r, S98 |
| (S165, ∀ ∪ 0..9 ∪ '\_' ∉ r) → S1 | S165 → ∀ ∪ 0..9 ∪ '\_' ∉ r, S1 |
| (S165, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S165→¬(∀∪0..9∪'\_'), S0 S165→¬(∀∪0..9∪'\_') |
| (S166, c) → S167 | S166 → c, S167 |
| (S166, ∀ ∪ 0..9 ∪ '\_' ∉ c) → S1 | S166 → ∀ ∪ 0..9 ∪ '\_' ∉ c, S1 |
| (S166, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S166→¬(∀∪0..9∪'\_'), S0 S166→¬(∀∪0..9∪'\_') |
| (S167, t) → S168 | S167 → t, S168 |
| (S167, ∀ ∪ 0..9 ∪ '\_' ∉ t) → S1 | S167 → ∀ ∪ 0..9 ∪ '\_' ∉ t, S1 |
| (S167, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S167→¬(∀∪0..9∪'\_'), S0 S167→¬(∀∪0..9∪'\_') |
| (S168, e) → S12 | S168 → e, S12 |
| (S168, ∀ ∪ 0..9 ∪ '\_' ∉ e) → S1 | S168 → ∀ ∪ 0..9 ∪ '\_' ∉ e, S1 |
| (S168, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S168→¬(∀∪0..9∪'\_'), S0 S168→¬(∀∪0..9∪'\_') |

Продолжение табл. 2

|  |  |
| --- | --- |
| 1 | 2 |
| (S169, i) → S170 | S169 → i, S170 |
| (S169, s) → S174 | S169 → s, S174 |
| (S169, ∀∪0..9∪'\_'∉i,s) → S1 | S169 → ∀ ∪ 0..9 ∪ '\_' ∉ (i, s), S1 |
| (S169, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S169→¬(∀∪0..9∪'\_'), S0 S169→¬(∀∪0..9∪'\_') |
| (S170, o) → S171 | S170 → o, S171 |
| (S170, ∀ ∪ 0..9 ∪ '\_' ∉ o) → S1 | S170 → ∀ ∪ 0..9 ∪ '\_' ∉ o, S1 |
| (S170, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S170→¬(∀∪0..9∪'\_'), S0 S170→¬(∀∪0..9∪'\_') |
| (S171, n) → S3 | S171 → n, S3 |
| (S171, ∀ ∪ 0..9 ∪ '\_' ∉ n) → S1 | S171 → ∀ ∪ 0..9 ∪ '\_' ∉ n, S1 |
| (S171, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S171→¬(∀∪0..9∪'\_'), S0 S171→¬(∀∪0..9∪'\_') |
| (S174, i) → S175 | S174 → i, S175 |
| (S174, ∀ ∪ 0..9 ∪ '\_' ∉ i) → S1 | S174 → ∀ ∪ 0..9 ∪ '\_' ∉ i, S1 |
| (S174, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S174→¬(∀∪0..9∪'\_'), S0 S174→¬(∀∪0..9∪'\_') |
| (S175, g) → S176 | S175 → g, S176 |
| (S175, ∀ ∪ 0..9 ∪ '\_' ∉ g) → S1 | S175 → ∀ ∪ 0..9 ∪ '\_' ∉ g, S1 |
| (S175, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S175→¬(∀∪0..9∪'\_'), S0 S175→¬(∀∪0..9∪'\_') |
| (S176, n) → S168 | S176 → n, S168 |
| (S176, ∀ ∪ 0..9 ∪ '\_' ∉ n) → S1 | S176 → ∀ ∪ 0..9 ∪ '\_' ∉ n, S1 |
| (S176, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S176→¬(∀∪0..9∪'\_'), S0 S176→¬(∀∪0..9∪'\_') |
| (S177, l) → S178 | S177 → l, S178 |
| (S177, ∀ ∪ 0..9 ∪ '\_' ∉ l) → S1 | S177 → ∀ ∪ 0..9 ∪ '\_' ∉ l, S1 |
| (S177, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S177→¬(∀∪0..9∪'\_'), S0 S177→¬(∀∪0..9∪'\_') |
| (S178, l) → S179 | S178 → l, S179 |
| (S178, ∀ ∪ 0..9 ∪ '\_' ∉ l) → S1 | S178 → ∀ ∪ 0..9 ∪ '\_' ∉ l, S1 |
| (S178, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S178→¬(∀∪0..9∪'\_'), S0 S178→¬(∀∪0..9∪'\_') |
| (S179, p) → S180 | S179 → p, S180 |

Продолжение табл. 2

|  |  |
| --- | --- |
| 1 | 2 |
| (S179, ∀ ∪ 0..9 ∪ '\_' ∉ p) → S1 | S179 → ∀ ∪ 0..9 ∪ '\_' ∉ p, S1 |
| (S179, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S179→¬(∀∪0..9∪'\_'), S0 S179→¬(∀∪0..9∪'\_') |
| (S180, t) → S30 | S180 → t, S30 |
| (S180, ∀ ∪ 0..9 ∪ '\_' ∉ t) → S1 | S180 → ∀ ∪ 0..9 ∪ '\_' ∉ t, S1 |
| (S180, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S180→¬(∀∪0..9∪'\_'), S0 S180→¬(∀∪0..9∪'\_') |
| (S181, a) → S182 | S181 → a, S182 |
| (S181, ∀ ∪ 0..9 ∪ '\_' ∉ a) → S1 | S181 → ∀ ∪ 0..9 ∪ '\_' ∉ a, S1 |
| (S181, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S181→¬(∀∪0..9∪'\_'), S0 S181→¬(∀∪0..9∪'\_') |
| (S182, t) → S183 | S182 → t, S183 |
| (S182, ∀ ∪ 0..9 ∪ '\_' ∉ t) → S1 | S182 → ∀ ∪ 0..9 ∪ '\_' ∉ t, S1 |
| (S182, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S182→¬(∀∪0..9∪'\_'), S0 S182→¬(∀∪0..9∪'\_') |
| (S183, i) → S53 | S183 → i, S53 |
| (S183, ∀ ∪ 0..9 ∪ '\_' ∉ i) → S1 | S183 → ∀ ∪ 0..9 ∪ '\_' ∉ i, S1 |
| (S183, ¬(∀ ∪ 0..9 ∪ '\_')) → S0 | S183→¬(∀∪0..9∪'\_'), S0 S183→¬(∀∪0..9∪'\_') |
| (S184, ¬(0..9)) → S5 | S184 → ¬(0..9), S5 |
| (S184, '\n', '\r', ' ', <, >, =, !, –, +, \*, /, %, |, &, ';', ':', (, ), {, }, [, ], #, ',', '.') → S130 | S184 → ('\n', '\r', ' ', <, >, =, !, –, +, \*, /, %, |, &, ';', ':', (, ), {, }, [, ], #, ',', '.'), S130 |

## Построение КС-грамматики

G = (T, N, P, S)

T = { int, double, size\_t, string, union, struct, id, const, cout, cin, endl, ‘,’, ‘.’, ‘;’, ‘"’, =, (, ), {, }, <}

N = {A, U, K, B, C, E, D, M, X, V, I, R, F, Y, Z, W}

S = A

P:

A → AA | U | C | Z

U → union I{K}B

K → KK | X;

B → ; | I; | I = {R};

C → struct I{E}D

E → EE | U | C | X;

D → ; | I; | I = {M};

M → V, M | V

X → YI | YI = V

V → I | “I” | R

I → id

R → const | F const

F → + | –

Y → int | double | size\_t | string

Z → cout W;

W → << VW | << V | << endl W | << endl



Рис. 6. Тестовая цепочка

Построение порождения:

**A** ⇒ A**A** ⇒ A**Z** ⇒ A cout **W**; ⇒ A cout << V**W**; ⇒ A cout <<**V**<< endl; ⇒ A cout<< “**I**” << endl; ⇒ **A** cout << “id” << endl; ⇒ **C** cout << “id” << endl; ⇒ struct I{E}**D** cout << “id” << endl; ⇒ struct I{**E**}; cout <<“id”<<endl; ⇒ struct I{E**E**}; cout<<“id” << endl; ⇒ struct I{EE**E**}; cout << “id” << endl; ⇒ struct I{EE**X**;}; cout << “id” << endl; ⇒ struct I{EE YI = **V**;}; cout << “id” << endl; ⇒ struct I{EE YI = **R**;}; cout << “id” << endl; ⇒ struct I{EE YI = **F** const;}; cout << “id” <<endl; ⇒ struct I{EE Y**I** = –const;}; cout << “id” << endl; ⇒ struct I{EE **Y** id = –const;}; cout << “id” << endl; ⇒ struct I{E**E** int id = –const;}; cout << “id”<<endl; ⇒ struct I{E**X**; int id = –const;}; cout << “id” << endl; ⇒ struct I{E YI = **V**; int id = –const;}; cout << “id” << endl; ⇒ struct I{E YI = “**I**”; int id = –const;}; cout << “id” << endl; ⇒ struct I{E Y**I** = “id”; int id = –const;}; cout << “id” << endl; ⇒ struct I{E **Y** id = “id”; int id = –const;}; cout<<“id”<<endl; ⇒ struct I{**E** string id=“id”;int id=–const;}; cout<<“id”<<endl; ⇒ struct I{U string id = “id”; int id = –const;}; cout << “id” << endl; ⇒ struct I{ union I {K}**B** string id = “id”; int id = –const;}; cout << “id” << endl; ⇒ struct I{ union I{K} I = {**R**}; string id = “id”; int id = –const;}; cout << “id” << endl; ⇒ struct I{ union I {K} I = {**F** const}; string id = “id”; int id = –const;}; cout << “id” << endl; ⇒ struct I { union I{K} **I** = {–const}; string id = “id”; int id = –const;}; cout<< “id” << endl; ⇒ struct I{ union I{**K**} id = {–const}; string id = “id”; int id = –const;}; cout << “id” << endl; ⇒ struct I{ union I{K**K**} id = {–const}; string id=“id”; int id= –const;}; cout << “id” <<endl; ⇒ struct I{ union I{KK**K**}id = {–const}; string id=“id”; int id=–const;}; cout<< “id” << endl; ⇒ struct I{ union I{KK**X**;} id = {–const}; string id = “id”; int id = –const;}; cout<< “id” << endl; ⇒ struct I{ union I{KK Y**I**;} id = {–const}; string id = “id”; int id = –const;}; cout << “id” << endl; ⇒ struct I { union I{KK **Y** id;} id = {–const}; string id = “id”; int id = –const;}; cout << “id” << endl; ⇒ struct I{ union I {K**K** double id;} id = {–const}; string id = “id”; int id=–const;}; cout <<“id”<< endl; ⇒ struct I{ union I{K**X**; double id;} id = {–const}; string id = “id”; int id = –const;}; cout << “id” << endl; ⇒ struct I{ union I{K Y**I**; double id;} id = {–const}; string id = “id”; int id = –const;}; cout << “id” << endl; ⇒ struct I{ union I{K **Y** id; double id;} id = {–const}; string id = “id”; int id = –const;}; cout << “id” << endl; ⇒ struct I{ union I{**K** size\_t id; double id;} id = {–const}; string id = “id”; int id = –const;}; cout << “id” << endl; ⇒ struct I{ union I{**X**; size\_t id; double id;} id = {–const}; string id = “id”; int id = –const;}; cout << “id”<<endl; ⇒ struct I{union I{Y**I**; size\_t id; double id;} id = {–const}; string id = “id”; int id = –const;}; cout << “id” << endl; ⇒ struct I { union I{**Y** id; size\_t id; double id;} id = {–const}; string id = “id”; int id = –const;}; cout << “id” << endl; ⇒ struct I{ union **I**{int id; size\_t id; double id;} id = {–const}; string id = “id”; int id = –const;}; cout << “id” << endl; ⇒ struct **I**{ union id{int id; size\_t id; double id;}id = {–const}; string id = “id”; int id = –const;}; cout << “id” << endl; ⇒ struct id{ union id{int id; size\_t id; double id;}id = {–const}; string id = “id”; int id = –const;}; cout << “id” << endl;

Дерево вывода представлено на рис. 7.

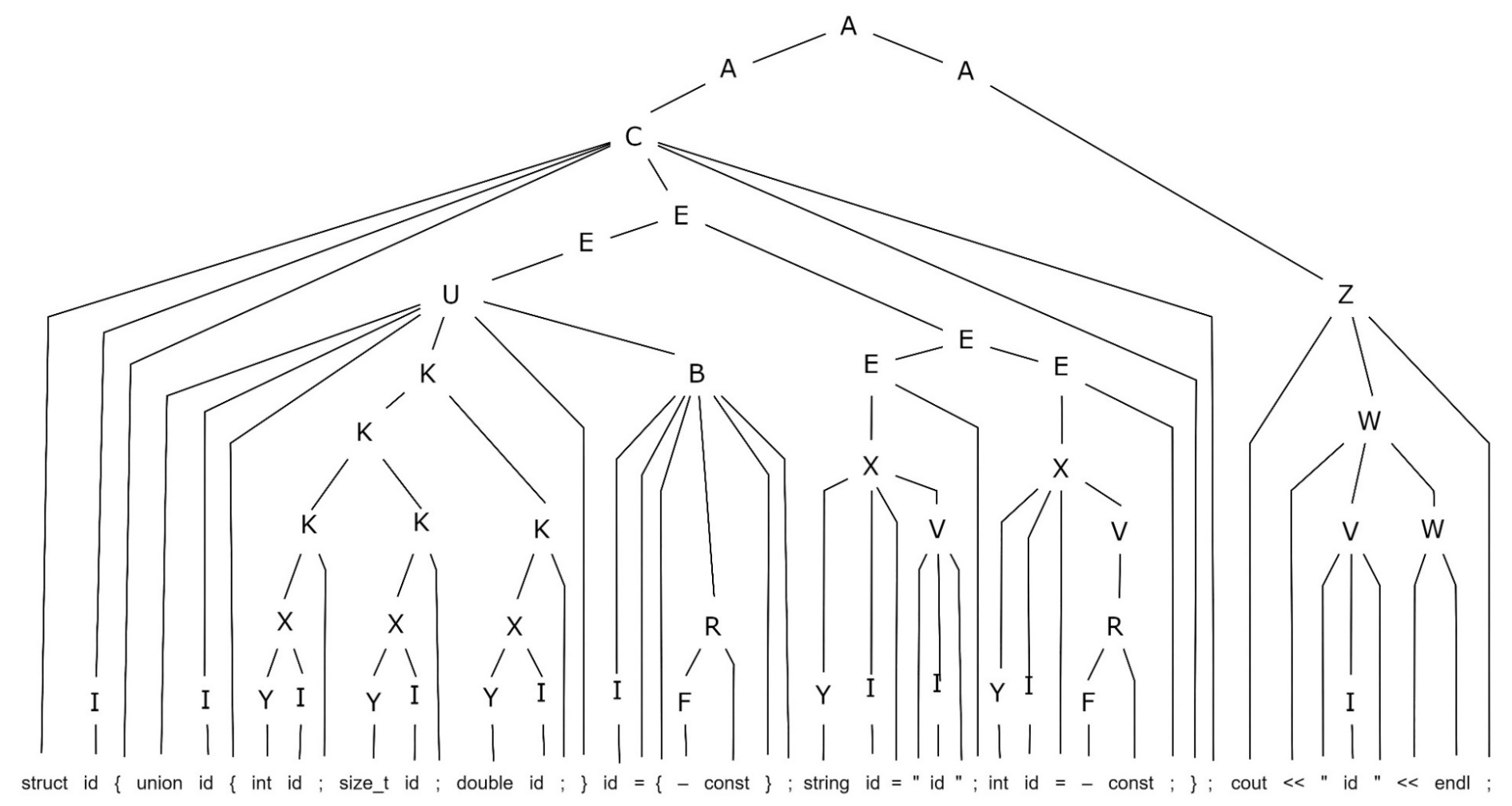


Рис. 7. Дерево вывода

# Логическое проектирование

## Блок-схемы алгоритмов

Блок-схема общего алгоритма работы программы представлена на рис. 8.

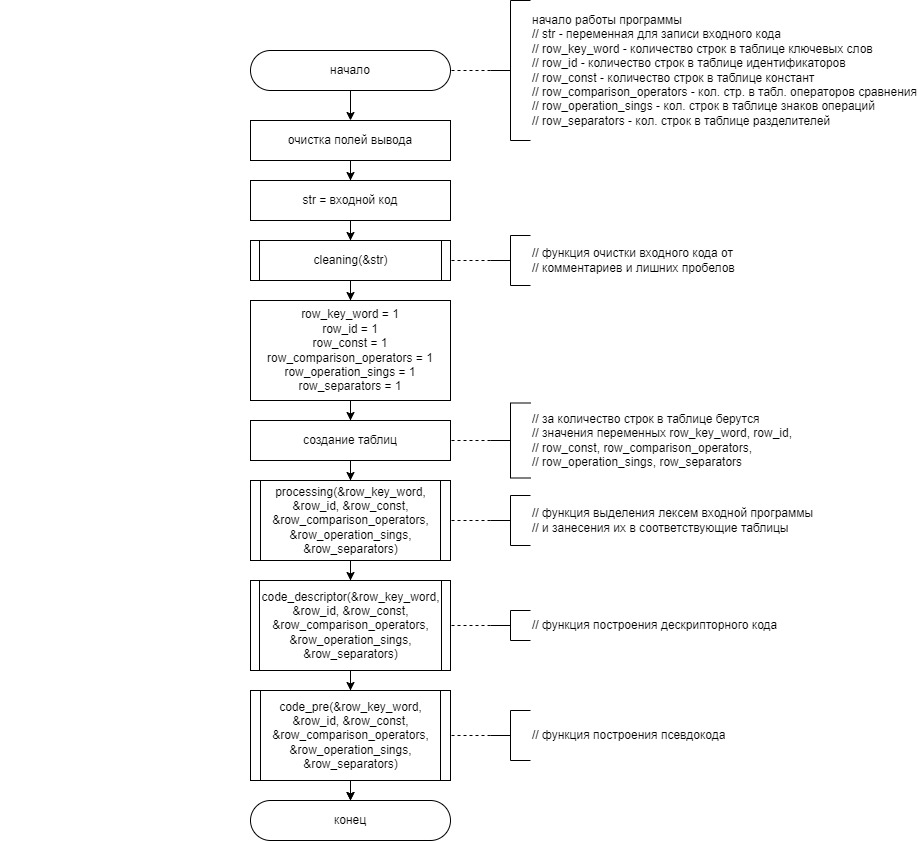


Рис. 8. Блок-схема общего алгоритма работы программы

Примерная блок-схема функции выделения лексем представлена на рис. 9.

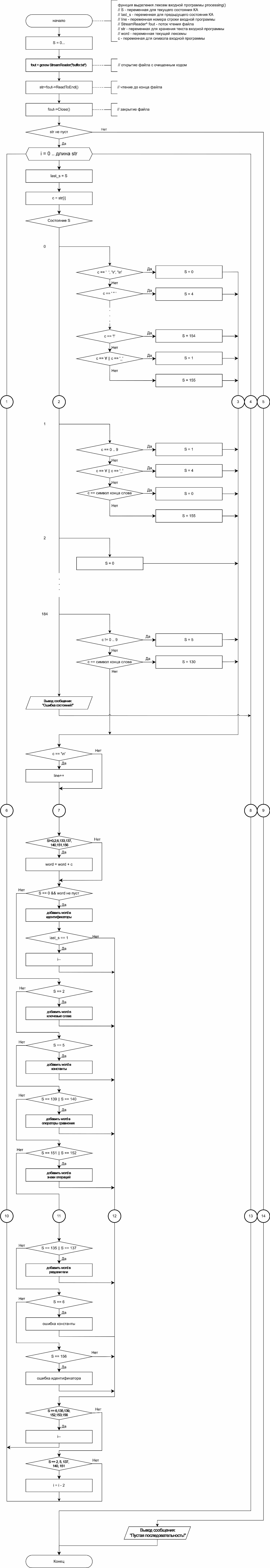
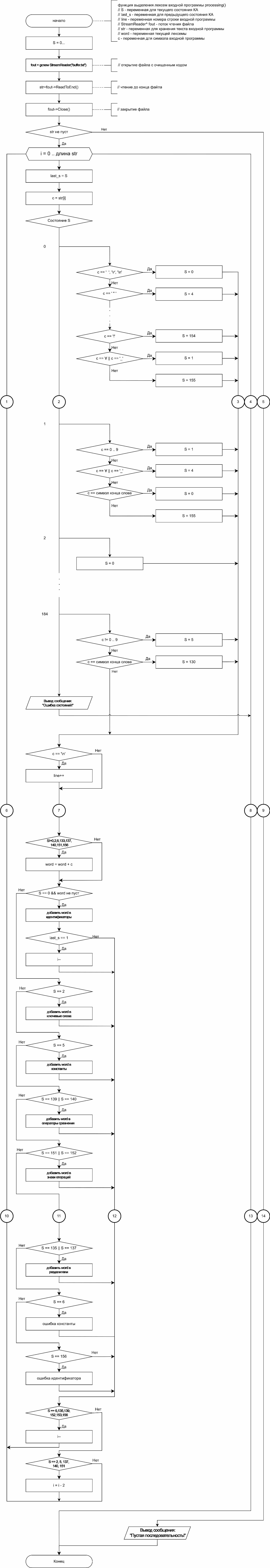
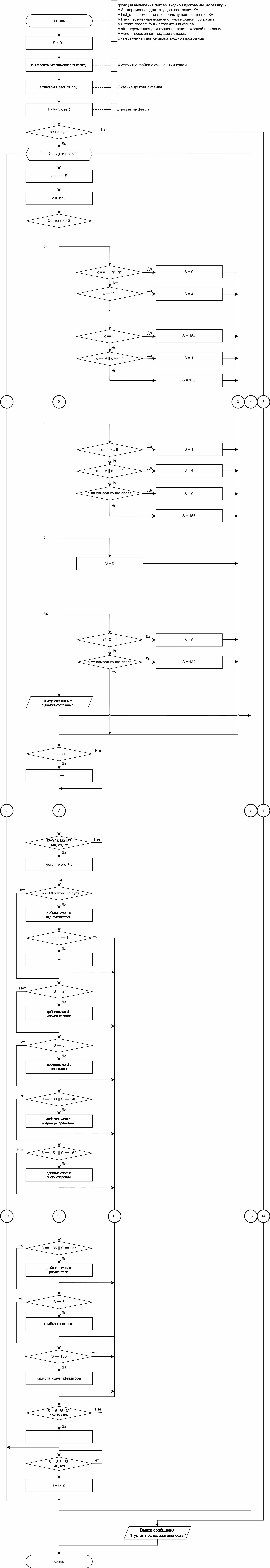


Рис. 9. Примерная блок-схема выделения лексем

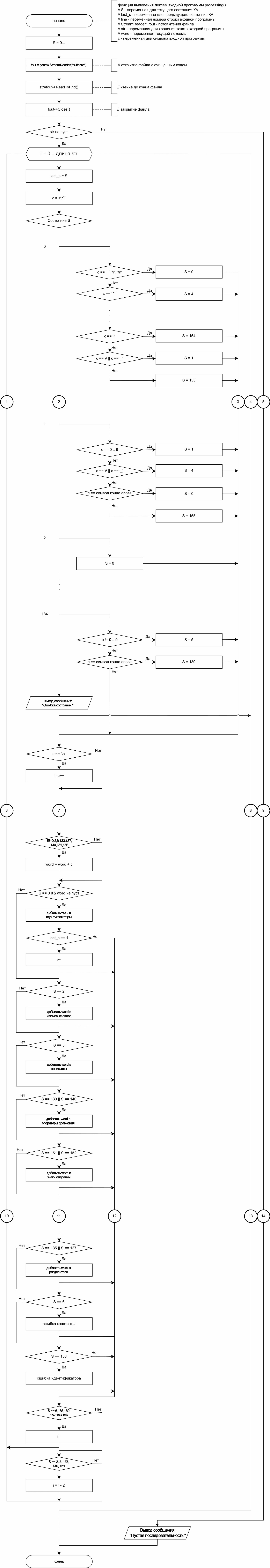
Продолжение рис. 9



Продолжение рис. 9



Продолжение рис. 9



## Словесное описание алгоритмов

Общий алгоритм работы программы:

1. Чтение входного кода.
2. Очистка кода от комментариев и лишних пробелов.
3. Выделение лексем из входной программы, запись лексем в соответствующие таблицы.
4. Построение дескрипторного кода и псевдокода.

Алгоритм работы оператора выбора:

1. Запись очищенного кода в переменную.
2. Посимвольное чтение очищенного кода:
   1. Чтение символа.
   2. Смена состояния в зависимости от символа и текущего состояния.
   3. Выполнние действий в зависимости от текущего состояния.

Для примера работы алгоритмов созданы тестовые входные последовательности на рис. 10.



Рис. 10. Тестовые данные

Этапы для кода 1):

* Чтение входного кода и очистка от коментариев (в данном случае их нет).
* Посимвольное чтение лексем: «int» и «size\_t» заносятся в таблицу ключевых слов, «148» – в таблицу констант, «(», «)», «{», «}» и «;» – в таблицу разделителей, «=» – в таблицу знаков операций, «a» и «main» – в таблицу идентификаторов.
* Дескр. код: (10,1)(20,1)(60,1)(60,2)(60,3)(10,2)(20,2)(50,1)(30,1)(60,4)(60,5), псевдокод: int id0 ( ) { size\_t id1 = const0 ; }

Этапы для кода 2) не отличаются от этапов кода 1), за исключением:

* Посимвольное чтение лексем: «int» и «double» заносятся в таблицу ключевых слов, «main» – в таблицу идентификаторов, а при чтении «st@r» возникает ошибка.
* Дескриптор. код: (10,1)(20,1)(60,1)(60,2)(60,3)(10,2)(50,1)(30,1)(60,4)(60,5), псевдокод: int id0 ( ) { double = const0 ; }

Коды 1) и 2) почти одинаковы, но появление неразрешённого символа «@» вызывает ошибку, программа игнорирует занесение в таблицу слова «st@r» и выводит сообщение об ошибке, из-за чего дескрипторные коды и псевдокоды программ различаются (нет одного идентификатора) при одинаковой структуре.

## Оценка сложности алгоритма

Сложность алгоритма — это количество элементарных операций (шагов цикла), где n – объём входных данных. Сложность алгоритмов обычно оценивают по времени выполнения или по используемой памяти, однако больше внимания уделяют асимптотической сложности (сложности при стремлении размера входных данных к бесконечности).

Сложность общего алгоритма программы: O(n2), где n – длина кода входной программы.

Обоснование:

1. Функция «cleaning» содержит один цикл, проходя через весь входной код, значит сложность функции O(n), где n – длина кода входной программы.
2. функция «processing» имеет также один цикл, значит сложность функции O(n), где n – длина кода очищенной программы (после функции «cleaning»).
3. Функция «code\_descriptor» имеет один главный цикл и вложеные в него циклы, которые либо не влияют на скорость работы главного, либо отвечают за проход по элементам таблицы. Размер таблиц ключевых слов, разделителей, знаков операций и операторов сравнения может быть известно (зависит от языка программирования), но количество констант и идентификаторов – нет, значит сложность функции O(n2), где n – длина кода очищенной программы.
4. Функция «code\_pre» имеет один главный цикл и вложеные в него циклы, которые не влияют на скорость работы главного, значит сложность функции O(n), где n – длина дескрипторного кода программы.

Сложив сложности получаем O(n) + O(n) + O(n2) + O(n). При стремлении n к бесконечности большее влияние оказывает O(n2), значит сложность равна O(n2).

# Физическое проектирование

1. Выбор структур данных

В программе используются типы данных, представленные в табл. 3.

Таблица 3

Структуры данных

|  |  |  |
| --- | --- | --- |
| Наименование | Обозначение | Тип данных |
| Номер состояния | S | int |
| Номер предыдущего состояния | last\_s | int |
| Номер строки | line | int |
| Количество строк в таблице ключевых слов | row\_key\_word | int |
| Количество строк в таблице идентификаторов | row\_id | int |
| Количество строк в таблице констант | row\_const | int |
| Количество строк в таблице операторов сравнения | row\_comparison\_  operators | int |
| Количество строк в таблице знаков операций | row\_operation\_sings | int |
| Количество строк в таблице разделителей | row\_separators | int |
| Буфер для хранения символа входного кода | с | wchar\_t |
| Буфер для хранения символа входного кода | a | String^ |
| Буфер для хранения входного кода | str | String^ |
| Буфер для хранения входной лексемы | word | String^ |
| Буфер для хранения дескрипторного кода элемента | out | String^ |
| Файл для хранения очищенного входного кода | fsave | StreamWriter^ |
| Файл для записи очищенного входного кода с номером строки | ftext | StreamWriter^ |
| Файл для записи дескрипторного кода | fcd | StreamWriter^ |
| Файл для записи псевдокода | fcp | StreamWriter^ |
| Файл для вывода очищенного входного кода с номером строки | fout | StreamReader^ |
| Файл для вывода дескрипторного кода | fcdout | StreamReader^ |
| Файл для вывода псевдокода | fcpout | StreamReader^ |

1. Спецификация функций

Используемые в программе функции представлены в табл. 4.

Таблица 4

Спецификация функций

|  |  |  |  |
| --- | --- | --- | --- |
| Имя модуля | Заголовок процедуры  или функции | Формальные  параметры | Выполняемое  действие |
| 1 | 2 | 3 | 4 |
| Поиск идентификатора | search\_id | String^\* word, int\* row | Ищет идентификатор в таблице |
| Добавление идентификатора | add\_id | String^\* word, int\* row\_id, int\* row\_keyword | Добавляет идентификатор в таблицу |
| Поиск ключевого слова | search\_keyword | String^\* word, int\* row | Ищет ключевое слово в таблице |
| Добавление ключевого слова | add\_keyword | String^\* word, int\* row | Добавляет ключевое слово в таблицу |
| Поиск константы | search\_const | String^\* word, int\* row | Ищет константу в таблице |
| Добавление константы | add\_const | String^\* word, int\* row | Добавляет константу в таблицу |
| Поиск оператора сравнения | search\_comparison  \_operators | String^\* word, int\* row | Ищет оператор сравнения в таблице |
| Добавление оператора сравнения | add\_comparison  \_operators | String^\* word, int\* row | Добавляет оператор сравнения в таблицу |
| Поиск знака операции | search\_operation  \_sings | String^\* word, int\* row | Ищет знак операции в таблице |
| Добавление знака операции | add\_operation\_sings | String^\* word, int\* row | Добавляет знак операции в таблицу |
| Поиск разделителя | search\_separators | String^\* word, int\* row | Ищет разделитель в таблице |
| Добавление разделителя | add\_separators | String^\* word, int\* row | Добавляет разде-литель в таблицу |

Продолжение табл. 4

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| Пропуск | skip | wchar\_t\* c | Возвращает является ли символ ' ', '\n', '\r' |
| Операторы сравнения | compariso  \_operators | wchar\_t\* c | Возвр. явл. ли сим. оператором сравнения |
| Знаки операций | operation\_  sings | wchar\_t\* c | Возвр. явл. ли сим. знаком операций |
| Разделители | separators | wchar\_t\* c | Возвр. явл. ли сим. разделителем |
| Прочее | other | wchar\_t\* c | Возвр. явл. ли сим. ‘ . ’ |
| Конец строки | end\_  condition | wchar\_t\* c | Возвр. результат работы skip, separators, other, comparison\_operators и operation\_sings |
| Разрешённые символы | allowed\_  character | wchar\_t\* c | Возвр. явл. ли сим. допустимым для записи идентификатора |
| Цифры кроме 0 | numbers | wchar\_t\* c | Возвр. явл. ли сим. цифрой от 1 до 9 |
| Ошибка | error | String^\* word, String^ type, int\* line | Обработчик ошибок |
| Очистка | cleaning | String^\* str | Очищает код от комментариев, пробелов |
| Обработка | processing | int\* row\_key\_word, int\* row\_id, int\* row\_const, int\* row\_comparison  \_operators, int\* row\_operation\_  sings, int\* row\_separators | Чтение лексем, запись лексем в таблицы |
| Дескриптор-ный код | code\_  descriptor | int\* row\_key\_word, int\* row\_id, int\* row\_const, int\* row\_comparison  \_operators, int\* row\_operation\_  sings, int\* row\_separators | Построение дескрипторного кода |
| Псевдокод | code\_pre | int\* row\_key\_word, int\* row\_id, int\* row\_const, int\* row\_comparison  \_operators, int\* row\_operation\_  sings, int\* row\_separators | Построение псевдокода |
| Старт | run\_Click | System::Object^ sender, System::EventArgs^ e | Начало работы программы |

# Проектирование интерфейса

Для разработки интерфейса используется среда CLR, которая позволяет легко загружать входные данные и в понятной форме представлять результаты работы программы. Тип интерфейса: визуальный (графический).

В интерфейсе содержатся поля ввода исходного кода и вывода очищенного кода с указанием номеров строк; 6 таблиц для 6 групп лексем: ключевые слова, идентификаторы, константы, разделители, знаки операций, операторы сравнения; поля для вывода дескрипторного и псевдокода; поля для вывода дескрипторного и псевдо- кодов; поле для вывода ошибок; кнопка «Старт» для начала работы программы. Итоговый вид интерфейса представлен на рис. 11.

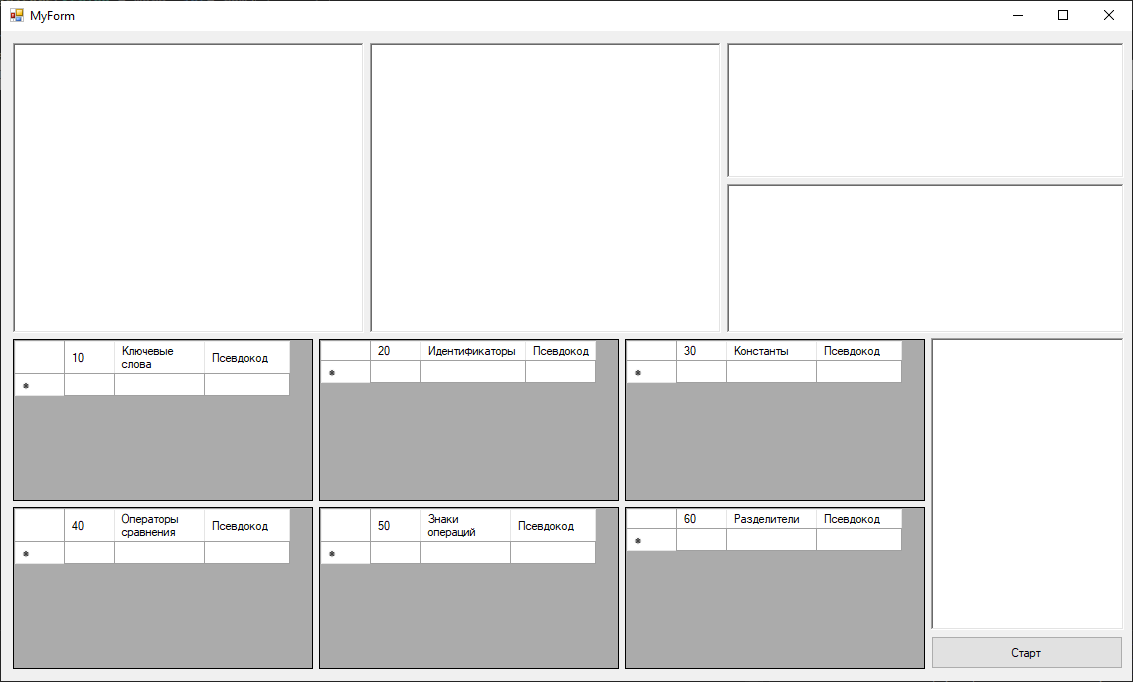


Рис. 11. Интерфейс программы

# Тестирование программного обеспечения

Для тестирования программы были подобраны тестовые данные в табл. 5.

Таблица 5

Тестовые данные

|  |  |  |
| --- | --- | --- |
| Исходные данные | Тестируемый модуль или подпрограмма | Ожидаемый результат |
| 1 | 2 | 3 |
| Текст с лишними пробелами, однострочными и многострочными комментариями | Очистка | Содержимое текста очищено от комментариев, а между словами только по одному пробелу |
| Колонка из всех предусмотренных ключевых слов | Ключевые слова | Занесение всех ключевых слов в соответствующую таблицу |
| 1e-abc  2  67.89  00.10  0.11.12  13.14e+00000  15e-16  -17.18+19++  20..21  35.16e-+02  \_15\_x  <<++>>  =>=<  int44  abc@d | Обработка и Ошибка | Занесение всех корректных данных в соответствующие таблицы и вывод некорректных в поле ошибок с номером строки и типом ошибки |
| { int x = 1; } | Дескрипторный код | (60,1)(10,1)(20,1)(50,1)(30,1) (60,2)(60,3) |
| (60,1)(10,1)(20,1)(50,1)(30,1) (60,2)(60,3) | Псевдокод | { int id0 = const0 ; } |

Результаты тестирования представлены в табл. 6.

Таблица 6

Результаты тестирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Дата и время  тестиро-вания | Тестируемый  модуль или  подпрограмма | Кто  проводил  тестирова-ние | Описание  теста | Результаты  тестирования |
| 20.05.24 | Очистка | Остапенко С. Я |  | Успех |
| 20.05.24 | Ключевые слова | Остапенко С. Я | Ввод всех предусмотренных ключевых слов | Успех |
| 20.05.24 | Обработка и Ошибка | Остапенко С. Я | Ввод корректных и некорректных данных | Успех |
| 20.05.24 | Дескрипторный код | Микуцких Г. А. | Ввод корректного кода, | Успех |
| 20.05.24 | Псевдокод | Микуцких Г. А. | Образованный дескрипторный код поступает на обработку | Успех |

# Заключение

В ходе куровой работы построен конечный автомат , моделирующий работу этапа лексического анализа компилятора, соответствующая ему регулярная грамматика, разработана программа, моделирующая работу алексического анализатора и построена КС-грамматика, описывающая синтаксис инструкций объединения и структуры («union» и «struct»).

Материал, полученный в ходе изучения дисциплин «Теория автоматов и формальных языков» и «Структуры и алгоритмы обработки данных», освоен.

# Список литературы

[1] Карпов Ю. Г. Теория и технология программирования. Основы построения трансляторов. – СПб.: БХВ-Петербург, 2005. – 272с.: ил.

[2] Опалева Э. А., Самойленко В. П. Языки программирования и методы трансляции. – СПб.: БХВ-Петербург, 2005. – 480с.: ил.

[3] Ганичева О. Г. Теория языков программирования и методы трансляции: Учеб. пособие. – Череповец: ГОУ ВПО ЧГУ, 2011. – 186 с.

[4] <https://studfile.net/preview/9763175/page:53/>

# Приложение 1

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное

образовательное учреждение высшего образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

­­ Институт информационных технологий

наименование института (факультета)

Математическое и программное обеспечение ЭВМ

наименование кафедры

Информатика

наименование дисциплины в соответствии с учебным планом

УТВЕРЖДАЮ

Зав. кафедрой МПО ЭВМ,

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

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г.

РАЗРАБОТКА АЛГОРИТМИЧЕСКОГО ОБЕСПЕЧЕНИЯ И ПОСТРОЕНИЕ ЛЕКСИЧЕСКОГО АНАЛИЗАТОРА КОМПИЛЯТОРА

Техническое задание на курсовую работу

Листов 8

|  |  |
| --- | --- |
| Руководители | Ганичева О. Г. |
|  | Ф.И.О преподавателя  Пышницкий К. М. |
|  | Ф.И.О преподавателя |
| Исполнитель |  |
| студент | 1ПИб-02-1оп-22 |
|  | группа |
|  | Микуцких Г. А. |
|  | Фамилия, имя, отчество |

2024 год

Введение

Чтобы разбираться в устройстве современных компиляторов, необходимо научиться их создавать. Целью курсовой работы является разработка программы, которая будет имитировать работу лексического анализатора компилятора на основе конечного автомата, и создание регулярной грамматики для этого конечного автомата и грамматики для синтаксического анализатора.

1. Основания для разработки

Основанием для разработки является задание на курсовую работу по дисциплине «Теория автоматов и формальных языков» и «Структуры и алгоритмы обработки данных», выданное на кафедре МПО ЭВМ ИИТ ЧГУ.

Дата утверждения: 13 февраля 2024 года.

Наименование темы разработки: «Разработка алгоритмического обеспечения и построение лексического анализатора компилятора».

1. Назначение разработки

Освоение на практике материала, полученного в ходе изучения дисциплин «Теория автоматов и формальных языков» и «Структуры и алгоритмы обработки данных».

1. Требования к программе
2. Требования к функциональным характеристикам

Программа должна выполнять следующие основные функции:

1. загрузка кода через интерфейс программы;
2. исключение из текста исходной программы комментариев (однострочных и многострочных), незначащих пробелов, символов-табуляции; подсчёт строк исходного текста программы;
3. распознавание классов лексем: ключевые слова, идентификаторы, константы, разделители, операторы, операторы сравнения; формирование таблиц классов лексем и их элементов;
4. формирование дескрипторного кода и псевдокода;
5. обнаружение, диагностирование и локализация лексических ошибок.

Дополнительные требования:

1. построить конечный автомат для моделирования работы этапа лексического анализа компилятора при работе со структурами и объединениями языка С++ (описание автомата сделать с использованием формальной модели и таблицы переходов);
2. по конечному автомату построить соответствующую ему регулярную грамматику, результат оформить в таблицу, отражающую соответствие между правилами конечного автомата и продукциями регулярной грамматики;
3. построить КС-грамматику, описывающую синтаксис инструкции языка программирования при работе со структурами и объединениями языка С++;
4. доказать возможность вывода заданной синтаксической конструкции с помощью КС-грамматики разработанной в п.3. Результатом работы синтаксического анализатора должно быть дерево вывода и соответствующее ему порождение рассматриваемой цепочки;
5. разработать блок-схемы общего алгоритма работы программы, определить и обосновать сложность алгоритма;
6. построить подробные блок-схемы работы оператора выбора, на котором основана работа конечного автомата;
7. выполнить словесное описание работы алгоритма на примере разбора тестовых входных последовательностей (лексически правильной и неправильной).
8. Требования к структуре программы

* должна быть выполнена предобработка исходного текста программы;
* программа должна быть реализована с использованием конструкций «switch-case» и «if-else».

1. Требования к надежности

Программа должна обрабатывать все символы текста исходной программы без ошибок.

1. Условия эксплуатации

Предъявляются с условиями эксплуатации ПК пользователя.

1. Требования к составу и параметрам технических средств

ЭВМ должна соответствовать следующим параметрам:

* исходные данные и результаты представляются в виде текста;
* минимальные системные требования:

1. 32-разрядный (x86) или 64-разрядный (x64) процессор с тактовой частотой 1 ГГц или выше;
2. 1 ГБ (для 32-разрядного процессора) или 2 ГБ (для 64-разрядного процессора) ОЗУ, 16 ГБ (для 32-разрядной системы) или 20 ГБ (для 64-разрядной системы) свободного места на жестком диске;
3. графическое устройство DirectX 9 с драйвером WDDM 1.0 или более поздней версии;

* поддержка клавиатуры, компьютерной мыши, монитора разрешением 1920×1080 пикселей.

1. Требования к информационной и программной совместимости

Программа должна корректно работать на ОС Windows 10 и выше.

1. Требования к маркировке и упаковке

Архив с программой будет распространяться в социальной сети «ВКонтакте» в личных чатах, на USB-флеш-накопителе и CD-диске.

1. Требования к транспортированию и хранению

Всем пользователям запрещается осуществлять действия под видом оригинального ПО, нарушающие статью 273 «Создание, использование и распространение вредоносных компьютерных программ».

1. Специальные требования

Для эффективной работы с программой рекомендуется наличие опыта нахождения в компьютерной и информационной средах.

1. Требование к программной документации
2. Содержание расчётно-пояснительной записки

Программная документация должна содержать расчётно-пояснительную записку, которая содержит:

Титульный лист

Аннотацию

Оглавление

Введение

Описание предметной области.

Логическое проектирование

Физическое проектирование

Тестирование

Заключение

Список литературы

Приложения.

1. Технико-экономические показатели

Требования не предъявляются.

1. Требования к оформлению

Элементы курсовой работы оформлены в соответствии с табл. П1.1.

Таблица П1.1

Требования к оформлению

|  |  |
| --- | --- |
| Элемент | Требования |
| Документ | 1. Печать на отдельных листах формата А4 (210х297 мм); оборотная сторона не заполняется; листы нумеруются. Печать возможна ч/б.  2. Файлы предъявляются на компакт-диске: РПЗ с ТЗ; программный код.  3. Листы и диск в конверте вложены в пластиковую папку скоросшивателя. |
| Страницы | 1. Ориентация – книжная; отдельные страницы, при необходимости, альбомная.  2. 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 программного кода – 8 (возможно в 2 колонки). |
| Рисунки | Подписывается под ним по центру: Рис. Х. Название  В приложениях: Рис. ПX.X. Название |
| Таблицы | 1. Подписывается: над таблицей, выравнивание по правому: «Таблица Х».  2. В следующей строке по центру Название  3. Надписи в «шапке» (имена столбцов, полей) – по центру.  4. В теле таблицы (записи) текстовые значения – выровнены по левому краю, числа, даты – по правому. |

1. Стадии и этапы разработки

Курсовая работа и программа будут разрабатываться в течение следующих этапов (табл. П1.2):

Таблица П1.2

Стадии и этапы разработки

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование  этапа разработки | Сроки разработки | Результат выполнения | Отметка о выполнении |
| 1 | 2 | 3 | 4 |
| Оформление технического задания | 07.03.24 | Составлено техническое задание |  |
| Написание функции лексического анализатора (ЛА) | 07.03.24 – 14.03.24 | Функция удаления лишних пробелов, комментариев и подсчёта количества строк из текста программы написана |  |
| Построить конечный автомат (КА) для распознавания идентификаторов и знаков операций | 14.03.24 – 21.03.24 | Построен КА и написана функция по распознаванию |  |
| Написать функцию ЛА для обработки строк на основе операторов выбора и формирования таблиц с классами лексем | 21.03.24 – 28.04.24 | Функция написана |  |
| Построение регулярной грамматики, реализация формирования дескрипторного и псевдокода | 28.04.24 – 19.05.24 | РГ построена, функции реализованы |  |
| Обработка лексических ошибок, построение КС-грамматики | 28.04.24 – 19.05.24 | КС-грамматика, правостороннее порождение и дерево вывода построены, обработка лексических ошибок реализована |  |

Продолжение табл. П1.2

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| Тестирование | 20.05.24 | Отладка программы, доработка функционала |  |
| Оформление расчётно-поясни-тельной записки | 28.04.24 – 14.06.24 | Оформлена расчётно-пояснительная записка |  |

1. Порядок контроля и приемки

Курсовая работа будет оформляться в течение этапов в табл. П1.3.

Таблица П1.3

Порядок контроля и приемки

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование контрольного этапа выполнения курсовой работы | Сроки  контроля | Результат выполнения | Отметка о приемке результата контрольного этапа |
| Сдача технического задания | 10.04.24 | Техническое задание принято |  |
| Исправление ошибок | 01.05.24 –  20.05.24 | Функционал программы доработан, тестирование прошло успешно |  |
| Демонстрация программы | 25.05.24 | Программа, реализующая поставленную задачу, принята |  |
| Сдача расчётно-пояснительной записки | 15.06.24 | Расчётно-пояснительная записка принята |  |
| Защита курсовой работы | 15.06.24 | Курсовая работа защищена |  |

# Приложение 2

Руководство пользователя.

Общие сведения о программе

Программа написана на языке программирования C++ и проводит этап лексического анализа программного кода пользователя.

Описание установки

Установки не требует. В случае возникновения проблем нужно установить актуальную версию «Microsoft Visual C++ Redistributable» от 2022 года.

Описание запуска

Ярлык «KursTA.exe» запустит программу. После запуска Вас встретит окно, где будет проходить вся работа (рис. П2.1).

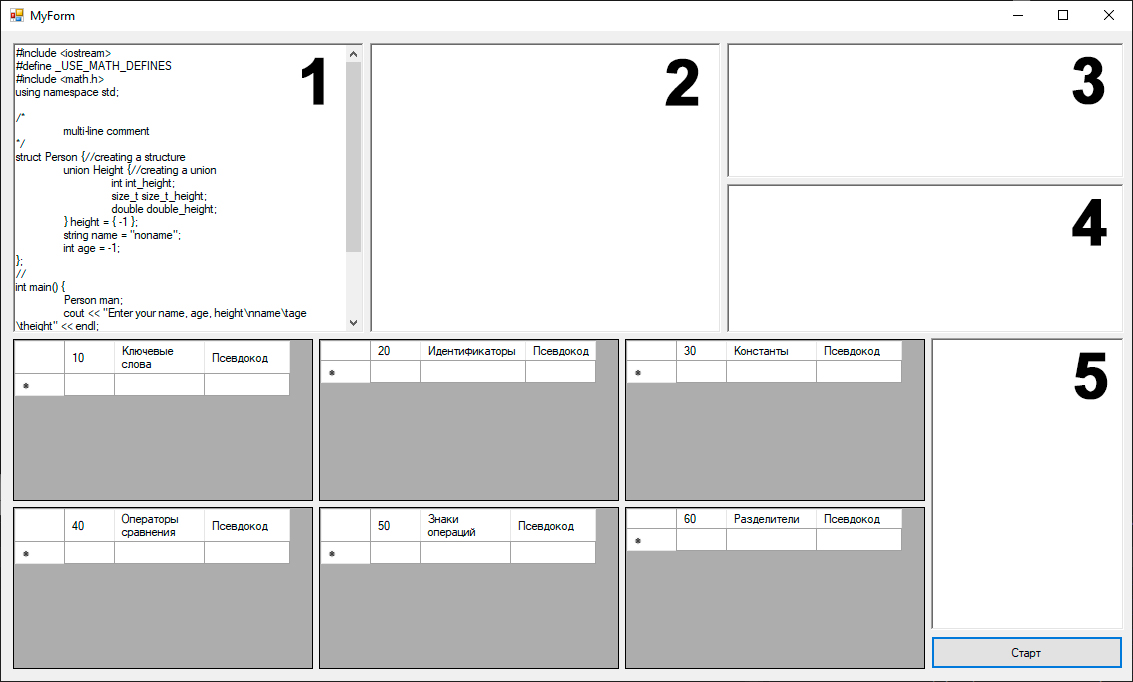


Рис. П2.1. Стартовое окно программы

Инструкции по работе

В окне 1 будет тестовый код для демонстрации работы программы, его можно заменить на любой другой код, написанный на языке C++. В окне 2 формируется код, очищенный от комментариев, лишних пробелов пустых строк. В окне 3 из очищенного кода формируется дескрипторный код, который преобразуется в псевдокод в окне 4. В случае ошибок, найденных в коде в процессе лексического анализа, выводятся соответствующие сообщения с указанием места и типа ошибки в окне 5. Таблицы под номерами 10, 20, 30, 40, 50, 60 — это таблицы ключевых слов, идентификаторов, констант, операторов сравнения, знаков операций и разделителей соответственно, которые будут найдены программой.

Для запуска лексического анализа нажмите на кнопку «Старт». По завершению работы будут сформированы таблицы лексем и заполнены окна 2, 3, 4 и 5 (в случае наличия ошибок). Пример работы представлен на рис. П2.2.

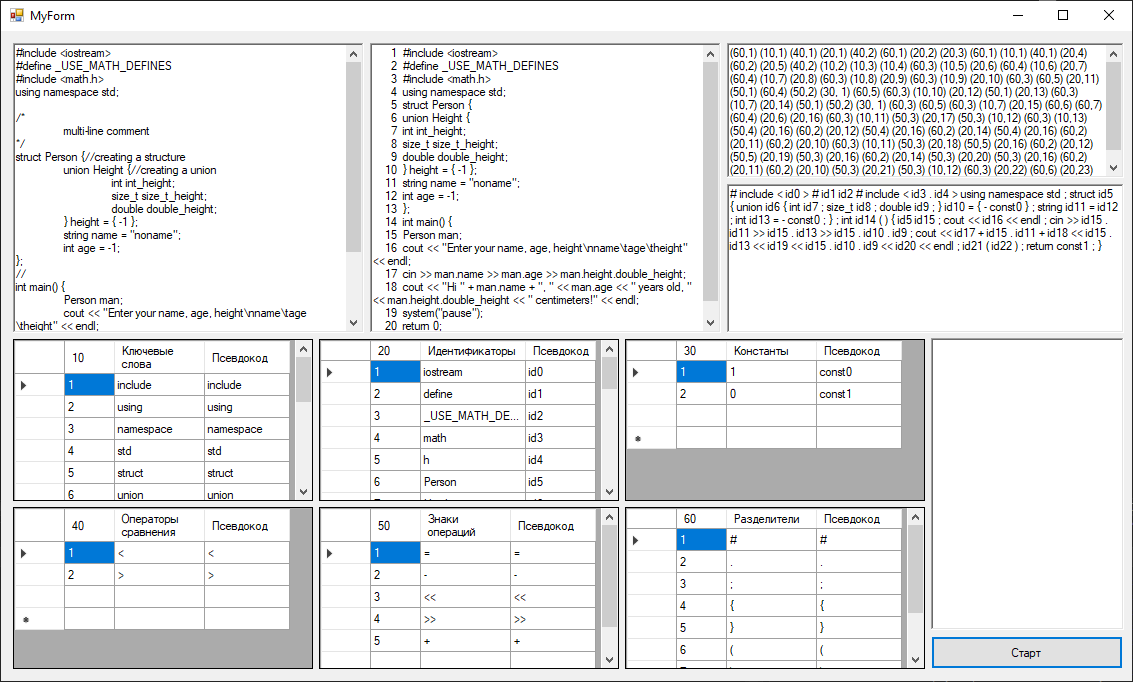


Рис. П2. 2. Пример работы программы

Сообщения пользователю

Отсутствуют

# Приложение 3

Текст программы.

Файл «MyForm.h»

#pragma once

namespace TA2coursework {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

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

{

public:

MyForm(void)

{

InitializeComponent();

}

protected:

~MyForm()

{

if (components)

{

delete components;

}

}

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

private: System::Windows::Forms::DataGridView^ table\_id;

private: System::Windows::Forms::DataGridView^ table\_lexeme;

private: System::Windows::Forms::DataGridView^ table\_const;

private: System::Windows::Forms::DataGridView^ table\_comparison\_operators;

private: System::Windows::Forms::DataGridView^ table\_operation\_sings;

private: System::Windows::Forms::DataGridView^ table\_separators;

private: System::Windows::Forms::RichTextBox^ text\_out;

private: System::Windows::Forms::RichTextBox^ descriptor\_code;

private: System::Windows::Forms::RichTextBox^ pre\_code;

private: System::Windows::Forms::RichTextBox^ list\_of\_erros;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column10;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column11;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column12;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column20;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column21;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column22;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column30;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column31;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column32;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column40;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column41;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column42;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column50;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column51;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column52;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column60;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column61;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column62;

private: System::Windows::Forms::RichTextBox^ text\_in;

private:

//поиск идентификаторов

bool search\_id(String^\*, int\*);

//добавление идентификаторов

void add\_id(String^\*, int\*, int\*);

//поиск лексем

bool search\_keyword(String^\*, int\*);

//добавлние лексем

void add\_keyword(String^\*, int\*);

//поиск констант

bool search\_const(String^\*, int\*);

//добавление констант

void add\_const(String^\*, int\*);

//поиск операторов сравнения

bool search\_comparison\_operators(String^\*, int\*);

//добавление операторов сравнения

void add\_comparison\_operators(String^\*, int\*);

//поиск знаков операций

bool search\_operation\_sings(String^\*, int\*);

//добавление знаков операций

void add\_operation\_sings(String^\*, int\*);

//поиск разделителей

bool search\_separators(String^\*, int\*);

//добавление разделителей

void add\_separators(String^\*, int\*);

//поиск скачков

bool skip(wchar\_t\*);

//поиск оператора сравнения

bool comparison\_operators(wchar\_t\*);

//поиск знаков операций

bool operation\_sings(wchar\_t\*);

//спец символы

bool separators(wchar\_t\*);

//другое

bool other(wchar\_t\*);

//поиск стоп-символа

bool end\_condition(wchar\_t\*);

//разрешённые элементы имён переменных

bool allowed\_character(wchar\_t\*);

//числа, не включая 0

bool numbers(wchar\_t\*);

//вывод сообщения об ошибке

void error(String^\*, String^, int\*);

private:

System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code

void InitializeComponent(void)

{

System::ComponentModel::ComponentResourceManager^ resources = (gcnew System::ComponentModel::ComponentResourceManager(MyForm::typeid));

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

this->table\_id = (gcnew System::Windows::Forms::DataGridView());

this->Column20 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->Column21 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->Column22 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->table\_lexeme = (gcnew System::Windows::Forms::DataGridView());

this->Column10 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->Column11 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->Column12 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->table\_const = (gcnew System::Windows::Forms::DataGridView());

this->Column30 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->Column31 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->Column32 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->table\_comparison\_operators = (gcnew System::Windows::Forms::DataGridView());

this->Column40 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->Column41 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->Column42 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->table\_operation\_sings = (gcnew System::Windows::Forms::DataGridView());

this->Column50 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->Column51 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->Column52 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->table\_separators = (gcnew System::Windows::Forms::DataGridView());

this->Column60 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->Column61 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->Column62 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->text\_out = (gcnew System::Windows::Forms::RichTextBox());

this->text\_in = (gcnew System::Windows::Forms::RichTextBox());

this->descriptor\_code = (gcnew System::Windows::Forms::RichTextBox());

this->pre\_code = (gcnew System::Windows::Forms::RichTextBox());

this->list\_of\_erros = (gcnew System::Windows::Forms::RichTextBox());

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

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

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

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

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

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

this->SuspendLayout();

//

// run

//

this->run->Location = System::Drawing::Point(930, 605);

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

this->run->Size = System::Drawing::Size(192, 33);

this->run->TabIndex = 2;

this->run->Text = L"Старт";

this->run->UseVisualStyleBackColor = true;

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

//

// table\_id

//

this->table\_id->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

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

this->Column20, this->Column21,

this->Column22

});

this->table\_id->Location = System::Drawing::Point(318, 308);

this->table\_id->Name = L"table\_id";

this->table\_id->RowHeadersWidth = 51;

this->table\_id->Size = System::Drawing::Size(300, 162);

this->table\_id->TabIndex = 4;

//

// Column20

//

this->Column20->HeaderText = L"20";

this->Column20->MinimumWidth = 6;

this->Column20->Name = L"Column20";

this->Column20->Width = 50;

//

// Column21

//

this->Column21->HeaderText = L"Идентификаторы";

this->Column21->MinimumWidth = 6;

this->Column21->Name = L"Column21";

this->Column21->Width = 105;

//

// Column22

//

this->Column22->HeaderText = L"Псевдокод";

this->Column22->MinimumWidth = 6;

this->Column22->Name = L"Column22";

this->Column22->Width = 70;

//

// table\_lexeme

//

this->table\_lexeme->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

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

this->Column10,

this->Column11, this->Column12

});

this->table\_lexeme->Location = System::Drawing::Point(12, 308);

this->table\_lexeme->Name = L"table\_lexeme";

this->table\_lexeme->RowHeadersWidth = 51;

this->table\_lexeme->Size = System::Drawing::Size(300, 162);

this->table\_lexeme->TabIndex = 5;

//

// Column10

//

this->Column10->HeaderText = L"10";

this->Column10->MinimumWidth = 6;

this->Column10->Name = L"Column10";

this->Column10->Width = 50;

//

// Column11

//

this->Column11->HeaderText = L"Лексемы";

this->Column11->MinimumWidth = 6;

this->Column11->Name = L"Column11";

this->Column11->Width = 90;

//

// Column12

//

this->Column12->HeaderText = L"Псевдокод";

this->Column12->MinimumWidth = 6;

this->Column12->Name = L"Column12";

this->Column12->Width = 85;

//

// table\_const

//

this->table\_const->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

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

this->Column30,

this->Column31, this->Column32

});

this->table\_const->Location = System::Drawing::Point(624, 308);

this->table\_const->Name = L"table\_const";

this->table\_const->RowHeadersWidth = 51;

this->table\_const->Size = System::Drawing::Size(300, 162);

this->table\_const->TabIndex = 6;

//

// Column30

//

this->Column30->HeaderText = L"30";

this->Column30->MinimumWidth = 6;

this->Column30->Name = L"Column30";

this->Column30->Width = 50;

//

// Column31

//

this->Column31->HeaderText = L"Константы";

this->Column31->MinimumWidth = 6;

this->Column31->Name = L"Column31";

this->Column31->Width = 90;

//

// Column32

//

this->Column32->HeaderText = L"Псевдокод";

this->Column32->MinimumWidth = 6;

this->Column32->Name = L"Column32";

this->Column32->Width = 85;

//

// table\_comparison\_operators

//

this->table\_comparison\_operators->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->table\_comparison\_operators->Columns->AddRange(gcnew cli::array< System::Windows::Forms::DataGridViewColumn^ >(3) {

this->Column40,

this->Column41, this->Column42

});

this->table\_comparison\_operators->Location = System::Drawing::Point(12, 476);

this->table\_comparison\_operators->Name = L"table\_comparison\_operators";

this->table\_comparison\_operators->RowHeadersWidth = 51;

this->table\_comparison\_operators->Size = System::Drawing::Size(300, 162);

this->table\_comparison\_operators->TabIndex = 7;

//

// Column40

//

this->Column40->HeaderText = L"40";

this->Column40->MinimumWidth = 6;

this->Column40->Name = L"Column40";

this->Column40->Width = 50;

//

// Column41

//

this->Column41->HeaderText = L"Операторы сравнения";

this->Column41->MinimumWidth = 6;

this->Column41->Name = L"Column41";

this->Column41->Resizable = System::Windows::Forms::DataGridViewTriState::False;

this->Column41->Width = 90;

//

// Column42

//

this->Column42->HeaderText = L"Псевдокод";

this->Column42->MinimumWidth = 6;

this->Column42->Name = L"Column42";

this->Column42->Width = 85;

//

// table\_operation\_sings

//

this->table\_operation\_sings->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->table\_operation\_sings->Columns->AddRange(gcnew cli::array< System::Windows::Forms::DataGridViewColumn^ >(3) {

this->Column50,

this->Column51, this->Column52

});

this->table\_operation\_sings->Location = System::Drawing::Point(318, 476);

this->table\_operation\_sings->Name = L"table\_operation\_sings";

this->table\_operation\_sings->RowHeadersWidth = 51;

this->table\_operation\_sings->Size = System::Drawing::Size(300, 162);

this->table\_operation\_sings->TabIndex = 8;

//

// Column50

//

this->Column50->HeaderText = L"50";

this->Column50->MinimumWidth = 6;

this->Column50->Name = L"Column50";

this->Column50->Width = 50;

//

// Column51

//

this->Column51->HeaderText = L"Знаки операций";

this->Column51->MinimumWidth = 6;

this->Column51->Name = L"Column51";

this->Column51->Width = 90;

//

// Column52

//

this->Column52->HeaderText = L"Псевдокод";

this->Column52->MinimumWidth = 6;

this->Column52->Name = L"Column52";

this->Column52->Width = 85;

//

// table\_separators

//

this->table\_separators->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

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

this->Column60,

this->Column61, this->Column62

});

this->table\_separators->Location = System::Drawing::Point(624, 476);

this->table\_separators->Name = L"table\_separators";

this->table\_separators->RowHeadersWidth = 51;

this->table\_separators->Size = System::Drawing::Size(300, 162);

this->table\_separators->TabIndex = 9;

//

// Column60

//

this->Column60->HeaderText = L"60";

this->Column60->MinimumWidth = 6;

this->Column60->Name = L"Column60";

this->Column60->Width = 50;

//

// Column61

//

this->Column61->HeaderText = L"Разделители";

this->Column61->MinimumWidth = 6;

this->Column61->Name = L"Column61";

this->Column61->Width = 90;

//

// Column62

//

this->Column62->HeaderText = L"Псевдокод";

this->Column62->MinimumWidth = 6;

this->Column62->Name = L"Column62";

this->Column62->Width = 85;

//

// text\_out

//

this->text\_out->Location = System::Drawing::Point(369, 12);

this->text\_out->Name = L"text\_out";

this->text\_out->Size = System::Drawing::Size(351, 290);

this->text\_out->TabIndex = 13;

this->text\_out->Text = L"";

//

// text\_in

//

this->text\_in->Location = System::Drawing::Point(12, 12);

this->text\_in->Name = L"text\_in";

this->text\_in->Size = System::Drawing::Size(351, 290);

this->text\_in->TabIndex = 14;

this->text\_in->Text = resources->GetString(L"text\_in.Text");

//

// descriptor\_code

//

this->descriptor\_code->Location = System::Drawing::Point(726, 12);

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

this->descriptor\_code->Size = System::Drawing::Size(397, 135);

this->descriptor\_code->TabIndex = 15;

this->descriptor\_code->Text = L"";

//

// pre\_code

//

this->pre\_code->Location = System::Drawing::Point(726, 153);

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

this->pre\_code->Size = System::Drawing::Size(397, 149);

this->pre\_code->TabIndex = 16;

this->pre\_code->Text = L"";

//

// list\_of\_erros

//

this->list\_of\_erros->Location = System::Drawing::Point(930, 307);

this->list\_of\_erros->Name = L"list\_of\_erros";

this->list\_of\_erros->Size = System::Drawing::Size(193, 292);

this->list\_of\_erros->TabIndex = 17;

this->list\_of\_erros->Text = L"";

//

// MyForm

//

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

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

this->ClientSize = System::Drawing::Size(1131, 650);

this->Controls->Add(this->list\_of\_erros);

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

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

this->Controls->Add(this->text\_in);

this->Controls->Add(this->text\_out);

this->Controls->Add(this->table\_separators);

this->Controls->Add(this->table\_operation\_sings);

this->Controls->Add(this->table\_comparison\_operators);

this->Controls->Add(this->table\_const);

this->Controls->Add(this->table\_lexeme);

this->Controls->Add(this->table\_id);

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

this->Name = L"MyForm";

this->StartPosition = System::Windows::Forms::FormStartPosition::CenterScreen;

this->Text = L"MyForm";

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

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

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

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

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

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

this->ResumeLayout(false);

}

#pragma endregion

//удаление пробелов и пустых строк

private: void cleaning(String^\*);

//чтение лексем, идентификаторов, констант, операторов сравнения, знаков операций, разделителей

private: void processing(int\*, int\*, int\*, int\*, int\*, int\*);

//формирование дескрипторного кода

private: void code\_descriptor(int\*, int\*, int\*, int\*, int\*, int\*);

//формирование псевдокода

private: void code\_pre(int\*, int\*, int\*, int\*, int\*, int\*);

//старт

private: System::Void run\_Click(System::Object^ sender, System::EventArgs^ e);

};

}

Файл «MyForm.cpp»

#include "MyForm.h"

using namespace System::IO;

using namespace System;

using namespace System::Windows::Forms;

[STAThreadAttribute]

int main(array<String^>^ args)

{

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

TA2coursework::MyForm form;

Application::Run(% form);

}

bool TA2coursework::MyForm::search\_id(String^\* word, int\* row)

{

for (size\_t i = 0; i <= \*row; i++)

if (Convert::ToString(table\_id->Rows[i]->Cells[1]->Value) == \*word)

return true;

return false;

}

void TA2coursework::MyForm::add\_id(String^\* word, int\* row\_id, int\* row\_keyword)

{

if (!search\_id(word, row\_id) && !search\_keyword(word, row\_keyword)) {

table\_id->Rows[\*row\_id - 1]->Cells[0]->Value = Convert::ToString(\*row\_id);

table\_id->Rows[\*row\_id - 1]->Cells[1]->Value = \*word;

table\_id->Rows[\*row\_id - 1]->Cells[2]->Value = "id" + Convert::ToString(\*row\_id - 1);

table\_id->RowCount = ++ \* row\_id + 1;

}

\*word = "";

}

bool TA2coursework::MyForm::search\_keyword(String^\* word, int\* row)

{

for (size\_t i = 0; i <= \*row; i++)

if (Convert::ToString(table\_lexeme->Rows[i]->Cells[1]->Value) == \*word)

return true;

return false;

}

void TA2coursework::MyForm::add\_keyword(String^\* word, int\* row)

{

if (!search\_keyword(word, row)) {

table\_lexeme->Rows[\*row - 1]->Cells[0]->Value = Convert::ToString(\*row);

table\_lexeme->Rows[\*row - 1]->Cells[1]->Value = \*word;

table\_lexeme->Rows[\*row - 1]->Cells[2]->Value = \*word;

table\_lexeme->RowCount = ++ \* row + 1;

}

\*word = "";

}

bool TA2coursework::MyForm::search\_const(String^\* word, int\* row)

{

for (size\_t i = 0; i <= \*row; i++)

if (Convert::ToString(table\_const->Rows[i]->Cells[1]->Value) == \*word)

return true;

return false;

}

void TA2coursework::MyForm::add\_const(String^\* word, int\* row)

{

if (!search\_const(word, row)) {

table\_const->Rows[\*row - 1]->Cells[0]->Value = Convert::ToString(\*row);

table\_const->Rows[\*row - 1]->Cells[1]->Value = \*word;

table\_const->Rows[\*row - 1]->Cells[2]->Value = "const" + Convert::ToString(\*row - 1);

table\_const->RowCount = ++ \* row + 1;

}

\*word = "";

}

bool TA2coursework::MyForm::search\_comparison\_operators(String^\* word, int\* row)

{

for (size\_t i = 0; i <= \*row; i++)

if (Convert::ToString(table\_comparison\_operators->Rows[i]->Cells[1]->Value) == \*word)

return true;

return false;

}

void TA2coursework::MyForm::add\_comparison\_operators(String^\* word, int\* row)

{

if (!search\_comparison\_operators(word, row)) {

table\_comparison\_operators->Rows[\*row - 1]->Cells[0]->Value = Convert::ToString(\*row);

table\_comparison\_operators->Rows[\*row - 1]->Cells[1]->Value = \*word;

table\_comparison\_operators->Rows[\*row - 1]->Cells[2]->Value = \*word;

table\_comparison\_operators->RowCount = ++ \* row + 1;

}

\*word = "";

}

bool TA2coursework::MyForm::search\_operation\_sings(String^\* word, int\* row)

{

for (size\_t i = 0; i <= \*row; i++)

if (Convert::ToString(table\_operation\_sings->Rows[i]->Cells[1]->Value) == \*word)

return true;

return false;

}

void TA2coursework::MyForm::add\_operation\_sings(String^\* word, int\* row)

{

if (!search\_operation\_sings(word, row)) {

table\_operation\_sings->Rows[\*row - 1]->Cells[0]->Value = Convert::ToString(\*row);

table\_operation\_sings->Rows[\*row - 1]->Cells[1]->Value = \*word;

table\_operation\_sings->Rows[\*row - 1]->Cells[2]->Value = \*word;

table\_operation\_sings->RowCount = ++ \* row + 1;

}

\*word = "";

}

bool TA2coursework::MyForm::search\_separators(String^\* word, int\* row)

{

for (size\_t i = 0; i <= \*row; i++)

if (Convert::ToString(table\_separators->Rows[i]->Cells[1]->Value) == \*word)

return true;

return false;

}

void TA2coursework::MyForm::add\_separators(String^\* word, int\* row)

{

if (!search\_separators(word, row)) {

table\_separators->Rows[\*row - 1]->Cells[0]->Value = Convert::ToString(\*row);

table\_separators->Rows[\*row - 1]->Cells[1]->Value = \*word;

table\_separators->Rows[\*row - 1]->Cells[2]->Value = \*word;

table\_separators->RowCount = ++ \* row + 1;

}

\*word = "";

}

bool TA2coursework::MyForm::skip(wchar\_t\* c)

{

return (\*c == ' ' || \*c == '\n' || \*c == '\r') ? true : false;

}

bool TA2coursework::MyForm::comparison\_operators(wchar\_t\* c)

{

return (\*c == '<' || \*c == '>' || \*c == '=' || \*c == '!') ? true : false;

}

bool TA2coursework::MyForm::operation\_sings(wchar\_t\* c)

{

return (\*c == '=' || \*c == '-' || \*c == '+' || \*c == '\*' || \*c == '/'

|| \*c == '%' || \*c == '|' || \*c == '&') ? true : false;

}

bool TA2coursework::MyForm::separators(wchar\_t\* c)

{

return (\*c == ';' || \*c == '(' || \*c == ')' || \*c == '{' || \*c == '}' || \*c == ':'

|| \*c == '#' || \*c == ',' || \*c == '[' || \*c == ']') ? true : false;

}

bool TA2coursework::MyForm::other(wchar\_t\* c)

{

return (\*c == '.') ? true : false;

}

bool TA2coursework::MyForm::end\_condition(wchar\_t\* c)

{

return (other(c) || separators(c) || comparison\_operators(c) || operation\_sings(c) || skip(c)) ? true : false;

}

bool TA2coursework::MyForm::allowed\_character(wchar\_t\* c)

{

return (\*c == '\_' || \*c == 'a' || \*c == 'b' || \*c == 'c' || \*c == 'd' || \*c == 'e' || \*c == 'f'

|| \*c == 'g' || \*c == 'h' || \*c == 'i' || \*c == 'j' || \*c == 'k' || \*c == 'l' || \*c == 'm'

|| \*c == 'n' || \*c == 'o' || \*c == 'p' || \*c == 'q' || \*c == 'r' || \*c == 's' || \*c == 't'

|| \*c == 'u' || \*c == 'v' || \*c == 'w' || \*c == 'x' || \*c == 'y' || \*c == 'z' || \*c == 'A'

|| \*c == 'B' || \*c == 'C' || \*c == 'D' || \*c == 'E' || \*c == 'F' || \*c == 'G' || \*c == 'H'

|| \*c == 'I' || \*c == 'J' || \*c == 'K' || \*c == 'L' || \*c == 'M' || \*c == 'N' || \*c == 'O'

|| \*c == 'P' || \*c == 'Q' || \*c == 'R' || \*c == 'S' || \*c == 'T' || \*c == 'U' || \*c == 'V'

|| \*c == 'W' || \*c == 'X' || \*c == 'Y' || \*c == 'Z') ? true : false;

}

bool TA2coursework::MyForm::numbers(wchar\_t\* c)

{

return (\*c == '1' || \*c == '2' || \*c == '3' || \*c == '4' || \*c == '5'

|| \*c == '6' || \*c == '7' || \*c == '8' || \*c == '9') ? true : false;

}

void TA2coursework::MyForm::error(String^\* word, String^ type, int\* line)

{

if (type == "const") type = "константы";

else if (type == "id") type = "идентефикатора";

else if (type == "multi-line\_comment") {

type = "многострочного комментария";

\*word = "Не найден закрывающий \*/";

}

list\_of\_erros->Text += "Строка " + (\*line + 1) + ": " + \*word + " – ошибка " + type + "\n";

\*word = "";

}

void TA2coursework::MyForm::cleaning(String^\* str)

{

int S = 0, n = 0, line = 0;

StreamWriter^ fsave = gcnew StreamWriter("buffer.txt", false);

StreamWriter^ ftext = gcnew StreamWriter("text\_out.txt", false);

ftext->Write(" " + System::Convert::ToString(++n) + " ");

String^ a = "";

for (int i = 0; i < (\*str)->Length; i++)

{

a = System::Convert::ToString((\*str)[i]);

switch (S) {

// состояние пустой строки

case 0:

if (a == " " || a == "\t" || a == "\n" || a == "\r") S = 0;

else if (a == "/") S = 9;

else S = 1;

break;

// состояние записи итогового кода

case 1:

if (a == " " || a == "\t") S = 2;

else if (a == "/") S = 3;

else if (a == "\n" || a == "\r") S = 8;

break;

// состояние нахождения пробелов

case 2:

if (a == " " || a == "\t") S = 14;

else if (a == "/") S = 3;

else if (a == "\n" || a == "\r") S = 8;

else S = 1;

break;

// определение типа комментария

case 3:

if (a == "/") S = 4;

else if (a == "\*") S = 5;

else if (a == "\n" || a == "\r") S = 8;

else {

a = "/" + a;

S = 1;

}

break;

// чтение однострочного комментария

case 4:

if (a == "\n" || a == "\r") S = 8;

break;

// чтение многострочного комментария

case 5:

if (a == "\*") S = 6;

break;

case 6:

if (a == "/") S = 7;

else S = 5;

break;

case 7:

if (a == "\*") S = 6;

else if (a == "\t" || a == " ") S = 2;

else if (a == "/") S = 3;

else if (a == "\n" || a == "\r") S = 0;

else S = 1;

break;

// состояние переноса строки

case 8:

S = 0;

break;

// комментарий в начале строки

case 9:

if (a == "/") S = 10;

else if (a == "\*") S = 11;

else if (a == "\n" || a == "\r") S = 0;

else {

a = "/" + a;

S = 1;

}

break;

case 10:

if (a == "\n" || a == "\r") S = 0;

break;

case 11:

if (a == "\*") S = 12;

break;

case 12:

if (a == "/") S = 13;

else S = 11;

break;

case 13:

if (a == " " || a == "\t" || a == "\n" || a == "\r") S = 0;

else if (a == "\*") S = 12;

else if (a == "/") S = 9;

else S = 1;

break;

// игнорирование пробелов

case 14:

if (a == "\n" || a == "\r") S = 8;

else if (a == "/") S = 3;

else if (a == " " || a == "\t") S = 14;

else S = 1;

break;

default: MessageBox::Show(L"Ошибка при удалении пустот!", L"Программа остановлена", MessageBoxButtons::OK, MessageBoxIcon::Error); return;

}

// заполнение таблицы

if (S == 8) {

if (n < 9) ftext ->Write("\n " + ++n + " ");

else if (n < 99) ftext ->Write("\n " + ++n + " ");

else if (n < 999) ftext ->Write("\n " + ++n + " ");

else ftext ->Write("\n" + ++n + " ");

fsave->Write("\n");

line++;

i--;

}

// запись символа

else if (S == 1) ftext ->Write(a), fsave->Write(a);

else if (S == 2) ftext ->Write(" "), fsave->Write(" ");

}

if (S == 11 || S == 5)

error(&a, "multi-line\_comment", &line);

fsave->Write(" ");

ftext ->Close();

fsave->Close();

StreamReader^ fout = gcnew StreamReader("text\_out.txt");

text\_out->Text = fout ->ReadToEnd();

ftext->Close();

}

void TA2coursework::MyForm::processing(int\* row\_keyword, int\* row\_id, int\* row\_const, int\* row\_comparison\_operators, int\* row\_operation\_sings, int\* row\_separators)

{

int S = 0, last\_s, line = 0;

StreamReader^ fout = gcnew StreamReader("buffer.txt");

String^ str = fout->ReadToEnd();

fout->Close();

String^ word = "";

wchar\_t c;

if (str != "") {

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

last\_s = S;

c = str[i];

switch (S) {

//0 состояние, запись идентификаторов

case 0:

if (skip(&c)) S = 0;

else if (c == '"') S = 4;//строка

else if (c == 'a') S = 8;//auto, and

else if (c == 'l') S = 11;//long

else if (c == 'b') S = 14;//bool, break

else if (c == 'u') S = 21;//using, union, unsigned

else if (c == 'c') S = 22;//case, catch, char, class, const, continue, cin, cout

else if (c == 'x') S = 28;//xor

else if (c == 'm') S = 31;//mutable

else if (c == 'd') S = 44;//default, delete, do, double

else if (c == 'e') S = 49;//else, enum, export, endl

else if (c == 'f') S = 54;//false, float, for, friend

else if (c == 'i') S = 58;//int, if, inline, include

else if (c == 'n') S = 62;//namespace, new, not, nullptr

else if (c == 'o') S = 71;//operator, or

else if (c == 'p') S = 76;//private, protected, public

else if (c == 's') S = 91;//short, sizeof, static, switch, string, struct, size\_t, std

else if (c == 'v') S = 103;//virtual, void, volatile

else if (c == 'r') S = 109;//return

else if (c == 't') S = 114;//template, this, throw, true, try, typename, typeid

else if (c == 'w') S = 162;//wchar\_t

else if (numbers(&c)) S = 128;//константа

else if (c == '0') S = 132;//0-константа

else if (c == '.' || c == ',' || c == '{' || c == '}' || c == '['

|| c == ']' || c == '(' || c == ')' || c == ';' || c == '#') S = 135;// . , { } [ ] ( ) : ; #

else if (c == ':') S = 136;//::

else if (c == '?') S = 136;//?

else if (c == '=') S = 141;//==

else if (c == '<') S = 142;//< <= <<

else if (c == '>') S = 143;//> >= >>

else if (c == '+') S = 144;//+ ++ +=

else if (c == '-') S = 145;//- -- -=

else if (c == '\*') S = 146;//\* \*=

else if (c == '/') S = 147;// / /=

else if (c == '%') S = 148;// % %=

else if (c == '|') S = 149;// | ||

else if (c == '&') S = 150;// & &&

else if (c == '!') S = 154;// ! !=

else if (allowed\_character(&c)) S = 1;//любой другой разрешённый символ

else S = 155;//ошибочный символ

break;

//чтение идентификаторов (прочее)

case 1:

if (numbers(&c) || c == '0') S = 1;

else if (allowed\_character(&c)) S = 1;

else if (end\_condition(&c)) S = 0;

else S = 155;

break;

//запись лексем

case 2:

S = 0;

break;

//проверка лексемы на пробел

case 3:

if (end\_condition(&c)) S = 2;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

//начало строки

case 4:

if (c == '"') S = 0;

else S = 7;

break;

//запись констант

case 5:

S = 0;

break;

//ошибочное состояние для констант

case 6:

S = 0;

break;

//чтение строки

case 7:

if (c == '"') S = 153;

break;

//чтение лексем

case 8:

if (c == 'u') S = 9;//auto

else if (c == 'n') S = 12;//and, friend

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 9:

if (c == 't') S = 10;//auto

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 10:

if (c == 'o') S = 3;//найдено auto, do

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 11:

if (c == 'o') S = 13;//long

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 12:

if (c == 'd') S = 3;//найдено and, friend, protected, void, unsigned, typeid

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 13:

if (c == 'n') S = 17;//long, using

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 14:

if (c == 'o') S = 15;//bool

else if (c == 'r') S = 18;//break

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 15:

if (c == 'o') S = 16;//bool

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 16:

if (c == 'l') S = 3;//найдено bool, virtual, endl

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 17:

if (c == 'g') S = 3;//найдено long, using, string

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 18:

if (c == 'e') S = 19;//break

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 19:

if (c == 'a') S = 20;//break

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 20:

if (c == 'k') S = 3;//найдено break

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 21:

if (c == 's') S = 25;//using

else if (c == 'n') S = 169;//union, unsigned

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 22:

if (c == 'a') S = 23;//case, catch

else if (c == 'h') S = 29;//char

else if (c == 'l') S = 32;//class

else if (c == 'o') S = 36;//const, continue, cout

else if (c == 'i') S = 113;//cin

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 23:

if (c == 's') S = 24;//case, else

else if (c == 't') S = 26;//catch

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 24:

if (c == 'e') S = 3;//найдено case, else, mutable, false, inline, include, continue, namespace, volatile

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 25:

if (c == 'i') S = 13;//using

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 26:

if (c == 'c') S = 27;//catch

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 27:

if (c == 'h') S = 3;//найдено catch, switch

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 28:

if (c == 'o') S = 30;//xor, operator

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 29:

if (c == 'a') S = 30;//char

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 30:

if (c == 'r') S = 3;//найдено char, xor, for, operator, nullptr

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 31:

if (c == 'u') S = 35;//mutable

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 32:

if (c == 'a') S = 33;//class

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 33:

if (c == 's') S = 43;//class

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 34:

if (c == 's') S = 3;//найдено this

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 35:

if (c == 't') S = 39;//mutable

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 36:

if (c == 'n') S = 37;//const, continue

else if (c == 'u') S = 38;//cout

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 37:

if (c == 's') S = 38;//const

else if (c == 't') S = 40;//continue

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 38:

if (c == 't') S = 3;//найдено const, default, float, not, int, short, export, size\_t, cout, wchar\_t

else if (c == 'l') S = 60;//inline

else if (c == 'c') S = 157;//include

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 39:

if (c == 'a') S = 52;//mutable

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 40:

if (c == 'i') S = 41;//continue

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 41:

if (c == 'n') S = 42;//continue

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 42:

if (c == 'u') S = 24;//continue

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 43:

if (c == 's') S = 3;//найдено class

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 44:

if (c == 'e') S = 45;//default, delete

else if (c == 'o') S = 160;//do, double

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 45:

if (c == 'f') S = 46;//default

else if (c == 'l') S = 81;//delete

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 46:

if (c == 'a') S = 47;//default

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 47:

if (c == 'u') S = 48;//default

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 48:

if (c == 'l') S = 38;//default

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 49:

if (c == 'l') S = 23;//else

else if (c == 'n') S = 50;//enum, endl

else if (c == 'x') S = 93;//export

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 50:

if (c == 'u') S = 51;//enum

else if (c == 'd') S = 16;//endl

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 51:

if (c == 'm') S = 3;//найдено enum

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 52:

if (c == 'b') S = 53;//mutable

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 53:

if (c == 'l') S = 24;//mutable

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 54:

if (c == 'a') S = 55;//false

else if (c == 'l') S = 56;//float

else if (c == 'o') S = 30;//for

else if (c == 'r') S = 87;//friend

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 55:

if (c == 'l') S = 23;//false

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 56:

if (c == 'o') S = 57;//float

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 57:

if (c == 'a') S = 38;//float

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 58:

if (c == 'n') S = 38;//int, inline, include

else if (c == 'f') S = 3;//if, sizeof

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 59:

if (c == '-' || c == '+') S = 69;//экспонента

else S = 130;

break;

case 60:

if (c == 'i') S = 61;//inline

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 61:

if (c == 'n') S = 24;//inline

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 62:

if (c == 'a') S = 63;//namespace

else if (c == 'e') S = 70;//new

else if (c == 'o') S = 38;//not

else if (c == 'u') S = 177;//nullptr

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 63:

if (c == 'm') S = 64;//namespace

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 64:

if (c == 'e') S = 65;//namespace

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 65:

if (c == 's') S = 66;//namespace

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 66:

if (c == 'p') S = 67;//namespace

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 67:

if (c == 'a') S = 68;//namespace

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 68:

if (c == 'c') S = 24;//namespace

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 69:

if (!end\_condition(&c)) S = 184;//вводится экспонента

else S = 130;

break;

case 70:

if (c == 'w') S = 3;//найдено new

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 71:

if (c == 'p') S = 72;//operator

else if (c == 'r') S = 3;//найдено or

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 72:

if (c == 'e') S = 73;//operator

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 73:

if (c == 'r') S = 74;//operator

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 74:

if (c == 'a') S = 75;//operator

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 75:

if (c == 't') S = 28;//operator

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 76:

if (c == 'r') S = 77;//private, protected

else if (c == 'u') S = 83;//public

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 77:

if (c == 'i') S = 78;//private

else if (c == 'o') S = 89;//protected

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 78:

if (c == 'v') S = 79;//private

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 79:

if (c == 'a') S = 80;//private

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 80:

if (c == 't') S = 82;//private, delete

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 81:

if (c == 'e') S = 80;//delete

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 82:

if (c == 'e') S = 3;//найдено delete, private, template, typename

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 83:

if (c == 'b') S = 84;//public

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 84:

if (c == 'l') S = 85;//public

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 85:

if (c == 'i') S = 86;//public

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 86:

if (c == 'c') S = 3;//найдено public, static

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 87:

if (c == 'i') S = 88;//friend

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 88:

if (c == 'e') S = 8;//friend

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 89:

if (c == 't') S = 90;//protected

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 90:

if (c == 'e') S = 166;//protected

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 91:

if (c == 'h') S = 92;//short

else if (c == 'w') S = 95;//switch

else if (c == 'i') S = 96;//sizeof, size\_t

else if (c == 't') S = 99;//static, string, struct, std

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 92:

if (c == 'o') S = 94;//short

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 93:

if (c == 'p') S = 92;//export

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 94:

if (c == 'r') S = 38;//export, short

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 95:

if (c == 'i') S = 23;//switch

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 96:

if (c == 'z') S = 97;//sizeof, size\_t

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 97:

if (c == 'e') S = 98;//sizeof, size\_t

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 98:

if (c == 'o') S = 58;//sizeof

else if (c == '\_') S = 38;//size\_t, wchar\_t

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 99:

if (c == 'a') S = 100;//static

else if (c == 'r') S = 101;//string, struct

else if (c == 'd') S = 3;//std

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 100:

if (c == 't') S = 85;//static

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 101:

if (c == 'i') S = 13;//string

else if (c == 'u') S = 102;//struct

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 102:

if (c == 'c') S = 126;//struct

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 103:

if (c == 'i') S = 105;//virtual

else if (c == 'o') S = 104;//void, volatile

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 104:

if (c == 'i') S = 12;//void

else if (c == 'l') S = 181;//volatile

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 105:

if (c == 'r') S = 106;//virtual

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 106:

if (c == 't') S = 107;//virtual

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 107:

if (c == 'u') S = 108;//virtual

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 108:

if (c == 'a') S = 16;//virtual

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 109:

if (c == 'e') S = 110;//return

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 110:

if (c == 't') S = 111;//return

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 111:

if (c == 'u') S = 112;//return

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 112:

if (c == 'r') S = 113;//return

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 113:

if (c == 'n') S = 3;//найдено return, cin

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 114:

if (c == 'e') S = 116;//template

else if (c == 'h') S = 119;//this, throw

else if (c == 'r') S = 115;//true, try

else if (c == 'y') S = 121;//typename

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 115:

if (c == 'u') S = 24;//true

else if (c == 'y') S = 3;//найдено try

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 116:

if (c == 'm') S = 117;//template

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 117:

if (c == 'p') S = 118;//template

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 118:

if (c == 'l') S = 79;//template

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 119:

if (c == 'i') S = 34;//this

else if (c == 'r') S = 120;//throw

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 120:

if (c == 'o') S = 70;//throw

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 121:

if (c == 'p') S = 122;//typename

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 122:

if (c == 'e') S = 123;//typename

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 123:

if (c == 'n') S = 124;//typename

else if (c == 'i') S = 12;//typeid

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 124:

if (c == 'a') S = 125;//typename

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 125:

if (c == 'm') S = 82;//typename

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

//запись структуры

case 126:

if (c == 't') S = 3;//struct

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

//целые константы

case 128:

if (separators(&c) || comparison\_operators(&c) || operation\_sings(&c) || skip(&c)) S = 5;//введена константа

else if (numbers(&c) || c == '0') S = 128;//вводится константа

else if (c == '.') S = 129;//вводится дробная константа

else if (c == 'e' || c == 'E') S = 59;//экспонента

else S = 130;

break;

case 129:

if (numbers(&c) || c == '0') S = 131;//вводится дробная константа

else S = 130;//ошибка

break;

//ошибка константы

case 130:

if (separators(&c) || comparison\_operators(&c) || operation\_sings(&c) || skip(&c)) S = 6;

break;

//вводится дробная константа

case 131:

if (end\_condition(&c)) S = 5;//введена константа

else if (numbers(&c) || c == '0') S = 131;//вводится дробная константа

else if (c == 'e' || c == 'E') S = 59;//экспонента

else S = 130;//ошибка

break;

//вводится 0-константа

case 132:

if (separators(&c) || comparison\_operators(&c) || operation\_sings(&c) || skip(&c)) S = 5;//введена 0-константа

else if (c == '.') S = 129;//вводится дробная 0-константа

else S = 130;

break;

//запись :: { } ( ) # . , ;

case 135:

S = 0;

break;

//разделители

//чтение ::

case 136:

if (c == ':') S = 135;

else S = 137;

break;

//запись :

case 137:

S = 0;

break;

//операторы сравнения

//чтение <= >=

case 138:

if (c == '=') S = 139;

else S = 140;

break;

//запись <= >= != == ? || &&

case 139:

S = 0;

break;

//запись < >

case 140:

S = 0;

break;

//чтение ==

case 141:

if (c == '=') S = 139;

else S = 151;

break;

//чтение < <= <<

case 142:

if (c == '=') S = 139;

else if (c == '<') S = 152;

else S = 140;

break;

//чтение > >= >>

case 143:

if (c == '=') S = 139;

else if (c == '>') S = 152;

else S = 140;

break;

//знаки операций

//чтение + ++ +=

case 144:

if (c == '+' || c == '=') S = 152;

else S = 151;

break;

//чтение- -- -=

case 145:

if (c == '-' || c == '=') S = 152;

else S = 151;

break;

//чтение \* \*=

case 146:

if (c == '=') S = 152;

else S = 151;

break;

//чтение / /=

case 147:

if (c == '=') S = 152;

else S = 151;

break;

//чтение % %=

case 148:

if (c == '=') S = 152;

else S = 151;

break;

//чтение | || |=

case 149:

if (c == '|') S = 139;

else if (c == '=') S = 152;

else S = 151;

break;

//чтение & && &=

case 150:

if (c == '&') S = 139;

else if (c == '=') S = 152;

else S = 151;

break;

//запись + - / \* = | &

case 151:

S = 0;

break;

//запись ++ += -- -= \* \*= / /= % %= |= &= >> <<

case 152:

S = 0;

break;

//запись строки

case 153:

S = 0;

break;

//чтение ! !=

case 154:

if (c == '=') S = 139;

else S = 151;

break;

//ошибка идентефикатора

//чтение слова

case 155:

if (end\_condition(&c)) S = 156;

break;

//запись слова

case 156:

S = 0;

break;

case 157:

if (c == 'l') S = 158;//include

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 158:

if (c == 'u') S = 159;//include

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 159:

if (c == 'd') S = 24;//include

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 160:

if (c == 'u') S = 161;//double

else if (end\_condition(&c)) S = 2;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 161:

if (c == 'b') S = 53;//double

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 162:

if (c == 'c') S = 163;//wchar\_t

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 163:

if (c == 'h') S = 164;//wchar\_t

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 164:

if (c == 'a') S = 165;//wchar\_t

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 165:

if (c == 'r') S = 98;//wchar\_t

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 166:

if (c == 'c') S = 167;//protected

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 167:

if (c == 't') S = 168;//protected

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 168:

if (c == 'e') S = 12;//protected, unsigned

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 169:

if (c == 'i') S = 170;//union

else if (c == 's') S = 174;//unsigned

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 170:

if (c == 'o') S = 171;//union

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 171:

if (c == 'n') S = 3;//union

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 174:

if (c == 'i') S = 175;//unsigned

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 175:

if (c == 'g') S = 176;//unsigned

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 176:

if (c == 'n') S = 168;//unsigned

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 177:

if (c == 'l') S = 178;//nullptr

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 178:

if (c == 'l') S = 179;//nullptr

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 179:

if (c == 'p') S = 180;//nullptr

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 180:

if (c == 't') S = 30;//nullptr

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 181:

if (c == 'a') S = 182;//volatile

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 182:

if (c == 't') S = 183;//volatile

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

case 183:

if (c == 'i') S = 53;//volatile

else if (end\_condition(&c)) S = 0;

else if (allowed\_character(&c)) S = 1; else S = 155;

break;

//вводится экспонента

case 184:

if (!(numbers(&c) || c == '0')) S = 5;

else if (end\_condition(&c)) S = 130;

break;

default: MessageBox::Show(L"Ошибка состояний!", L"Программа остановлена", MessageBoxButtons::OK, MessageBoxIcon::Error);

return;

}

if (c == '\n')

line++;

if (S != 0 && S != 2 && S != 5 && S != 6 && S != 133 && S != 137 && S != 140 && S != 151 && S != 156 && S != 173)

word += c;

if (S == 0 && word != "")//если нет этого слова в других категориях

add\_id(&word, row\_id, row\_keyword), last\_s == 1 ? i-- : 0;

else if (S == 2)

add\_keyword(&word, row\_keyword);

else if (S == 5)

add\_const(&word, row\_const);

else if ((S == 139 || S == 140))

add\_comparison\_operators(&word, row\_comparison\_operators);

else if ((S == 151 || S == 152))

add\_operation\_sings(&word, row\_operation\_sings);

else if ((S == 135 || S == 137))

add\_separators(&word, row\_separators);

else if (S == 6)

error(&word, "const", &line);

else if (S == 156)

error(&word, "id", &line);

else if (S == 173)

error(&word, "clstun", &line);

if (S == 6 || S == 135 || S == 139 || S == 152 || S == 153 || S == 156 || S == 173) i--;//предпросмотр для занесения слова без чтения символа

else if (S == 2 || S == 5 || S == 137 || S == 140 || S == 151) i -= 2;

}

}

else MessageBox::Show(L"Пустая последовательность!", L"Программа остановлена", MessageBoxButtons::OK, MessageBoxIcon::Error);

}

void TA2coursework::MyForm::code\_descriptor(int\* row\_lexeme, int\* row\_id, int\* row\_const, int\* row\_comparison\_operators, int\* row\_operation\_sings, int\* row\_separators)

{

StreamReader^ fout = gcnew StreamReader("buffer.txt");

String^ str = fout->ReadToEnd();

fout->Close();

StreamWriter^ fcd = gcnew StreamWriter("code\_descriptor.txt", false);

String^ word = "";

String^ out = "";

wchar\_t c;

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

c = str[i];

if (c == '"')

{

do word += c; while (++i < str->Length && (c = str[i]) != '"');

word += c;

}

else if (numbers(&c) || c == '0')

{

do {

word += c;

if (++i < str->Length) c = str[i];

} while (i < str->Length && (numbers(&c) || c == '0' || c == '.'));

if (c == 'e' || c == 'E') {

word += c;

if (i + 1 < str->Length) {

c = str[++i];

if ((c == '+' || c == '-') && ++i < str->Length) {

word += c;

c = str[i];

do {

word += c;

if (++i < str->Length) c = str[i];

} while (i < str->Length && (numbers(&c) || c == '0'));

}

else i--;

}

}

else i--;

}

else if (!end\_condition(&c))//любой другой символ не разделитель

{

do word += c; while (++i < str->Length && !end\_condition(&(c = str[i])));

i--;

}

else if (comparison\_operators(&c))

{

word += c;

if (++i < str->Length && comparison\_operators(&(c = str[i]))) word += c;

else i--;

}

else if (other(&c))

{

word += c;

}

else if (operation\_sings(&c))

{

word += c;

if (++i < str->Length && operation\_sings(&(c = str[i])))

word += c;

else i--;

}

else if (separators(&c))

{

word += c;

}

//вывод дескрипторного кода

if (word != "") {

for (size\_t j = 0; j < \*row\_lexeme; j++)

if (word == System::Convert::ToString(table\_lexeme->Rows[j]->Cells[1]->Value))

out = "(" + Column10->HeaderCell->Value + "," + (j + 1) + ")";

for (size\_t j = 0; j < \*row\_id; j++)

if (word == System::Convert::ToString(table\_id->Rows[j]->Cells[1]->Value))

out = "(" + Column20->HeaderCell->Value + "," + (j + 1) + ")";

for (size\_t j = 0; j < \*row\_const; j++)

if (word == System::Convert::ToString(table\_const->Rows[j]->Cells[1]->Value))

out = "(" + Column30->HeaderCell->Value + ", " + (j + 1) + ")";

for (size\_t j = 0; j < \*row\_comparison\_operators; j++)

if (word == System::Convert::ToString(table\_comparison\_operators->Rows[j]->Cells[1]->Value))

out = "(" + Column40->HeaderCell->Value + "," + (j + 1) + ")";

for (size\_t j = 0; j < \*row\_operation\_sings; j++)

if (word == System::Convert::ToString(table\_operation\_sings->Rows[j]->Cells[1]->Value))

out = "(" + Column50->HeaderCell->Value + "," + (j + 1) + ")";

for (size\_t j = 0; j < \*row\_separators; j++)

if (word == System::Convert::ToString(table\_separators->Rows[j]->Cells[1]->Value))

out = "(" + Column60->HeaderCell->Value + "," + (j + 1) + ")";

if (out != "") fcd->Write(out + " "), out = "";

word = "";

}

}

fcd->Close();

StreamReader^ fcdout = gcnew StreamReader("code\_descriptor.txt");

descriptor\_code->Text = fcdout->ReadToEnd();

fcdout->Close();

}

void TA2coursework::MyForm::code\_pre(int\* row\_lexeme, int\* row\_id, int\* row\_const, int\* row\_comparison\_operators, int\* row\_operation\_sings, int\* row\_separators)

{

StreamReader^ fcdout = gcnew StreamReader("code\_descriptor.txt");

String^ str = fcdout->ReadToEnd();

fcdout->Close();

StreamWriter^ fcp = gcnew StreamWriter("code\_pre.txt", false);

String^ table = "";

String^ number = "";

String^ pre\_word = "";

int int\_number;

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

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

i++;

while (str[i] != ',' && str[i])

table += str[i++];

i++;

while (str[i] != ')' && str[i])

number += str[i++];

int\_number = System::Convert::ToInt32(number) - 1;

if (table == System::Convert::ToString(Column10->HeaderCell->Value))

pre\_word = System::Convert::ToString(table\_lexeme->Rows[int\_number]->Cells[2]->Value);

else if (table == System::Convert::ToString(Column20->HeaderCell->Value))

pre\_word = System::Convert::ToString(table\_id->Rows[int\_number]->Cells[2]->Value);

else if (table == System::Convert::ToString(Column30->HeaderCell->Value))

pre\_word = System::Convert::ToString(table\_const->Rows[int\_number]->Cells[2]->Value);

else if (table == System::Convert::ToString(Column40->HeaderCell->Value))

pre\_word = System::Convert::ToString(table\_comparison\_operators->Rows[int\_number]->Cells[2]->Value);

else if (table == System::Convert::ToString(Column50->HeaderCell->Value))

pre\_word = System::Convert::ToString(table\_operation\_sings->Rows[int\_number]->Cells[2]->Value);

else if (table == System::Convert::ToString(Column60->HeaderCell->Value))

pre\_word = System::Convert::ToString(table\_separators->Rows[int\_number]->Cells[2]->Value);

fcp->Write(pre\_word + " ");

pre\_word = "";

table = "";

number = "";

}

}

fcp->Close();

StreamReader^ fcpout = gcnew StreamReader("code\_pre.txt");

pre\_code->Text = fcpout->ReadToEnd();

fcpout->Close();

}

System::Void TA2coursework::MyForm::run\_Click(System::Object^ sender, System::EventArgs^ e)

{

list\_of\_erros->Text = "";

String^ str = text\_in->Text;

cleaning(&str);

int row\_lexeme = 1, row\_id = 1, row\_const = 1, row\_comparison\_operators = 1, row\_operation\_sings = 1, row\_separators = 1;

//создание таблиц

table\_id->RowCount = 1;

table\_lexeme->RowCount = 1;

table\_const->RowCount = 1;

table\_comparison\_operators->RowCount = 1;

table\_operation\_sings->RowCount = 1;

table\_separators->RowCount = 1;

table\_id->RowCount = row\_id + 1;

table\_id->ColumnCount = 3;

table\_id->Rows[0]->Cells[0]->Value = "";

table\_id->Rows[0]->Cells[1]->Value = "";

table\_id->Rows[0]->Cells[2]->Value = "";

table\_lexeme->RowCount = row\_lexeme + 1;

table\_lexeme->ColumnCount = 3;

table\_lexeme->Rows[0]->Cells[0]->Value = "";

table\_lexeme->Rows[0]->Cells[1]->Value = "";

table\_lexeme->Rows[0]->Cells[2]->Value = "";

table\_const->RowCount = row\_const + 1;

table\_const->ColumnCount = 3;

table\_const->Rows[0]->Cells[0]->Value = "";

table\_const->Rows[0]->Cells[1]->Value = "";

table\_const->Rows[0]->Cells[2]->Value = "";

table\_comparison\_operators->RowCount = row\_comparison\_operators + 1;

table\_comparison\_operators->ColumnCount = 3;

table\_comparison\_operators->Rows[0]->Cells[0]->Value = "";

table\_comparison\_operators->Rows[0]->Cells[1]->Value = "";

table\_comparison\_operators->Rows[0]->Cells[2]->Value = "";

table\_operation\_sings->RowCount = row\_operation\_sings + 1;

table\_operation\_sings->ColumnCount = 3;

table\_operation\_sings->Rows[0]->Cells[0]->Value = "";

table\_operation\_sings->Rows[0]->Cells[1]->Value = "";

table\_operation\_sings->Rows[0]->Cells[2]->Value = "";

table\_separators->RowCount = row\_separators + 1;

table\_separators->ColumnCount = 3;

table\_separators->Rows[0]->Cells[0]->Value = "";

table\_separators->Rows[0]->Cells[1]->Value = "";

table\_separators->Rows[0]->Cells[2]->Value = "";

processing(&row\_lexeme, &row\_id, &row\_const, &row\_comparison\_operators, &row\_operation\_sings, &row\_separators);

code\_descriptor(&row\_lexeme, &row\_id, &row\_const, &row\_comparison\_operators, &row\_operation\_sings, &row\_separators);

code\_pre(&row\_lexeme, &row\_id, &row\_const, &row\_comparison\_operators, &row\_operation\_sings, &row\_separators);

}

# Приложение 4

Граф конечного автомата