минобрнауки россии

федеральное государственное бюджетное

образовательное учреждение высшего образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

|  |  |
| --- | --- |
| Институт (факультет) | Институт Информационных Технологий |
| Кафедра | Математического и Программного Обеспечения ЭВМ |

**КУРСОВАЯ РАБОТА**

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

|  |  |
| --- | --- |
| на тему | «Разработка алгоритмического обеспечения и построение |
| лексическогоанализатора компилятора» | |

|  |
| --- |
| Выполнил студент группы |
| 1ПИб-02-3оп-22 |
| направление подготовки (специальности) |
| 09.03.04, Программная инженерия |
| *шифр, наименование* |
| Маркелов Сергей Александрович |
| *фамилия, имя, отчество* |

|  |
| --- |
| Руководители |
| Ганичева Оксана Георгиевна  Пышницкий Константин Михайлович |
| *фамилия, имя, отчество* |
| доцент |
| *должность* |

|  |
| --- |
| Дата представления работы |
| «\_\_14\_\_»\_\_\_\_\_\_\_июня\_\_\_\_\_2024 г. |
|  |
| Заключение о допуске к защите |
|  |
|  |
|  |
|  |
| Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| количество баллов |
| Подпись преподавателя\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Череповец, 2024

*Год*

Оглавление

[Введение 3](#_Toc168790662)

[1. Изучение и описание предметной области 5](#_Toc168790663)

[2. Основная часть 13](#_Toc168790664)

[2.1. Постановка задачи 13](#_Toc168790665)

[2.2. Описание конечного автомата 17](#_Toc168790666)

[2.3. Построение грамматики по конечному автомату 21](#_Toc168790667)

[2.4. Построение КС-грамматики, порождения и дерева вывода 30](#_Toc168790668)

[3. Логическое проектирование 49](#_Toc168790669)

[3.1. Блок-схемы 49](#_Toc168790670)

[3.2. Словесное описание алгоритма 60](#_Toc168790671)

[3.3. Оценка сложности алгоритма 61](#_Toc168790672)

[4. Физическое проектирование 63](#_Toc168790673)

[4.1. Выбор структур данных 63](#_Toc168790674)

[4.2. Выбор функций 64](#_Toc168790675)

[5. Проектирование интерфейса 65](#_Toc168790676)

[6. Тестирование программного обеспечения 67](#_Toc168790677)

[Заключение 71](#_Toc168790678)

[Список литературы 72](#_Toc168790679)

[Приложение 1. Техническое задание 74](#_Toc168790680)

[Приложение 2. Руководство пользователя 82](#_Toc168790681)

[Приложение 3. Текст программы 91](#_Toc168790682)

[Приложение 4. Граф конечного автомата 171](#_Toc168790683)

# Введение

Программы играют огромную роль в современном мире. Существует большое количество различных видов программ: драйвера, утилиты, плееры, браузеры, различные редакторы (текстовые, графические, музыкальные видеоредакторы), мессенджеры, архиваторы, антивирусы и многие другие.

Благодаря программам мы можем смотреть, обрабатывать или создавать фотографии, видео, музыку, работать с документами, обрабатывать данные и многое другое.

Программы создаются благодаря языкам программирования – формальным знаковым системам, предназначенным для описания алгоритмов обработки данных. Каждый язык программирования имеет свой алфавит и синтаксис [[5]](#_Источники).

Компиляция программы – это процесс преобразования исходного кода, написанного на каком-либо языке программирования, в исполняемый файл, который компьютер может запустить [6].

Основными этапами компиляции программы являются анализ и синтез.

Во время анализа исходный код программы разбивается на составные части и создается его промежуточное представление. Этот процесс включает в себя лексический анализ, синтаксический анализ и семантический анализ. Лексический анализ выделяет лексемы (ключевые слова, идентификаторы и т. д.), синтаксический анализ строит структуру программы на основе этих лексем, а семантический анализ проверяет смысловую корректность кода.

Во время синтеза из промежуточного представления конструируется целевая программа. Этот этап включает в себя генерацию промежуточного кода, оптимизацию этого кода для улучшения производительности или эффективности, а затем генерацию итогового кода.

Целью курсовой работы является разработка лексического анализатора – программного обеспечения, выполняющего первую фазу компиляции программы.

Разработка программы была осуществлена на языке C++. Для создания программы использовалась среда разработки Visual Studio 2022. Для создания интерфейса программы применялись Windows Forms.

# Изучение и описание предметной области

Лексический анализ – это первая фаза компиляции программы, в ходе которой входной текст программы преобразуется из линейной последовательности символов в последовательность лексем [7]. Основное назначение лексического анализа – разбиение текста программы на лексемы, такие как идентификаторы, ключевые слова, операторы, разделители, константы и т. д.

Основные задачи и функции лексического анализа включают в себя:

* распознавание и классификацию лексем;
* удаление из текста программы незначащих символов (например, комментариев или лишних пробелов и строк);
* подсчет количества строк в программе;
* создание таблиц лексем для передачи их следующему этапу компиляции;
* выявление ошибок в написании лексем, вывод сообщений о таких ошибках;
* формирование дескрипторного кода и псевдокода.

Лексический анализатор работает по следующему принципу:

1. Из входного потока берется символ, который будет использован для запуска лексического анализатора. Анализатор распознает данный символ принимает решение о том, является ли он частью какой-либо лексемы или же является ошибочным.
2. Далее лексический анализатор продолжает просматривать входной поток и выделять символы, образующие лексему, до тех пор, пока не будет обнаружен символ, который завершает лексему, или символ, который является ошибочным. Границами лексем могут служить пробелы, знаки табуляции, точки с запятой, двоеточия и некоторые другие символы.
3. При успешном распознавании лексемы она заносится в таблицу, соответствующую типу распознанной лексемы, после чего анализатор возвращается к пункту 1 и продолжает рассматривать входной поток символов, начиная с позиции, где был прерван в прошлый раз.
4. Если лексический анализатор обнаружит ошибочный символ, дальнейшее поведение программы зависит от ее реализации: может быть выведено сообщение об ошибке, анализатор может продолжить поиск лексем или выполнение программы может быть прервано.

Принципы построения лексического анализатора включают в себя использование регулярных выражений для описания лексических конструкций языка программирования и построение конечного автомата для распознавания лексем.

Конечный автомат (КА) – это математическая модель, используемая для представления и анализа дискретных систем, которые могут находиться в одном из конечного числа состояний в любой момент времени [2].

Конечный автомат обозначается как A = {X, S, S0, F, δ}, где:

* X – конечное множество входных символов (алфавит);
* S – конечное множество состояний;
* S0 – начальное состояние, элемент множества S;
* F – множество заключительных (конечных) состояний, подмножество множества S;
* δ – функция переходов, определяющая переходы между различными состояниями множества S.

Конечный автомат работает по следующему принципу:

1. Автомат начинается в начальном состоянии S0.
2. Он читает входные символы по одному за раз.
3. В зависимости от текущего состояния и считанного символа, переходная функция определяет новое состояние.
4. Этот процесс продолжается до тех пор, пока не будут прочитаны все символы входной последовательности.
5. Если автомат находится в одном из заключительных состояний после обработки всех входных символов, входная последовательность считается принятой. Иначе она отвергается.

Конечный автомат имеет следующие характеристики:

* Детерминированность: конечный автомат может быть либо детерминированным (т. е. каждый переход у автомата определяется однозначно для каждого состояния и каждого входного символа), либо недетерминированным (т. е. переход может быть определен неоднозначно, для одного и того же состояния и входного символа может быть несколько возможных переходов).
* Синхронность: конечный автомат может быть либо синхронным (т. е. он изменяет свои состояния при поступлении синхронизирующего сигнала), либо асинхронным (т. е. он изменяет свои состояния, основываясь только на поступающих входных сигналах, которые могут поступить в любой момент времени).
* Тип автомата: конечные автоматы бывают 2 типов – распознаватели и преобразователи. Автомат-распознаватель принимает или отвергает входные строки на основе того, соответствуют ли они определенному языку или шаблону. Автомат-преобразователь преобразует входные данные в выходные на основе определенных правил или функций. В отличие от распознавателя, преобразователь не просто принимает или отвергает входные данные, а еще и генерирует соответствующий выход.

Регулярные выражения – это последовательности символов, описывающие шаблоны для поиска, сопоставления и манипулирования строками текста. Регулярные выражения используются для определения и нахождения подстрок, соответствующих определённым критериям [3].

Синтаксический анализ – это процесс анализа структуры предложения в соответствии с формальной грамматикой языка программирования. Основное назначение синтаксического анализа – проверка правильности синтаксиса программы, выявление ошибок и построение структурного представления программы. Синтаксический анализатор работает на основе набора правил грамматики языка программирования и использует выводы лексического анализатора (лексемы) для построения структурного представления программы [8].

Результатом синтаксического анализа является дерево разбора (синтаксическое дерево).

Основные задачи и функции синтаксического анализа включают в себя:

* Поиск и выделение синтаксических конструкций в тексте программы;
* Определение типа и проверку правильности синтаксических конструкций;
* Представление синтаксических конструкций в виде, удобном для дальнейшего использования при генерации кода итоговой программы.

В основе синтаксического анализатора лежит распознаватель входной программы на основе грамматик входного языка. Синтаксические конструкции языка можно описать с помощью контекстно-свободных грамматик, либо с помощью регулярных грамматик.

Грамматика – это набор правил, определяющих синтаксис языка, то есть структуру правильных выражений, конструкций и программ, написанных на этом языке. Грамматики играют ключевую роль в компиляторах и интерпретаторах для анализа и обработки исходного кода [2].

Грамматика обозначается как G (T, N, P, S), где:

* G – обозначение самой грамматики;
* T – множество терминальных символов (алфавит). Эти символы появляются в конечных строках языка;
* N – множество нетерминальных символов. Эти символы используются только для определения правил и исчезают в процессе порождения строки.
* P – множество правил продукции. Эти правила определяют, как одни символы могут быть заменены другими.
* S – начальный символ, с которого начинается процесс порождения строки.

Иерархия Хомского – это классификация формальных грамматик, предложенная Ноамом Хомским. Она разделяет грамматики на четыре уровня, в зависимости от их сложности и ограничений, наложенных на правила продукции [9].

Уровни иерархии Хомского:

* Тип 0: неограниченные грамматики. На структуру их правил не накладывается никаких ограничений. Для грамматики вида G (T, N, P, S); V = N ∪ T, правила имеют вид α → β, где α ∈ V⁺, β ∈ V\*. Это самый общий вид грамматик, охватывающий вообще все формальные грамматики. Практическое применение у таких грамматик отсутствует
* Тип 1: контекстно-зависимые грамматики (КЗ-грамматики). В этот тип входят два основных класса грамматик: контекстно-зависимые и неукорачивающие грамматики. У контекстно-зависимых грамматик G (T, N, P, S); V = N ∪ T правила имеют вид: α1Aα2 → α1βα2, где α1, α2 ∈ V\*, A ∈ N, β ∈ V+. При построении предложений заданного ими языка один и тот же нетерминальный символ может быть заменен на ту или иную цепочку символов в зависимости от того контекста, в котором он встречается. Именно поэтому эти грамматики называются «контекстно-зависимыми». У неукорачивающих грамматик G (T, N, P, S); V = N ∪ T правила имеют вид: α → β, где α, β ∈ V+, |β| ≥ |α|. При построении предложений языка, заданного такой грамматикой, любая цепочка символов может быть заменена на цепочку символов не меньшей длины. Эти 2 класса грамматик являются эквивалентными. Это значит, что для любого языка, заданного контекстно-зависимой грамматикой, можно построить не укорачивающую грамматику, которая будет задавать эквивалентный язык.
* Тип 2: контекстно-свободные грамматики (КС-грамматики). У такой грамматики G (T, N, P, S); V= N ∪ T правила имеют вид: A → β, где A ∈ N, β ∈ V+. Такие грамматики также иногда называют неукорачивающими контекстно-свободными грамматиками (НКС-грамматиками), потому что в правой части правил у них всегда должен стоять как минимум один символ. Также существует почти эквивалентный им класс грамматик – укорачивающие контекстно-свободные грамматики (УКС-грамматики). У такой грамматики G (T, N, P, S), V = N ∪ T правила имеют вид: A → β, где β ∈ V\*, A ∈ VN. Разница между этими 2 классами грамматик заключается лишь в том, что в УКС-грамматиках в правой части правил может присутствовать пустая цепочка, а в НКС-грамматиках – нет. Язык, заданный НКС-грамматикой, не может содержать пустой цепочки. Эти 2 класса грамматик почти эквивалентны. КС-грамматики широко используются при описании синтаксических конструкций языков программирования. Синтаксис большинства известных языков программирования основан именно на КС-грамматиках.
* Тип 3: регулярные грамматики. Они определяют в точности все регулярные языки и поэтому эквивалентны конечным автоматам и регулярным выражениям. Регулярная грамматика может быть задана набором правил как левая или правая регулярная грамматика. Классы правых и левых регулярных грамматик эквивалентны – каждый в отдельности достаточен для задания всех регулярных языков. Любая регулярная грамматика может быть преобразована из левой в правую и наоборот.

Каждый уровень иерархии Хомского включает в себя все уровни ниже его. Т. е., все регулярные грамматики (тип 3) являются контекстно-свободными (тип 2); все контекстно-свободные грамматики (тип 2) являются контекстно-зависимыми (тип 1); все контекстно-зависимые грамматики (тип 1) являются неограниченными (тип 0).

Порождение – это процесс создания строк из языка на основе правил грамматики. Порождение описывает, как из начального символа можно получить любую строку языка путем последовательного применения правил продукции.

Порождение строки из языка может происходить двумя основными способами: левосторонним и правосторонним порождением. Эти методы определяют порядок применения правил продукции при преобразовании начального символа в конечную строку.

Левостороннее порождение – это процесс, при котором на каждом шаге порождения первым выбирается и заменяется самый левый нетерминальный символ.

Правостороннее порождение – это процесс, при котором на каждом шаге порождения первым выбирается и заменяется самый правый нетерминальный символ.

Оба метода по итогу приводят к одной и той же конечной строке, но имеют различные промежуточные шаги.

Дерево вывода (дерево разбора, синтаксическое дерево) – это древовидная структура, которая иллюстрирует процесс порождения строки из языка на основе правил грамматики. Каждый узел дерева представляет собой применение одного из правил продукции, и путь от корня до листа показывает, как начальный символ трансформируется в терминальные символы.

Дерево вывода состоит из следующих компонентов:

* Корень дерева – начальный символ грамматики;
* Внутренние узлы – нетерминальные символы, к которым применяются правила продукции;
* Листья – терминальные символы, составляющие конечную строку;
* Ребра (ветви) – соединения между узлами, показывающие, какие символы заменяются какими.

Взаимодействие между лексическим и синтаксическим анализаторами обычно осуществляется путем передачи последовательности лексем с выхода лексического анализатора на вход синтаксическому анализатору. Таким образом, синтаксический анализатор работает с уже классифицированными лексемами, что позволяет ему проводить анализ структуры программы и выявлять синтаксические ошибки.

# Основная часть

## Постановка задачи

В данной курсовой работе требуется разработать модель работы компилятора, а именно этапы лексического и синтаксического анализов. Для этого необходимо разработать программу, которая моделирует работу лексического анализатора, и решает следующие задачи:

* Выделение из текста входной программы всех лексем, входящих в заданную языковую конструкцию;
* Удаление лишних пробелов и комментариев из входной строки;
* Подсчет количества строк в программе.
* Диагностика и локализация лексических ошибок.

Моделирование лексического анализатора должно базироваться на конечном автомате. Автомат должен быть описан с использованием формальной модели и таблицы переходов.

Также необходимо построить КС-грамматику, описывающую синтаксис инструкции языка программирования согласно своему варианту. Результатом работы синтаксического анализатора должно быть дерево вывода и соответствующее ему левостороннее порождение рассматриваемой цепочки.

Вариантом курсовой работы является задание «Оператор switch и if языка С++». По данному варианту был написан код для проведения анализа:

#include <iostream>

using namespace std;

void main(){

int a = 0, b = 0, c = 0;

cout << "Введите число от 0 до 5: ";

cin >> a;

cout << "Введите любое число: ";

cin >> b;

switch (a) { //оператор switch

case 0:

if (b != 0) {

c = a + b;

cout << c << endl;

}

else c = a - b;

break;

case 1:

if (b >= 0) {

c = a++;

cout << c << endl;

}

else c = b--;

break;

case 2:

if (b > 0) {

c = a \* b;

cout << c << endl;

}

else c = a / b;

break;

case 3:

if (b < 0) {

a += b;

cout << a << endl;

}

else b -= a;

c = a + b;

break;

case 4:

if (b <= 0) {

a \*= b;

cout << a << endl;

}

else b /= a;

c = a + b;

break;

case 5:

if (b == 0) c = b % a;

else c = a / b;

break;

/\*case 6:

if (b == a) c = a;

else c = b;

break;\*/

default:

cout << "Ошибка!" << endl;

break;

}

cout << c << endl;

}

Алгоритм программы:

1. Инициализируются переменные a, b, c типа int. Им всем присваивается начальные значения 0.
2. Программа просит пользователя ввести значение переменной a. Оно должно быть в интервале от 0 до 5.
3. Программа просит пользователя ввести значение переменной b. Оно может быть любым целым числом.
4. В зависимости от введенного значения переменной a:
   1. При a = 0: если значение переменной b не равно 0, переменной c присваивается значение a + b, затем значение c выводится на экран; иначе переменной c присваивается значение a – b.
   2. При a = 1: если значение переменной b больше либо равно 0, переменной c присваивается значение a++, затем значение c выводится на экран; иначе переменной c присваивается значение b--.
   3. При a = 2: если значение переменной b больше 0, переменной c присваивается значение a \* b, затем значение c выводится на экран; иначе переменной c присваивается значение a / b.
   4. При a = 3: если значение переменной b меньше 0, переменная a увеличивается на b; иначе переменная b уменьшается на a. Затем переменной c присваивается значение a + b.
   5. При a = 4: если значение переменной b меньше либо равно 0, переменная a умножается на b, затем значение a выводится на экран; иначе переменная b делится на a. Затем переменной c присваивается значение a + b.
   6. При a = 5: если значение переменной b равно значению переменной a, переменной c присваивается значение b % a (остаток от деления b на a); иначе переменной c присваивается значение a / b.
   7. При других значениях a выводится сообщение «Ошибка!».
5. Значение переменной c выводится на экран.

Результаты работы программы при различных значениях a и b представлены на рис. 1-4.

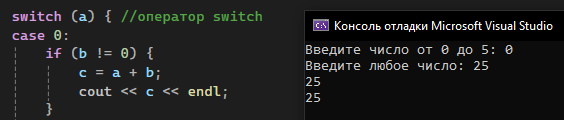


Рис. 1. Результат работы программы при a = 0, b = 25

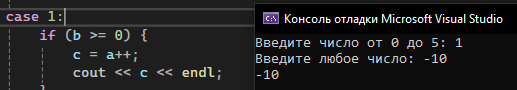


Рис. 2. Результат работы программы при a = 1, b = -10

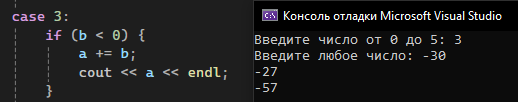


Рис. 3. Результат работы программы при a = 3, b = -30

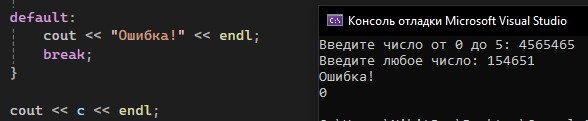


Рис. 4. Результат работы программы при a = 4565465, b = 154651

## Описание конечного автомата

Для описания работы лексического анализатора был смоделирован детерминированный конечный автомат.

Конечный автомат (КА) – это нефункциональный преобразователь информации, у которого реакция на входные данные зависит не только от текущего входа, но и от предыдущей истории входов. Он работает в дискретные моменты времени и представляет собой обобщенную диаграмму переходов, используемую для компиляции регулярных выражений в распознавателе [2].

Преимущества использования КА для построения лексического анализатора:

* КА может решать различные задачи компиляции, особенно лексического анализа;
* КА отличается высоким быстродействием;
* Моделирование КА требует фиксированного объема памяти, что упрощает управление памятью;
* Существует множество теорем и алгоритмов для конструирования и упрощения КА.

По типу переходов и построения графов, конечные автоматы делятся на детерминированные (ДКА) и недетерминированные (НКА).

ДКА имеют более высокую скорость распознавания лексем, так как переход по состояниям осуществляется единожды для определенного символа. Однако ДКА занимают больше места, поскольку переходы определены для каждого символа. НКА также способны распознавать регулярные множества и могут быть преобразованы в ДКА.

Функции переходов КА можно реализовать через таблицу переходов, где строки описывают состояния, а столбцы – входные символы. Основным преимуществом таблицы переходов является быстрый и понятный доступ к переходам, но она может занимать много места при большом входном алфавите.

Также КА можно представить в виде графа переходов, в котором вершины графа – это состояния, а дуги – переходы между ними, помеченные символами.

При программной реализации каждое состояние КА может быть представлено функцией или блоком функций, выполняющим не только переходы, но и дополнительные операции с входными данными, такие как обработка, удаление или проверка правильности.

КА является подходящей моделью для алгоритмов распознавания лексем в лексическом анализаторе. По любому КА можно построить регулярную грамматику, продукции которой соответствуют правилам переходов КА.

Для построения конечного автомата, необходимого для разработки лексического анализатора, был создан его граф. Фрагмент графа представлен на рис. 5. Полный граф представлен в приложении 4.

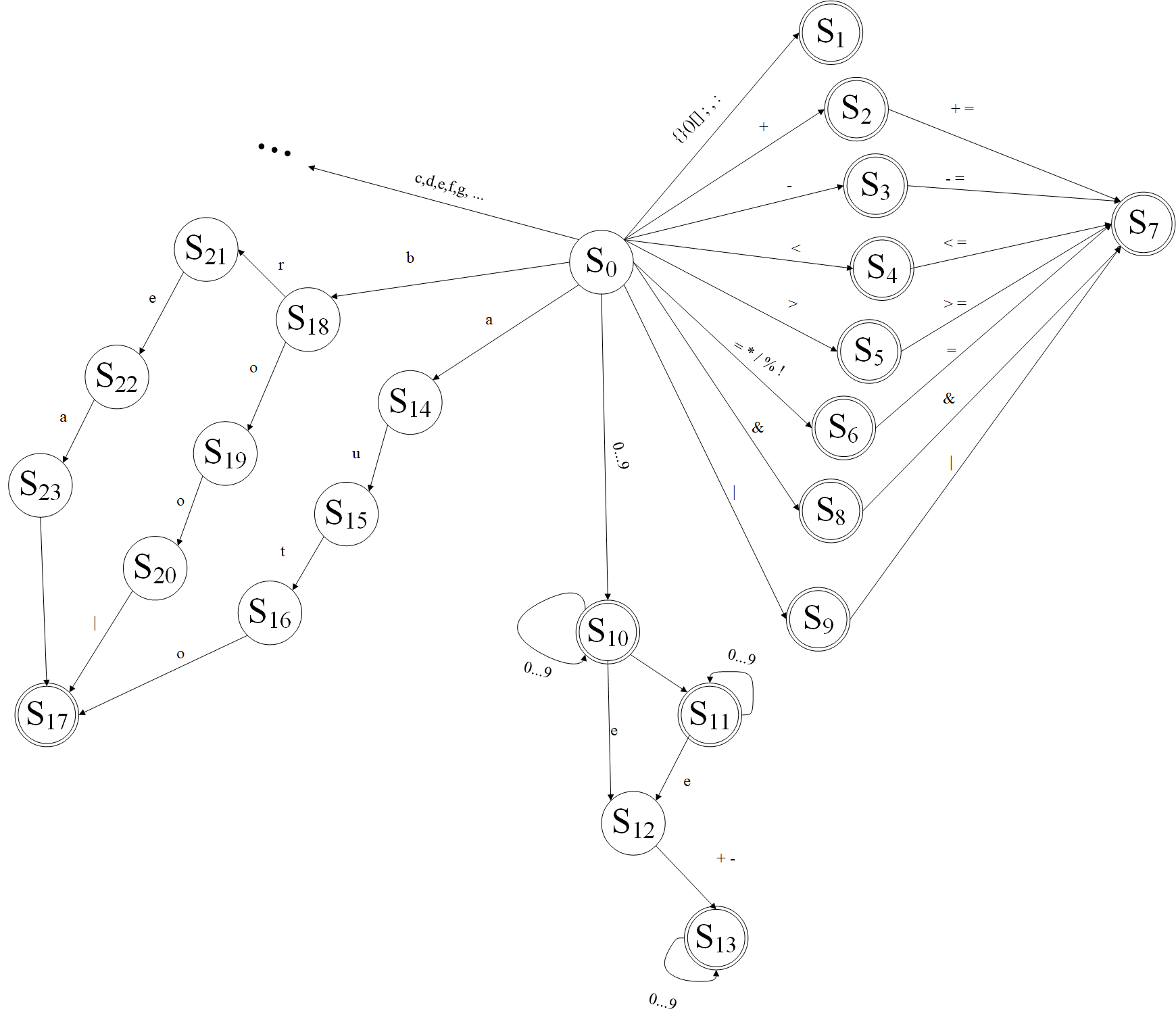


Рис. 5. Фрагмент графа конечного автомата

Описание конечного автомата:

A = {X, S, S0, F, δ}

X = {A…Z, a…z, 0…9, (, ), {, }, [, ], ‘, “, ;, ,, ., +, –, \*, /, %, =, <, >, &, |, #}

S = {S0, S1, S2, … S193, S194, S999}

S0 = {S0}

F = {S1, S2, …, S11, S13, S17, S999}, где:

S1 – разделители

S2 – S9 – знаки операций

S10, S11, S13 – константы

S17 – ключевые слова

S999 – идентификаторы

δ – функция переходов представлена в табл. 1

Таблица 1

Функция переходов

|  |  |  |  |
| --- | --- | --- | --- |
| (S0, ‘ ’) → S0 | (S0, \n) → S0 | (S0, {) → S1 | (S0, }) → S1 |
| (S0, ( ) → S1 | (S0, ) ) → S1 | (S0, [) → S1 | (S0, ]) → S1 |
| (S0, ;) → S1 | (S0, ,) → S1 | (S0, :) → S1 | (S0, +) → S2 |
| (S0, –) → S3 | (S0, <) → S4 | (S0, >) → S5 | (S0, =) → S6 |
| (S0, \*) → S6 | (S0, /) → S6 | (S0, %) → S6 | (S0, !) → S6 |
| (S0, &) → S8 | (S0, |) → S9 | (S0, =) → S6 | (S0, 0…9) → S10 |
| (S0, a) → S14 | (S0, b) → S18 | (S0, c) → S24 | (S0, d) → S40 |
| (S0, e) → S53 | (S0, f) → S65 | (S0, g) → S77 | (S0, i) → S80 |
| (S0, l) → S85 | (S0, m) → S88 | (S0, n) → S94 | (S0, o) → S108 |
| (S0, p) → S115 | (S0, r) → S131 | (S0, s) → S136 | (S0, t) → S158 |
| (S0, u) → S171 | (S0, v) → S183 | (S0, w) → S191 | (S0, h) → S999 |
| (S0, j) → S999 | (S0, k) → S999 | (S0, q) → S999 | (S0, x) → S999 |
| (S0, y) → S999 | (S0, z) → S999 | (S0, \_) → S999 | (S0, A…Z) → S999 |
| (S1) → S0 | (S2, +) → S7 | (S2, =) → S7 | (S3, –) → S7 |
| (S3, =) → S7 | (S4, <) → S7 | (S4, =) → S7 | (S5, >) → S7 |
| (S5, =) → S7 | (S6, =) → S7 | (S7) → S0 | (S8, &) → S7 |
| (S9, |) → S7 | (S10, 0…9) → S10 | (S10, .) → S11 | (S10, e) → S12 |
| (S10, E) → S12 | (S11, 0…9) → S11 | (S11, e) → S12 | (S11, E) → S12 |
| (S12, +) → S13 | (S12, –) → S13 | (S13, 0…9) → S13 | (S14, u) → S15 |
| (S15, t) → S16 | (S16, o) → S17 | (S17) → S0 | (S18, o) → S19 |
| (S18, r) → S21 | (S19, o) → S20 | (S20, l) → S17 | (S21, e) → S22 |
| (S22, a) → S23 | (S23, k) → S17 | (S24, a) → S25 | (S24, h) → S29 |
| (S24, l) → S31 | (S24, o) → S34 | (S25, s) → S26 | (S25, t) → S27 |
| (S26, e) → S17 | (S27, c) → S28 | (S28, h) → S17 | (S29, a) → S30 |

Продолжение табл. 1

|  |  |  |  |
| --- | --- | --- | --- |
| (S30, a) → S17 | (S31, a) → S32 | (S32, s) → S33 | (S33, s) → S17 |
| (S34, n) → S35 | (S35, с) → S36 | (S35, s) → S39 | (S36, e) → S37 |
| (S37, p) → S38 | (S38, t) → S17 | (S39, t) → S17 | (S40, e) → S41 |
| (S40, o) → S49 | (S41, f) → S42 | (S41, l) → S46 | (S42, a) → S43 |
| (S43, u) → S44 | (S44, l) → S45 | (S45, t) → S17 | (S46, e) → S47 |
| (S47, t) → S48 | (S48, e) → S17 | (S49, u) → S50 | (S50, b) → S51 |
| (S51, l) → S52 | (S52, e) → S17 | (S53, l) → S54 | (S53, n) → S56 |
| (S53, x) → S58 | (S54, s) → S55 | (S55, e) → S17 | (S56, u) → S57 |
| (S57, m) → S17 | (S58, p) → S59 | (S58, t) → S62 | (S59, o) → S60 |
| (S60, r) → S61 | (S61, t) → S17 | (S62, e) → S63 | (S63, r) → S64 |
| (S64, n) → S17 | (S65, a) → S66 | (S65, l) → S69 | (S65, o) → S72 |
| (S65, r) → S73 | (S66, l) → S67 | (S67, s) → S68 | (S68, e) → S17 |
| (S69, o) → S70 | (S70, a) → S71 | (S71, t) → S71 | (S72, r) → S17 |
| (S73, i) → S74 | (S74, e) → S75 | (S75, n) → S76 | (S76, d) → S17 |
| (S77, o) → S78 | (S78, t) → S79 | (S79, o) → S17 | (S80, f) → S17 |
| (S80, n) → S81 | (S81, l) → S82 | (S82, t) → S17 | (S82, i) → S83 |
| (S83, n) → S84 | (S84, e) → S17 | (S85, o) → S86 | (S86, n) → S87 |
| (S87, g) → S17 | (S88, u) → S89 | (S89, t) → S90 | (S90, a) → S91 |
| (S91, b) → S92 | (S92, l) → S93 | (S93, e) → S17 | (S94, a) → S95 |
| (S94, e) → S102 | (S94, u) → S103 | (S95, m) → S96 | (S96, a) → S97 |
| (S97, s) → S98 | (S98, p) → S99 | (S99, a) → S100 | (S100, c) → S101 |
| (S101, e) → S17 | (S102, w) → S17 | (S103, l) → S104 | (S104, l) → S105 |
| (S105, p) → S106 | (S106, t) → S107 | (S107, r) → S17 | (S108, p) → S109 |
| (S109, e) → S110 | (S110, r) → S111 | (S111, a) → S112 | (S112, t) → S113 |
| (S113, o) → S114 | (S114, r) → S17 | (S115, r) → S116 | (S115, u) → S127 |
| (S116, i) → S117 | (S117, v) → S118 | (S118, a) → S119 | (S119, t) → S120 |
| (S120, e) → S17 | (S121, t) → S122 | (S122, e) → S123 | (S123, c) → S124 |
| (S124, t) → S125 | (S125, e) → S126 | (S126, d) → S127 | (S127, b) → S128 |
| (S128, l) → S129 | (S129, i) → S130 | (S130, c) → S17 | (S131, e) → S132 |
| (S132, t) → S133 | (S133, u) → S134 | (S134, r) → S135 | (S135, n) → S17 |
| (S136, h) → S137 | (S136, i) → S140 | (S136, t) → S147 | (S136, w) → S154 |
| (S137, o) → S138 | (S138, r) → S139 | (S139, i) → S17 | (S140, g) → S141 |
| (S140, z) → S144 | (S141, n) → S142 | (S142, e) → S143 | (S143, d) → S17 |
| (S144, e) → S145 | (S145, o) → S146 | (S146, f) → S17 | (S147, a) → S148 |
| (S147, r) → S151 | (S148, t) → S149 | (S149, i) → S150 | (S150, c) → S17 |
| (S151, u) → S152 | (S152, с) → S153 | (S153, t) → S17 | (S154, i) → S155 |
| (S155, t) → S156 | (S156, c) → S157 | (S157, h) → S17 | (S158, e) → S159 |
| (S158, h) → S165 | (S158, r) → S169 | (S159, m) → S160 | (S160, p) → S161 |
| (S161, l) → S162 | (S162, a) → S163 | (S163, t) → S164 | (S164, e) → S17 |
| (S165, i) → S166 | (S165, r) → S167 | (S166, s) → S17 | (S167, o) → S168 |
| (S168, w) → S17 | (S169, u) → S170 | (S169, y) → S17 | (S170, e) → S17 |
| (S171, n) → S172 | (S171, s) → S180 | (S172, i) → S173 | (S172, s) → S175 |
| (S173, o) → S174 | (S174, n) → S17 | (S175, i) → S176 | (S176, g) → S177 |
| (S177, n) → S178 | (S178, e) → S179 | (S179, d) → S17 | (S180, i) → S181 |
| (S181, n) → S182 | (S182, g) → S17 | (S183, i) → S184 | (S183, o) → S189 |
| (S184, r) → S185 | (S185, t) → S186 | (S186, u) → S187 | (S187, a) → S188 |
| (S188, l) → S17 | (S189, i) → S190 | (S190, d) → S17 | (S191, h) → S192 |
| (S192, i) → S193 | (S193, l) → S194 | (S194, e) → S17 |  |

## Построение грамматики по конечному автомату

Грамматика – это один из точных способов задания языка. Она определяет, является ли конкретная языковая конструкция правильной в рамках данного языка, то есть принадлежит ли она этому языку. Грамматика задает правила порождения цепочек символов, составляющих язык, и таким образом является генератором этих цепочек [3].

Грамматику можно описывать различными методами, однако существует единый способ, называемый формальным определением грамматики. Все грамматики классифицируются по типам, основываясь на структуре их правил, по иерархии Хомского. Если все правила грамматики соответствуют определенной структуре, эта грамматика относится к конкретному типу в иерархии Хомского. Достаточно одного правила, не соответствующего общей структуре, чтобы грамматика не попадала в заданный тип. Один из типов в иерархии Хомского – регулярные или автоматные грамматики, которые будут использоваться в данной курсовой работе.

Регулярные (автоматные) грамматики – это тип формальных грамматик, которые используются для описания простейших конструкций языка. Эти грамматики удобны для описания языков, которые можно распознавать конечными автоматами.

Регулярные грамматики точно определяют все регулярные языки, поэтому они эквивалентны конечным автоматам и регулярным выражениям.

Регулярная грамматика может быть задана набором правил в виде левой или правой регулярной грамматики. Классы правых и левых регулярных грамматик эквивалентны: каждый из них способен задать все регулярные языки. Любую регулярную грамматику можно преобразовать из левой в правую и наоборот.

Совмещенная таблица соответствия команды конечного автомата и правила регулярной грамматики представлена в табл. 2.

Таблица 2

Совмещенная таблица соответствия команды конечного автомата и правила регулярной грамматики

|  |  |
| --- | --- |
| Формальное описание конечного автомата | Формальное описание регулярной грамматики по заданному конечному автомату |
| X = {A…Z, a…z, 0…9, (, ), {, }, [, ], ‘, “, ;, ,, ., +, –, \*, /, %, =, <, >, &, |, #}  S = {S0, S1, S2, … S193, S194, S999}  S0 = {S0}  F = {S1, S2, …, S11, S13, S17, S999},  где:  S1 – разделители  S2 – S9 – знаки операций  S10, S11, S13 – константы  S17 – ключевые слова  S999 – идентификаторы  δ: |  |
| (S0, ‘ ’) → S0 | S0 → ‘ ’ S0  S0 → ‘ ’ |
| (S0, \n) → S0 | S0 → \n S0  S0 → \n |
| (S0, {) → S1 | S0 → { S1  S0 → { |
| (S0, }) → S1 | S0 → } S1  S0 → } |
| (S0, ( ) → S1 | S0 → ( S1  S0 → ( |
| (S0, ) ) → S1 | S0 → ) S1  S0 → ) |
| (S0, [) → S1 | S0 → [ S1  S0 → [ |
| (S0, ]) → S1 | S0 → ] S1  S0 → ] |
| (S0, ;) → S1 | S0 → ; S1  S0 → ; |
| (S0, ,) → S1 | S0 → , S1  S0 → , |
| (S0, :) → S1 | S0 → : S1  S0 → : |
| (S0, +) → S2 | S0 → + S2  S0 → + |
| (S0, –) → S3 | S0 → – S3  S0 → – |
| (S0, <) → S4 | S0 → < S4  S0 → < |
| (S0, >) → S5 | S0 → > S5  S0 → > |

Продолжение табл. 2

|  |  |
| --- | --- |
| (S0, =) → S6 | S0 → = S6  S0 → = |
| (S0, \*) → S6 | S0 → \* S6  S0 → \* |
| (S0, /) → S6 | S0 → / S6  S0 → / |
| (S0, %) → S6 | S0 → % S6  S0 → % |
| (S0, !) → S6 | S0 → ! S6  S0 → ! |
| (S0, &) → S8 | S0 → & S8  S0 → & |
| (S0, |) → S9 | S0 → | S9  S0 → | |
| (S0, 0…9) → S10 | S0 → 0…9 S10  S0 → 0…9 |
| (S0, a) → S14 | S0 → a S14 |
| (S0, b) → S18 | S0 → b S18 |
| (S0, c) → S24 | S0 → c S24 |
| (S0, d) → S40 | S0 → d S40 |
| (S0, e) → S53 | S0 → e S53 |
| (S0, f) → S65 | S0 → f S65 |
| (S0, g) → S77 | S0 → g S77 |
| (S0, i) → S80 | S0 → i S80 |
| (S0, l) → S85 | S0 → l S85 |
| (S0, m) → S88 | S0 → m S88 |
| (S0, n) → S94 | S0 → n S94 |
| (S0, o) → S108 | S0 → o S108 |
| (S0, p) → S115 | S0 → p S115 |
| (S0, r) → S131 | S0 → r S131 |
| (S0, s) → S136 | S0 → s S136 |
| (S0, t) → S158 | S0 → t S158 |
| (S0, u) → S171 | S0 → u S171 |
| (S0, v) → S183 | S0 → v S183 |
| (S0, w) → S191 | S0 → w S191 |
| (S0, h) → S999 | S0 → h S999  S0 → h |
| (S0, j) → S999 | S0 → j S999  S0 → j |
| (S0, k) → S999 | S0 → k S999  S0 → k |
| (S0, q) → S999 | S0 → q S999  S0 → q |
| (S0, x) → S999 | S0 → x S999  S0 → x |
| (S0, y) → S999 | S0 → y S999  S0 → y |
| (S0, z) → S999 | S0 → z S999  S0 → z |

Продолжение табл. 2

|  |  |
| --- | --- |
| (S0, \_) → S999 | S0 → \_ S999  S0 → \_ |
| (S0, A…Z) → S999 | S0 → A…Z S999  S0 → A…Z |
| (S1) → S0 | S1 → S0 |
| (S2, +) → S7 | S2 → + S7  S2 → + |
| (S2, =) → S7 | S2 → = S7  S2 → = |
| (S3, –) → S7 | S3 → – S7  S3 → – |
| (S3, =) → S7 | S3 → = S7  S3 → = |
| (S4, <) → S7 | S4 → < S7  S4 → < |
| (S4, =) → S7 | S4 → = S7  S4 → = |
| (S5, >) → S7 | S5 → > S7  S5 → > |
| (S5, =) → S7 | S5 → = S7  S5 → = |
| (S6, =) → S7 | S6 → = S7  S6 → = |
| (S7) → S0 | S7 → S0 |
| (S8, &) → S7 | S8 → & S7  S8 → & |
| (S9, |) → S7 | S9 → | S7  S9 → | |
| (S10, 0…9) → S10 | S10 → 0…9 S7  S10 → 0…9 |
| (S10, .) → S11 | S10 → . S7  S10 → . |
| (S10, e) → S12 | S10 → e S12  S10 → e |
| (S10, E) → S12 | S10 → E S12  S10 → E |
| (S11, 0…9) → S11 | S11 → 0…9 S11  S11 → 0…9 |
| (S11, e) → S12 | S11 → e S12  S11 → e |
| (S11, E) → S12 | S11 → E S12  S11 → E |
| (S12, +) → S13 | S12 → + S13  S12 → + |
| (S12, –) → S13 | S12 → – S13  S12 → – |
| (S13, 0…9) → S13 | S13 → 0…9 S13  S13 → 0…9 |
| (S14, u) → S15 | S14 → u S15 |

Продолжение табл. 2

|  |  |
| --- | --- |
| (S15, t) → S16 | S15 → t S16 |
| (S16, o) → S17 | S16 → o S17  S16 → o |
| (S17) → S0 | S17 → S0 |
| (S18, o) → S19 | S18 → o S19 |
| (S18, r) → S21 | S19 → r S17 |
| (S19, o) → S20 | S19 → o S20 |
| (S20, l) → S17 | S20 → l S17  S20 → l |
| (S21, e) → S22 | S21 → e S22 |
| (S22, a) → S23 | S22 → a S23 |
| (S23, k) → S17 | S23 → k S17  S23 → k |
| (S24, a) → S25 | S24 → a S25 |
| (S24, h) → S29 | S24 → h S29 |
| (S24, l) → S31 | S24 → l S31 |
| (S24, o) → S34 | S24 → o S34 |
| (S25, s) → S26 | S25 → s S26 |
| (S25, t) → S27 | S25 → t S27 |
| (S26, e) → S17 | S26 → e S17  S26 → e |
| (S27, c) → S28 | S27 → c S28 |
| (S28, h) → S17 | S28 → h S17  S28 → h |
| (S29, a) → S30 | S29 → a S30 |
| (S30, a) → S17 | S30 → a S17  S30 → a |
| (S31, a) → S32 | S31 → a S32 |
| (S32, s) → S33 | S32 → s S33 |
| (S33, s) → S17 | S33 → s S17  S33 → s |
| (S34, n) → S35 | S34 → n S35 |
| (S35, с) → S36 | S35 → c S36 |
| (S35, s) → S39 | S35 → s S39 |
| (S36, e) → S37 | S36 → e S37 |
| (S37, p) → S38 | S37 → p S38 |
| (S38, t) → S17 | S38 → t S17  S38 → t |
| (S39, t) → S17 | S39 → t S17  S39 → t |
| (S40, e) → S41 | S40 → e S41 |
| (S40, o) → S49 | S40 → o S49 |
| (S41, f) → S42 | S41 → f S42 |
| (S41, l) → S46 | S41 → l S46 |
| (S42, a) → S43 | S42 → a S43 |
| (S43, u) → S44 | S43 → u S44 |
| (S44, l) → S45 | S44 → l S45 |
| (S45, t) → S17 | S45 → t S17  S45 → t |

Продолжение табл. 2

|  |  |
| --- | --- |
| (S46, e) → S47 | S46 → e S47 |
| (S47, t) → S48 | S47 → t S48 |
| (S48, e) → S17 | S48 → t S17  S48 → t |
| (S49, u) → S50 | S49 → u S50 |
| (S50, b) → S51 | S50 → b S51 |
| (S51, l) → S52 | S51 → l S52 |
| (S52, e) → S17 | S52 → e S17  S52 → e |
| (S53, l) → S54 | S53 → l S54 |
| (S53, n) → S56 | S53 → n S56 |
| (S53, x) → S58 | S53 → x S58 |
| (S54, s) → S55 | S54 → s S55 |
| (S55, e) → S17 | S55 → e S17  S55 → e |
| (S56, u) → S57 | S56 → u S57 |
| (S57, m) → S17 | S57 → m S17  S57 → m |
| (S58, p) → S59 | S58 → p S59 |
| (S58, t) → S62 | S58 → t S62 |
| (S59, o) → S60 | S59 → o S60 |
| (S60, r) → S61 | S60 → r S61 |
| (S61, t) → S17 | S61 → t S17  S61 → t |
| (S62, e) → S63 | S62 → e S63 |
| (S63, r) → S64 | S63 → e S64 |
| (S64, n) → S17 | S64 → n S17  S64 → n |
| (S65, a) → S66 | S65 → a S66 |
| (S65, l) → S69 | S65 → l S69 |
| (S65, o) → S72 | S65 → o S72 |
| (S65, r) → S73 | S65 → r S73 |
| (S66, l) → S67 | S66 → l S67 |
| (S67, s) → S68 | S67 → s S68 |
| (S68, e) → S17 | S68 → e S17  S68 → e |
| (S69, o) → S70 | S69 → o S70 |
| (S70, a) → S71 | S70 → a S71 |
| (S71, t) → S17 | S71 → t S17  S71 → t |
| (S72, r) → S17 | S72 → r S17  S72 → r |
| (S73, i) → S74 | S73 → i S74 |
| (S74, e) → S75 | S74 → e S75 |
| (S75, n) → S76 | S75 → n S76 |
| (S76, d) → S17 | S76 → d S17  S76 → d |
| (S77, o) → S78 | S77 → o S78 |

Продолжение табл. 2

|  |  |
| --- | --- |
| (S78, t) → S79 | S78 → t S79 |
| (S79, o) → S17 | S79 → o S17  S79 → o |
| (S80, f) → S17 | S80 → f S17  S80 → f |
| (S80, n) → S81 | S80 → n S81 |
| (S81, l) → S82 | S81 → l S82 |
| (S82, t) → S17 | S82 → t S17  S82 → t |
| (S82, i) → S83 | S82 → i S83 |
| (S83, n) → S84 | S83 → n S84 |
| (S84, e) → S17 | S84 → e S17  S84 → e |
| (S85, o) → S86 | S85 → e S86 |
| (S86, n) → S87 | S86 → n S87 |
| (S87, g) → S17 | S87 → g S17  S87 → g |
| (S88, u) → S89 | S88 → u S89 |
| (S89, t) → S90 | S89 → t S90 |
| (S90, a) → S91 | S90 → a S91 |
| (S91, b) → S92 | S91 → b S92 |
| (S92, l) → S93 | S92 → l S93 |
| (S93, e) → S17 | S93 → e S17  S93 → e |
| (S94, a) → S95 | S94 → a S95 |
| (S94, e) → S102 | S94 → e S102 |
| (S94, u) → S103 | S94 → u S103 |
| (S95, m) → S96 | S95 → m S96 |
| (S96, a) → S97 | S96 → a S97 |
| (S97, s) → S98 | S97 → s S98 |
| (S98, p) → S99 | S98 → p S99 |
| (S99, a) → S100 | S99 → a S100 |
| (S100, c) → S101 | S100 → c S101 |
| (S101, e) → S17 | S101 → e S17  S101 → e |
| (S102, w) → S17 | S102 → w S17  S102 → w |
| (S103, l) → S104 | S103 → l S104 |
| (S104, l) → S105 | S104 → l S105 |
| (S105, p) → S106 | S105 → p S106 |
| (S106, t) → S107 | S106 → t S107 |
| (S107, r) → S17 | S107 → r S17  S107 → r |
| (S108, p) → S109 | S108 → p S109  S108 → p |
| (S109, e) → S110 | S109 → e S110  S109 → e |
| (S110, r) → S111 | S110 → r S111  S110 → r |

Продолжение табл. 2

|  |  |
| --- | --- |
| (S111, a) → S112 | S111 → a S112  S111 → a |
| (S112, t) → S113 | S112 → t S113 |
| (S113, o) → S114 | S113 → o S114 |
| (S114, r) → S17 | S114 → r S17  S114 → r |
| (S115, r) → S116 | S115 → r S116 |
| (S115, u) → S127 | S115 → u S127 |
| (S116, i) → S117 | S116 → i S117 |
| (S117, v) → S118 | S117 → v S117 |
| (S118, a) → S119 | S118 → a S119 |
| (S119, t) → S120 | S119 → t S120 |
| (S120, e) → S17 | S120 → e S17  S120 → e |
| (S121, t) → S122 | S121 → t S122 |
| (S122, e) → S123 | S122 → e S123 |
| (S123, c) → S124 | S123 → c S124 |
| (S124, t) → S125 | S124 → t S125 |
| (S125, e) → S126 | S125 → e S126 |
| (S126, d) → S127 | S126 → d S127 |
| (S127, b) → S128 | S127 → b S128 |
| (S128, l) → S129 | S128 → l S129 |
| (S129, i) → S130 | S129 → i S130 |
| (S130, c) → S17 | S130 → c S17  S130 → c |
| (S131, e) → S132 | S131 → e S132 |
| (S132, t) → S133 | S132 → t S133 |
| (S133, u) → S134 | S133 → u S134 |
| (S134, r) → S135 | S134 → r S135 |
| (S135, n) → S17 | S135 → n S17  S135 → n |
| (S136, h) → S137 | S136 → h S137 |
| (S136, i) → S140 | S136 → i S140 |
| (S136, t) → S147 | S136 → t S147 |
| (S136, w) → S154 | S136 → w S154 |
| (S137, o) → S138 | S137 → o S138 |
| (S138, r) → S139 | S138 → r S139 |
| (S139, i) → S17 | S139 → i S17  S139 → i |
| (S140, g) → S141 | S140 → g S141 |
| (S140, z) → S144 | S140 → z S144 |
| (S141, n) → S142 | S141 → n S142 |
| (S142, e) → S143 | S142 → e S143 |
| (S143, d) → S17 | S143 → d S17  S143 → d |
| (S144, e) → S145 | S144 → e S145 |
| (S145, o) → S146 | S145 → o S146 |

Продолжение табл. 2

|  |  |
| --- | --- |
| (S146, f) → S17 | S146 → f S17  S146 → f |
| (S147, a) → S148 | S147 → a S148 |
| (S147, r) → S151 | S147 → r S151 |
| (S148, t) → S149 | S148 → t S149 |
| (S149, i) → S150 | S149 → i S150 |
| (S150, c) → S17 | S150 → c S17  S150 → c |
| (S151, u) → S152 | S151 → u S152 |
| (S152, с) → S153 | S152 → c S153 |
| (S153, t) → S17 | S153 → t S17  S153 → t |
| (S154, i) → S155 | S154 → i S155 |
| (S155, t) → S156 | S155 → t S156 |
| (S156, c) → S157 | S156 → c S157 |
| (S157, h) → S17 | S157 → h S17  S157 → h |
| (S158, e) → S159 | S158 → e S159 |
| (S158, h) → S165 | S158 → h S165 |
| (S158, r) → S169 | S158 → r S169 |
| (S159, m) → S160 | S159 → m S160 |
| (S160, p) → S161 | S160 → p S161 |
| (S161, l) → S162 | S161 → l S162 |
| (S162, a) → S163 | S162 → a S163 |
| (S163, t) → S164 | S163 → t S164 |
| (S164, e) → S17 | S164 → e S17  S164 → e |
| (S165, i) → S166 | S165 → i S166 |
| (S165, r) → S167 | S165 → r S167 |
| (S166, s) → S17 | S166 → s S17  S166 → s |
| (S167, o) → S168 | S167 → o S168 |
| (S168, w) → S17 | S168 → w S17  S168 → w |
| (S169, u) → S170 | S169 → u S170 |
| (S169, y) → S17 | S169 → y S17  S169 → y |
| (S170, e) → S17 | S170 → e S17  S170 → e |
| (S171, n) → S172 | S171 → n S172 |
| (S171, s) → S180 | S171 → s S180 |
| (S172, i) → S173 | S172 → i S173 |
| (S172, s) → S175 | S172 → s S175 |
| (S173, o) → S174 | S173 → o S174 |
| (S174, n) → S17 | S174 → n S17  S174 → n |
| (S175, i) → S176 | S175 → i S176 |
| (S176, g) → S177 | S176 → g S177 |
| (S177, n) → S178 | S177 → n S178 |

Продолжение табл. 2

|  |  |
| --- | --- |
| (S178, e) → S179 | S178 → e S179 |
| (S179, d) → S17 | S179 → d S17  S179 → d |
| (S180, i) → S181 | S180 → i S181 |
| (S181, n) → S182 | S181 → n S182 |
| (S182, g) → S17 | S182 → g S17  S182 → g |
| (S183, i) → S184 | S183 → i S184 |
| (S183, o) → S189 | S183 → o S189 |
| (S184, r) → S185 | S184 → r S185 |
| (S185, t) → S186 | S185 → t S186 |
| (S186, u) → S187 | S186 → u S187 |
| (S187, a) → S188 | S187 → a S188 |
| (S188, l) → S17 | S188 → l S17  S188 → l |
| (S189, i) → S190 | S189 → i S190 |
| (S190, d) → S17 | S190 → d S17  S190 → d |
| (S191, h) → S192 | S191 → h S192 |
| (S192, i) → S193 | S192 → i S193 |
| (S193, l) → S194 | S193 → l S194 |
| (S194, e) → S17 | S194 → e S17 |

## Построение КС-грамматики, порождения и дерева вывода

В основе синтаксического анализатора лежит распознаватель, построенный на основе класса контекстно-свободных грамматик (КС-грамматик). КС-грамматика является частным случаем формальной грамматики по иерархии Хомского и используется для описания грамматической структуры многих языков программирования [10].

Язык, определенный с использованием КС-грамматики, может быть представлен либо построением дерева вывода, либо порождением. При порождении, правила грамматики определяются как правила, где нетерминал из левой части заменяется строкой из правой части продукции. Из стартового символа грамматики выводится строка, содержащая терминальные символы, которая называется предложением грамматики.

Упорядоченное помеченное дерево D называется деревом вывода цепочки в КС-грамматике G = (N, T, P, S), если выполнены следующие условия:

1. Корень дерева D помечен как S;
2. Каждый лист помечен либо как a ∈ T, либо как e;
3. Каждая внутренняя вершина помечена как нетерминал A ∈ N;
4. Если X – это нетерминал, которым помечена внутренняя вершина и X1, ..., Xn – метки ее прямых потомков в указанном порядке, то X → X1...Xk - правило из множества P;
5. Цепочка, составленная из выписанных слева направо меток листьев, равна w.

Существуют две основные стратегии синтаксического анализа:

1. Нисходящий анализ, где дерево вывода строится от корня к листьям, и аксиома грамматики развертывается в цепочку.
2. Восходящий анализ, где дерево вывода строится от листьев к корню, и анализируемая цепочка сворачивается в аксиому.

Каждому выводу в КС-грамматике, начинающемуся с нетерминального символа, соответствует ориентированный граф, который является деревом вывода. Это дерево вывода связано не с грамматикой, а с конкретным выводом в данной грамматике.

Построение КС-грамматики для заданного варианта задания:

Код для построения КС-грамматики:

switch (a) {

case 0:

if (b != 0) {

c = a + b;

cout << c << endl;

}

else c = a - b;

break;

case 1:

if (b >= 0) {

c = a++;

cout << c << endl;

}

else c = b--;

break;

case 2:

if (b > 0) {

c = a \* b;

cout << c << endl;

}

else c = a / b;

break;

case 3:

if (b < 0) {

a += b;

cout << a << endl;

}

else b -= a;

c = a + b;

break;

}

Описание КС-грамматики:

G (T, N, P, S), где

T — набор (алфавит) терминальных символов

N — набор (алфавит) нетерминальных символов

P — набор правил вида: «левая часть» -> «правая часть», где:

«левая часть» — непустая последовательность терминалов и нетерминалов, содержащая хотя бы один нетерминал

«правая часть» — любая последовательность терминалов и нетерминалов

S — стартовый (или начальный) символ грамматики из набора нетерминалов.

T = {switch, case, if, cout, endl, else, break, const, id, {, }, (, ), :, ;, !=, =, +, <<, –, >=, ++, – –, >, \*, /, <, +=, –=}

N = {S, W, N, C, B, X, I, E, A, P, Y, V, D, F}

P:

S -> W

W -> switch(N){C}

N -> id; | id | const | const; | id: | const:

C -> case NBC | case NB

B -> I | A; B | P; | X

X -> break;

I -> if (F) {B} E

E -> else B

A -> NZ F | D | N

P -> VYNYV

Y -> <<

V -> cout | endl

D -> NZ

F -> NZN

Z -> != | >= | > | < | = | + | – | + | / | += | –= | ++ | – –

S = {S}

Левостороннее порождение:

S -> **W**

S -> switch(**N**) {C}

S -> switch(id) {**C**}

S -> switch(id) { case **N**BC }

S -> switch(id) { case const: **B**C }

S -> switch(id) { case const: **I**C }

S -> switch(id) { case const: if (**F**) { B } EC }

S -> switch(id) { case const: if (**N**ZN) { B } EC }

S -> switch(id) { case const: if (id **Z**N) { B } EC }

S -> switch(id) { case const: if (id != **N**) { B } EC }

S -> switch(id) { case const: if (id != const) { **B** } EC }

S -> switch(id) { case const: if (id != const) { **A**; B } EC }

S -> switch(id) { case const: if (id != const) { **N**ZF; B } EC }

S -> switch(id) { case const: if (id != const) { id **Z**F; B } EC }

S -> switch(id) { case const: if (id != const) { id = **F**; B } EC }

S -> switch(id) { case const: if (id != const) { id = **N**ZN; B } EC }

S -> switch(id) { case const: if (id != const) { id = id **Z**N; B } EC }

S -> switch(id) { case const: if (id != const) { id = id + **N**; B } EC }

S -> switch(id) { case const: if (id != const) { id = id + id; **B** } EC }

S -> switch(id) { case const: if (id != const) { id = id + id; **P**; } EC }

S -> switch(id) { case const: if (id != const) { id = id + id; **V**YNYV; } EC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout **Y**NYV; } EC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << **N**YV; } EC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id **Y**V; } EC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << **V**; } EC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } **E**C }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else **B**C }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else **A**; BC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else **N**ZF; BC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id **Z**F; BC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = **F**; BC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = **N**ZN; BC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id **Z**N; BC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – **N**; BC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; **B**C }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; **X**C }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; **C** }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case **N**BC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: **B**C }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: **I**C }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (**F**) { B } EC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (**N**ZN) { B } EC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id **Z**N) { B } EC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= **N**) { B } EC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { **B** } EC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { **A**; B } EC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { **N**ZD; B } EC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { **N**ZD; B } EC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id **Z**D; B } EC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = **D**; B } EC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = **N**Z; B } EC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id **Z**; B } EC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; **B** } EC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; **P** } EC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; **V**YNYV; } EC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout **Y**NYV; } EC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << **N**YV; } EC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id **Y**V; } EC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << **V**; } EC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } **E**C }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else **B**C }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else **A**; BC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else **N**ZD; BC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id **Z**D; BC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id = **D**; BC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id = **N**Z; BC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id Z; BC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; **B**C }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; **X**C }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; **C** }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case **N**BC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const: **B**C }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const: **I**C }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (**F**) { B } EC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (**N**ZN) { B } EC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id **Z**N) { B } EC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > **N**) { B } EC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { **B** } EC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { **A**; B } EC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { **N**ZF; B } EC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id **Z**F; B } EC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = **F**; B } EC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = **N**ZN; B } EC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id **Z**N; B } EC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* **N**; B } EC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; **B** } EC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; **P** } EC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; **V**YNYV; } EC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout **Y**NYV; } EC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << **N**YV; } EC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id **Y**V; } EC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << **V**; } EC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } **E**C }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else **A**; BC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else **N**ZF; BC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id **Z**F; BC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = **F**; BC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = **N**ZN; BC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = id **Z**N; BC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = id / **N**; BC }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = id / id; **B**C }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = id / id; **X**C }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = id / id; break; **C** }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = id / id; break; **C** }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = id / id; break; case **N**B }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = id / id; break; case **N**B }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = id / id; break; case const: **B** }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = id / id; break; case const: **I** }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = id / id; break; case const:if (**F**) { B } E }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = id / id; break; case const:if (**N**ZN) { B } E }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = id / id; break; case const:if (**N**ZN) { B } E }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = id / id; break; case const:if (id **Z**N) { B } E }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = id / id; break; case const:if (id < N) { B } E }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = id / id; break; case const:if (id < **N**) { B } E }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = id / id; break; case const:if (id < const) { **B** } E }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = id / id; break; case const:if (id < const) { **A**; B } E }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = id / id; break; case const:if (id < const) { **N**ZN; B } E }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = id / id; break; case const:if (id < const) { id **Z**N; B } E }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = id / id; break; case const:if (id < const) { id += **N**; B } E }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = id / id; break; case const:if (id < const) { id += **N**; B } E }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = id / id; break; case const:if (id < const) { id += id; **B** } E }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = id / id; break; case const:if (id < const) { id += id; **P** } E }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = id / id; break; case const:if (id < const) { id += id; **V**YNYV } E }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = id / id; break; case const:if (id < const) { id += id; cout **Y**NYV } E }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = id / id; break; case const:if (id < const) { id += id; cout << **N**YV } E }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = id / id; break; case const:if (id < const) { id += id; cout << id **Y**V } E }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = id / id; break; case const:if (id < const) { id += id; cout << id << **V** } E }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = id / id; break; case const:if (id < const) { id += id; cout << id << endl } **E** }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = id / id; break; case const:if (id < const) { id += id; cout << id << endl } else **A**; B }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = id / id; break; case const:if (id < const) { id += id; cout << id << endl } else **A**; B }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = id / id; break; case const:if (id < const) { id += id; cout << id << endl } else **N**ZN; B }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = id / id; break; case const:if (id < const) { id += id; cout << id << endl } else id **Z**N; B }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = id / id; break; case const:if (id < const) { id += id; cout << id << endl } else id –= N; **B** }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = id / id; break; case const:if (id < const) { id += id; cout << id << endl } else id –= N; **A**; B }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = id / id; break; case const:if (id < const) { id += id; cout << id << endl } else id –= N; **N**ZF; B }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = id / id; break; case const:if (id < const) { id += id; cout << id << endl } else id –= N; id **Z**F; B }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = id / id; break; case const:if (id < const) { id += id; cout << id << endl } else id –= N; id= **F**; B }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = id / id; break; case const:if (id < const) { id += id; cout << id << endl } else id –= N; id= **N**ZN; B }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = id / id; break; case const:if (id < const) { id += id; cout << id << endl } else id –= N; id= id **Z**N; B }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = id / id; break; case const:if (id < const) { id += id; cout << id << endl } else id –= N; id= id + **N**; B }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = id / id; break; case const:if (id < const) { id += id; cout << id << endl } else id –= N; id= id + id; **B** }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = id / id; break; case const:if (id < const) { id += id; cout << id << endl } else id –= N; id= id + id; **X** }

S -> switch(id) { case const: if (id != const) { id = id + id; cout << id << endl; } else id = id – id; break; case const: if (id >= const) { id = id**++**; cout << id << endl; } else id =id– –; break; case const:if (id > const) { id = id \* id; cout << id << endl; } else id = id / id; break; case const:if (id < const) { id += id; cout << id << endl } else id –= N; id= id + id; break; }

На рис. 6 представлено нисходящее дерево вывода по заданной грамматике.



Рис. 6. Дерево вывода

# Логическое проектирование

## 3.1. Блок-схемы

С помощью блок-схем можно рассмотреть работу алгоритма всей программы или некоторых ее функций. Ниже представлена общая блок-схема алгоритма программы (рис. 7).

Блок-схема первого автомата, отвечающего за очистку кода от лишних символов и комментариев, представлена на рис. 8.1-8.5.

Фрагмент блок-схемы второго автомата, отвечающего за распознавание лексем, вывод их в таблицы и формирование дескрипторного кода и псевдокода, представлен на рис. 9.1-9.7. Пропущенные элементы работают аналогично тем, что представлены на схеме.



Рис. 7. Общая блок-схема алгоритма программы



Рис. 8.1. Блок-схема первого автомата для состояний 0-2



Рис. 8.2. Блок-схема первого автомата для состояний 3-5



Рис. 8.3. Блок-схема первого автомата для состояний 6-9



Рис. 8.4. Блок-схема первого автомата для состояний 10-11



Рис. 8.5. Блок-схема первого автомата: обработка ошибки незакрытого комментария



Рис. 9.1. Блок-схема второго автомата для состояний 0-3



Рис. 9.2. Блок-схема второго автомата для состояний 4-7



Рис. 9.3. Блок-схема второго автомата для состояний 8-11



Рис. 9.4. Блок-схема второго автомата для состояний 12-15



Рис. 9.5. Блок-схема второго автомата для состояний 16-194, 999



58

Рис. 9.6. Блок-схема второго автомата: вывод в таблицы 1-3



59

Рис. 9.7. Блок-схема второго автомата: вывод в таблицы 4-5

## 3.2. Словесное описание алгоритма

1. Программа просит пользователя ввести код для обработки. Это можно сделать 2 вариантами: либо ввести его вручную в текстовом блоке, либо загрузив его из текстового файла.

2. Код записывается в строку. Затем программа посимвольно считывает эту строку и подает каждый символ на вход первого автомата, который очищает код от комментариев и лишних пробелов.

3. Автомат реализован с помощью оператора switch. Он меняет свои состояния в зависимости от полученного символа и значения предыдущего состояния.

4. Символы из исходной строки записываются в другую строку с обработанным кодом. При обнаружении 2 и более пробелов подряд, пустых строк (т. е. строк, содержащих только символ переноса строки) или комментариев (т. е. любых символов, идущих после символов «//» и до символа переноса строки, либо после символов «/\*» и до символов «\*/»), они в строку не записываются.

5. При обнаружении символа переноса строки увеличивается счетчик строк (кроме случаев, когда перенос строки встречается в многострочном комментарии).

6. При достижении конца строки с кодом очищенный код и количество строк выводятся в соответствующие текстовые блоки.

7. Получившаяся строка с очищенным кодом снова считывается посимвольно и каждый символ подается на вход второго автомата, который распознает лексемы.

8. Автомат реализован с помощью оператора switch. Он меняет свои состояния в зависимости от полученного символа и значения предыдущего состояния.

9. Программа записывает в строку каждый символ распознаваемой лексемы.

10. Как только при распознавании лексемы встретится символ, являющийся ее границей, значение строки заносится в одну из пяти таблиц, соответствующих типу распознанной лексемы. Также в таблицу заносится соответствующий лексеме псевдокод и порядковый номер распознанной лексемы (для формирования дескрипторного кода) – значение счетчика, соответствующего типу лексемы. Затем значение этого счетчика увеличивается на 1.

11. После добавления новой лексемы в одну из пяти таблиц, соответствующие им идентификаторы в виде (номер таблицы; номер лексемы) записываются в текстовый блок с дескрипторным кодом, а соответствующие им псевдокоды – в текстовый блок с псевдокодом.

12. Если при распознавании лексемы будет найден ошибочный символ, эта лексема будет занесена в отдельную таблицу с ошибочными лексемами. Также в таблице будет указан номер строки и тип ошибки. В текстовые блоки с дескрипторным и псевдокодом в этом случае ничего не заносится.

13. После добавления лексемы в таблицу, строка в которую эта лексема записывалась, очищается для дальнейшего распознавания лексем.

## 3.3. Оценка сложности алгоритма

Сложность алгоритма или вычислительная сложность – это зависимость объема работы, необходимого для выполнения алгоритма, от размера входных данных. Размер самой программы обычно имеет незначительное влияние на производительность её исполнения. Многие структуры данных позволяют значительно ускорить работу по сравнению с более простыми реализациями [11].

Сложность данной реализации лексического анализатора, при выборе в качестве числа n количества символов в исходном файле, будет равна O(n), поскольку данная характеристика опускает незначительные коэффициенты рядом с n. Многие символы будут отброшены после этапа предобработки, что может заметно ускорить обработку важной части программы. Более медленной частью программы теоретически является поиск лексем в таблице (сложность O(n)), реализованный полным перебором. Однако в реальном сценарии маловероятно, что каждый символ будет выделен в отдельную лексему. Кроме того, количество лексем, кроме идентификаторов, ограничено и невелико.

На практике самой медленной частью программы является обработка ошибочных лексем. Программа с наличием таких лексем обрабатывается медленнее исправленной версии такой же программы.

# Физическое проектирование

## Выбор структур данных

Спецификация структур данных, использованных в программе, отображена в табл. 3.

Таблица 3

Структуры данных

|  |  |  |
| --- | --- | --- |
| Обозначение | Тип данных | Назначение |
| prog | String^ | Строка, в которой содержится код программы после обработки в первом автомате |
| id | String^ | Строка, содержащая номер лексемы (для вывода дескрипторного кода) |
| psevdo | String^ | Строка, содержащая псевдокод программы |
| count\_r | int | Счетчик разделителей |
| count\_z | int | Счетчик знаков операций |
| count\_const | int | Счетчик констант |
| count\_key | int | Счетчик ключевых слов |
| count\_id | int | Счетчик идентификаторов |
| count | int | Счетчик строк (для вывода кода на экран) |
| count\_str | int | Счетчик строк (для вывода номера строки с ошибкой) |
| c | char | Символ, подаваемый на вход автомата |
| str1 | String^ | Строка, в которую записывается обработанный код во время работы первого автомата |
| str00 | String^ | Строка, содержащая исходный код, введенный пользователем |
| str | String^ | Строка, в которую записывается распознанная лексема |
| str0 | string | Строка str00, преобразованная в другой тип данных |
| s | int | Номер состояния |
| symbol | String^ | Строка, в которую записываются лексемы для вывода в таблицы |

Продолжение табл. 3

|  |  |  |
| --- | --- | --- |
| count | String^ | Строка, содержащая номер лексемы (для вывода в таблицы) |
| pk | String^ | Строка, содержащая псевдокод лексемы (для вывода в таблицы) |
| f | bool | Флаг, подтверждающий окончание лексемы |
| err | bool | Флаг, подтверждающий нахождение ошибочного символа |
| i | int | Счетчик символов во входной строке автомата |

## Выбор функций

Спецификация функций, использованных в программе, отображена в табл. 4.

Таблица 4

Спецификация функций

|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Тип | Формальные параметры | Назначение |
| Automat1 | int | char c, int s | Автомат, очищающий код от лишних символов и комментариев |
| Automat2 | void | char c, int& s, string& str, int& i, bool& f, bool& err | Автомат, распознающий лексемы |
| search | bool | DataGridView^ dataGridView, String^ symbol | Поиск распознанной лексемы в таблице (чтобы не вносить ее повторно) |
| button1\_Click | System::Void | System::Object^ sender, System::EventArgs^ e | Запуск первого автомата при нажатии кнопки |
| button2\_Click | System::Void | System::Object^ sender, System::EventArgs^ e | Вызов диалогового окна выбора файла при нажатии кнопки |
| button3\_Click | System::Void | System::Object^ sender, System::EventArgs^ e | Запуск второго автомата при нажатии кнопки |
| button4\_Click | System::Void | System::Object^ sender, System::EventArgs^ e | Очистка всех текстовых блоков и таблиц при нажатии кнопки |

# Проектирование интерфейса

Интерфейс программы представлен на рис. 10.

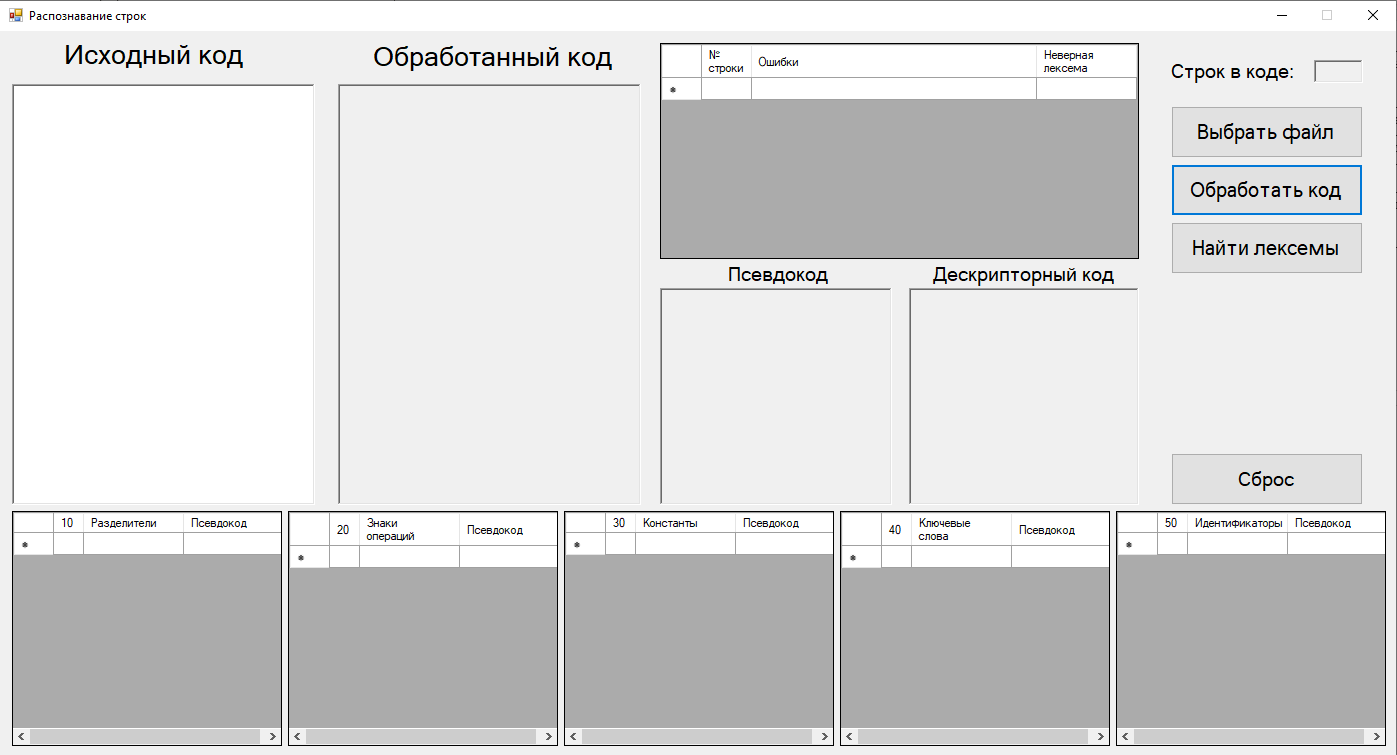


Рис. 10. Интерфейс программы

Интерфейс программы состоит из 5 текстовых блоков:

* Исходный код;
* Обработанный код;
* Псевдокод;
* Дескрипторный код;
* Количество строк в коде.

Также интерфейс программы содержит 6 таблиц:

* Таблица разделителей;
* Таблица знаков операций;
* Таблица констант;
* Таблица ключевых слов;
* Таблица идентификаторов;
* Таблица ошибок.

Для работы программы установлены 4 кнопки:

* «Выбрать файл»;
* «Обработать код»;
* «Найти лексемы»;
* «Сброс».

# Тестирование программного обеспечения

На этапе тестирования были проверены все возможные варианты кодов и ошибок в них.

1. Проверка удаления комментариев и лишних символов, а также счетчика строк (рис. 11-12):

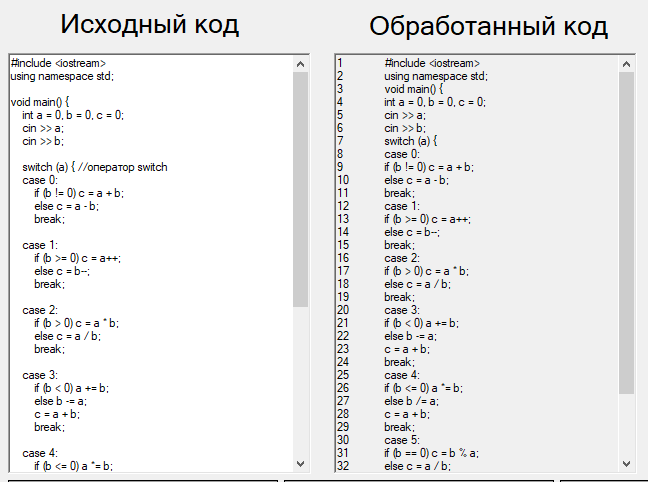


Рис. 11. Проверка удаления комментариев и лишних символов



Рис. 12. Вывод количества строк

Результат: все лишние символы и комментарии были удалены. Количество строк подсчитано верно.

1. Проверка занесения лексем в соответствующие таблицы (рис. 13-14):

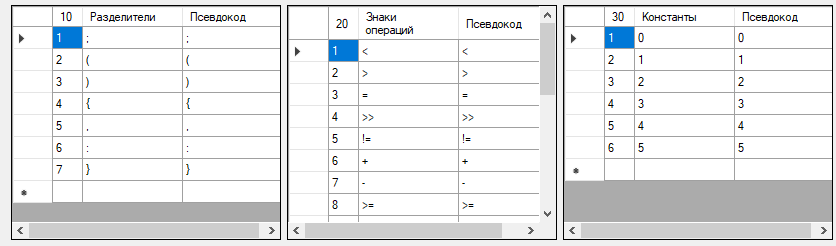


Рис. 13. Занесение лексем (разделителей, знаков операций и констант) в таблицы

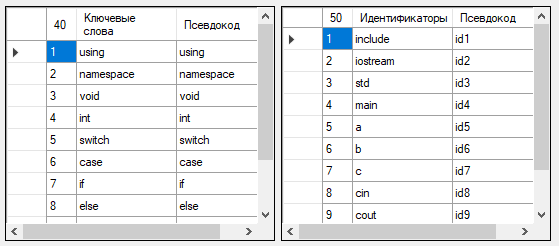


Рис. 14. Занесение лексем (ключевых слов и идентификаторов) в таблицы

Результат: все лексемы распознаны верно и занесены в нужные таблицы.

1. Проверка правильности формирования дескрипторного кода и псевдокода (рис. 15):

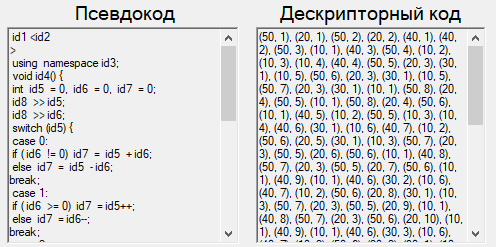


Рис. 15. Вывод дескрипторного и псевдокода

Результат: дескрипторный код и псевдокод выведены верно.

1. Проверка распознавания ошибочных лексем (рис. 16):

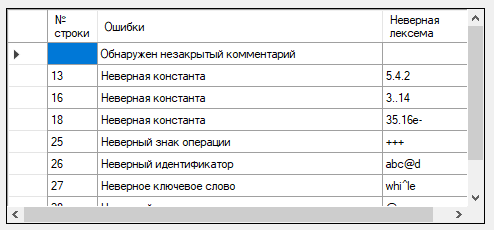


Рис. 16. Распознавание ошибочных лексем

Результат: ошибочные лексемы обнаружены верно и выведены в отдельную таблицу.

Результаты тестирования программы приведены в таблице 5

Таблица 5

Результаты тестирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Дата тестирования | Тестируемый модуль | Тестирование проводил | Описание теста | Результат тестирования |
| 08.05.2024 | Analizator.exe | Маркелов С. А. | Проверка корректного удаления лишних символов и комментариев, а также проверка корректного подсчета строк | Успех |
| 15.05.2024 | Analizator.exe | Маркелов С. А. | Проверка корректного распознания лексем и занесения их в таблицы | Успех |
| 22.05.2024 | Analizator.exe | Маркелов С. А. | Проверка корректного формирования дескрипторного кода и псевдокода | Успех |
| 29.05.2024 | Analizator.exe | Маркелов С. А. | Проверка корректного распознавания ошибочных лексем и занесения их в отдельную таблицу | Успех |

# Заключение

В ходе выполнения курсовой работы была разработана программа, которая выполняет лексический анализ введенного кода программы: очищает код от лишних символов и комментариев, подсчитывает количество строк, распознает лексемы, записывает их в таблицы лексем, а также формирует дескрипторный код и псевдокод.

Также была разработана регулярная грамматика для лексического анализатора и КС-грамматика для синтаксического анализатора, было построено дерево вывода и разобрано левостороннее порождение.

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

# Список литературы

1. Методика и организация самостоятельной работы студентов: учебно-методическое пособие / Е.В. Ершов, Л.Н. Виноградова, В.В. Селивановских [и др.]. – Череповец: ФГБОУ ВПО ЧГУ, 2015. – 243 с.
2. Ганичева, О. Г. Теория языков программирования и методы трансляции: Учеб. Пособие. – Череповец: ГОУ ВПО ЧГУ, 2011. – 186 с.
3. Карпов, Ю. Г. Теория автоматов. – СПб.: Питер, 2003. – 208 с.
4. Карпов, Ю. Г. Теория и технология программирования. Основы построения трансляторов. – СПб.: БХВ-Петербург, 2005. – 272 с.
5. Язык программирования — Википедия [электр.ресурс] <https://ru.wikipedia.org/wiki/Язык_программирования>. Дата обращения: 12.06.2024.
6. Компилятор — Википедия [электр.ресурс] https://ru.wikipedia.org/wiki/Компилятор. Дата обращения: 12.06.2024.
7. Лексический анализ — Википедия [электр.ресурс] https://ru.wikipedia.org/wiki/Лексический\_анализ. Дата обращения: 12.06.2024.
8. Синтаксический анализ — Википедия [электр.ресурс] https://ru.wikipedia.org/wiki/Синтаксический\_анализ. Дата обращения: 12.06.2024.
9. Иерархия Хомского формальных грамматик — ИТМО [электр.ресурс] https://neerc.ifmo.ru/wiki/index.php?title=Иерархия\_Хомского\_формальных\_грамматик. Дата обращения: 12.06.2024.
10. Контекстно-свободные грамматики, вывод, лево- и правосторонний вывод, дерево разбора — ИТМО [электр.ресурс] https://neerc.ifmo.ru/wiki/index.php?title=Контекстно-свободные\_грамматики,\_вы вод,\_лево-\_и\_правосторонний\_вывод,\_дерево\_разбора. Дата обращения: 12.06.2024.
11. Вычислительная сложность — Википедия [электр.ресурс] https://ru.wikipedia.org/wiki/Вычислительная\_сложность. Дата обращения: 12.06.2024.

# Приложение 1

МИНОБРАНАУКИ РОССИИ

федеральное государственное бюджетное

образовательное учреждение высшего образования

ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Институт информационных технологий

наименование института (факультета)

Математическое и программное обеспечение ЭВМ

наименование кафедры

Модуль: Информатика

наименование дисциплины в соответствии с учебным планом

УТВЕРЖДАЮ

Зав. кафедрой МПО ЭВМ

д. т.н. \_\_\_\_\_\_\_\_\_ Ершов Е.В.

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_2024 г.

Разработка алгоритмического обеспечения и построение лексического анализатора компилятора

Техническое задание на курсовую работу

Листов 8

Руководители: Ганичева О. Г.,

Пышницкий К. М.

Исполнитель: студент гр. 1ПИб-02-3оп-22

Маркелов С. А.

2024 г.

Введение

Курсовая работа направлена на разработку алгоритмического обеспечения и построение лексического анализатора, которое позволит пользователю наглядно увидеть, какие лексемы и каких типов имеются в коде, и понять часть принципа работы компилятора.

1. Основания для разработки

Основанием для разработки является задание на курсовую работу по модулю «Информатика», выданное на кафедре МПО ЭВМ.

Дата утверждения: 14 февраля 2024 года.

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

2. Назначение разработки

Назначением разработки является создание программы, моделирующей работу лексического анализатора и решающей следующие задачи:

* выделение из текста входной программы всех лексем, входящих в языковую конструкцию;
* удаление лишних пробелов и комментариев из входной строки;
* подсчёт количества строк в программе;
* построение дескрипторного кода и псевдокода.

3. Требования к программе

* 1. Требования к функциональным характеристикам

К разрабатываемой программе предъявляются следующие требования:

* модель лексического анализатора должна обрабатывать введенный пользователем текст программы и удалять из него лишние пробелы, знаки табуляции, однострочные и многострочные комментарии, излишние переходы на новую строку;
* модель лексического анализатора должна выполнять подсчет количества строк в программе;
* модель лексического анализатора должна выделять из текста программы все лексемы, определять их тип и заносить их в соответствующие таблицы классов лексем;
* модель лексического анализатора должна строить и выводить на экран дескрипторный код и псевдокод;
* модель лексического анализатора должна диагностировать и локализовать лексические ошибки;
* работа модели лексического анализатора должна моделироваться с помощью конечного автомата, граф переходов которого должен быть представлен в расчетно-пояснительной записке.
  1. Требования к надежности

Для правильного функционирования программы предъявляются следующие требования:

* входной файл должен быть текстового формата;
* входной файл должен иметь кодировку символов ASCII (или совместимую с ней) и содержать только ASCII-символы;
* размер свободной оперативной памяти должен быть как минимум в 2,5 раза больше размера открываемого файла без учета памяти, требуемой для запуска самой программы.
  1. Условия эксплуатации

Для корректной работы программы необходимо:

* наличие любого устройства, на котором возможно запускать файлы формата .exe (компьютер, ноутбук);
* наличие на устройстве современной версии любой операционной системы.
  1. Требования к составу и параметрам технических средств

Для корректной работы программы необходимо:

* оперативная память: не менее 1 Гб для 32-разрядной ОС, 2 Гб для 64-разрядной ОС;
* процессор не менее чем с 2 ядрами и тактовой частотой не ниже 1 ГГц;
* разрешение экрана не менее 800 х 600;
* видеокарта с видеопамятью не менее 1 Гб;
* наличие свободного места на диске – не менее 350 Мб;
* наличие монитора, мыши и клавиатуры.
  1. Требования к информационной и программной совместимости

Для корректной работы программы необходимо:

* установленная на устройстве современная операционная система (например, Windows 7, 8.1, 10, 11, macOS, Linux);
* установленный на устройстве пакет библиотек «Microsoft Visual C++».
  1. Требования к маркировке и упаковке

Требования к маркировке и упаковке программы не предъявляются.

* 1. Требования к транспортированию и хранению

Программа может храниться и транспортироваться в виде файлов, необходимых для ее работы, на носителе информации (флешка, CD-диск).

* 1. Специальные требования

Программа должна иметь простой и удобный интерфейс. От пользователя требуется минимальное умение пользоваться компьютером. Рекомендуемый возраст использования – с 12 лет.

4. Требования к программной документации

* 1. Содержание расчетно-пояснительной записки

Программная документация должна содержать расчётно-пояснительную записку с содержанием:

1. Титульный лист
2. Оглавление
3. Введение
4. Описание предметной области
5. Основная часть
6. Логическое проектирование
7. Физическое проектирование
8. Проектирование интерфейса
9. Тестирование программного обеспечения
10. Заключение
11. Список литературы
12. Приложения
    1. Требования к оформлению

В данном пункте представлены требования к оформлению документации в соответствии с ГОСТ (табл. П1.1).

Таблица П1.1

|  |  |
| --- | --- |
| Документ | Печать на отдельных листах формата А4 (210х297 мм); оборотная сторона не заполняется; листы нумеруются. Печать возможна ч/б. Файлы предъявляются на компакт-диске: РПЗ с ТЗ; программный код. Листы и диск в конверте вложены в пластиковую папку скоросшивателя. |
| Страницы | Ориентация – книжная; отдельные страницы, при необходимости, альбомная. Поля: верхнее, нижнее – по 2 см, левое – 3 см , правое – 1 см. |
| Абзацы | Межстрочный интервал – 1,5, перед и после абзаца – 0. |
| Шрифты | Кегль – 14. В таблицах шрифт 12. Шрифт листинга – 10 (возможно в 2 колонки). |
| Рисунки | Подписывается под ним по центру: Рис.Х. Название. В приложениях: Рис.П1.3. Название |
| Таблицы | Подписывается: над таблицей, выравнивание по правому: «Таблица Х». В следующей строке по центру Название Надписи в «шапке» (имена столбцов, полей) – по центру. В теле таблицы (записи) текстовые значения – выровнены по левому краю, числа, даты – по правому. |

5. Технико-экономические показатели

Требования не предъявляются.

6. Стадии и этапы разработки

В данном пункте описаны стадии и этапы разработки программы (табл. П1.2).

Таблица П1.2

Стадии и этапы разработки

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование этапа разработки | Сроки разработки | Результат выполнения | Отметка о выполнении |
| Разработка технического задания | 13.03.2024 | Готовое техническое задание |  |
| Разработка дополнительных функций лексического анализатора | 27.03.2024 | Готовые дополнительные функции лексического анализатора |  |
| Разработка функции распознавания заданных слов языка программирования | 10.04.2024 | Готовая функция распознавания заданных слов языка программирования |  |
| Разработка функции построения таблиц классов лексем | 24.04.2024 | Готовая функция построения таблиц классов лексем |  |
| Разработка функций построения дескрипторного кода и псевдокода | 08.05.2024 | Готовые функции построения дескрипторного кода и псевдокода |  |
| Разработка функции обработки ошибок на этапе лексического анализа | 15.05.2024 | Готовая функция обработки ошибок на этапе лексического анализа |  |
| Построение КС-грамматики | 22.05.2024 | Построенная КС-грамматика |  |
| Оформление расчетно-пояснительной записки | 29.05.2024 | Оформленная расчетно-пояснительная записка |  |

7. Порядок контроля и приемки

В данном пункте описан порядок контроля и приемки курсовой работы (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 | 13.03.2024 | Согласованное техническое задание |  |
| Демонстрация работы программы | 08.05.2024 | Работа программы проверена |  |
| Сдача расчетно-пояснительной записки | 29.05.2024 | Согласованная расчетно-пояснительная записка |  |
| Сдача курсовой работы | 14.06.2024 | Получение оценки за выполненную работу |  |

# Приложение 2

Руководство пользователя

1. Общие сведения о программе

Программа предназначена для моделирования работы лексического анализатора.

Программа удаляет лишние пробелы и комментарии, подсчитывает количество строк, распознает лексемы, распределяет их по таблицам (классам) лексем, формирует псевдокод и дескрипторный код, выводит лексические ошибки.

1. Описание установки

Установка программы не требуется. Достаточно лишь запустить exe-файл с программой. Сторонних программ для работы также не требуется

1. Описание запуска

Для запуска необходимо открыть файл «Analizator.exe» (рис. П2.1).

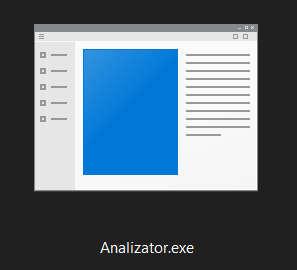


Рис. П2.1. Файл Analizator.exe

1. Инструкции по работе

В начале работы программы пользователю необходимо ввести код для дальнейшей обработки. Это можно сделать двумя способами: либо ввести с клавиатуры в текстовый блок «Исходный код» (рис. П2.2), либо загрузить код из файла, нажав на кнопку «Выбрать файл» (рис. П2.3) и затем в открывшемся диалоговом окне выбрав нужный файл (рис. П2.4). В случае с загрузкой кода из файла, текст программы вставится в блок «Исходный код» автоматически.

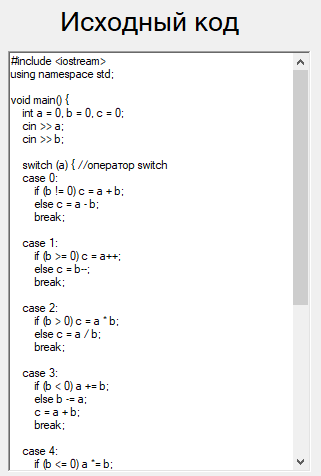


Рис. П2.2. Ввод кода в текстовый блок «Исходный код»

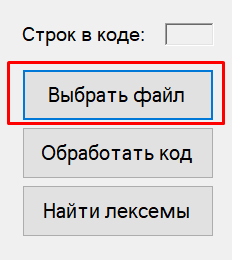


Рис. П2.3. Кнопка «Выбрать файл»

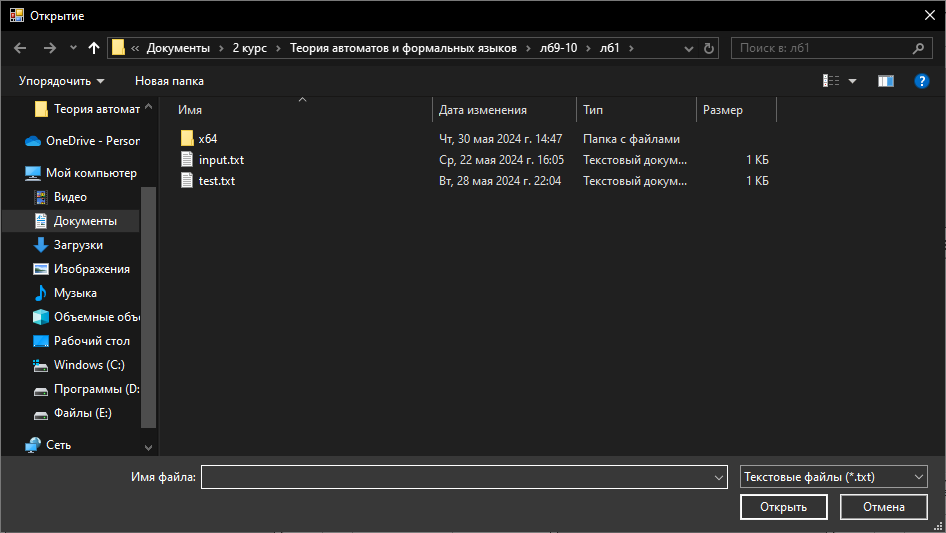


Рис. П2.4. Диалоговое окно выбора файлов

После ввода исходного кода необходимо нажать на кнопку «Обработать код» (рис. П2.5), после чего в текстовом блоке «Обработанный код» появится код, очищенный от лишних символов и комментариев (рис. П2.6), а в текстовом блоке «Строк в коде:» появится количество строк в этом коде (рис. П2.7).

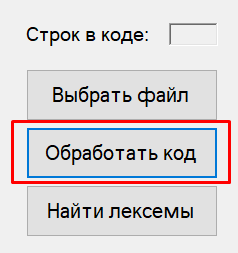


Рис. П2.5. Кнопка «Обработать код»

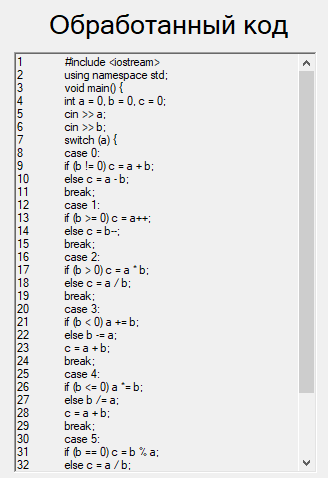


Рис. П2.6. Очищенный код в блоке «Обработанный код»



Рис. П2.7. Количество строк в блоке «Строк в коде:»

После обработки кода необходимо нажать на кнопку «Найти лексемы» (рис. П2.8), после чего в 5 таблицах в нижней части программы отобразятся все распознанные лексемы в зависимости от их класса (рис. П2.9-П2.10):

* В 1 таблице – разделители;
* Во 2 таблице – знаки операций;
* В 3 таблице – константы;
* В 4 таблице – ключевые слова;
* В 5 таблице – идентификаторы.

Сами таблицы состоят из 3 столбцов:

* 1 столбец – номера лексем;
* 2 столбец – сами лексемы;
* 3 столбец – псевдокод, соответствующий данным лексемам.

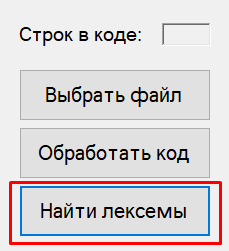


Рис. П2.8. Кнопка «Найти лексемы»

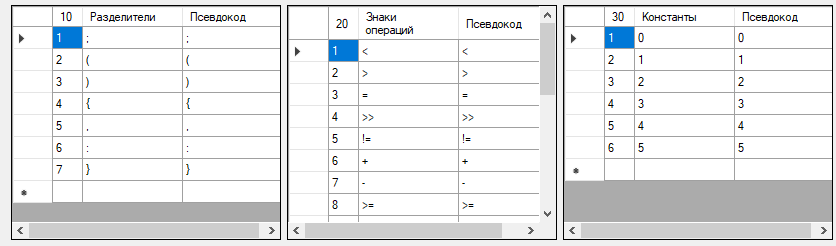


Рис. П2.9. Таблицы разделителей, знаков операций и констант

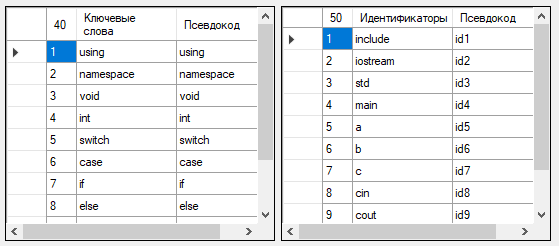


Рис. П2.10. Таблицы ключевых слов и идентификаторов

Также вместе с распознаванием лексем и записью их в таблицы будут построены дескрипторный код и псевдокод. Они будут выведены в текстовые блоки «Дескрипторный код» и «Псевдокод» (рис. П2.11).

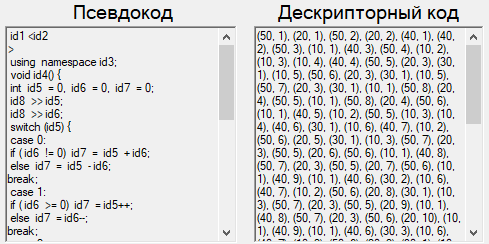


Рис. П2.11. Текстовые блоки «Псевдокод» и «Дескрипторный код»

Если в введенном пользователем коде встретятся ошибочные лексемы, они будут записаны в отдельную таблицу в верхней части программы (рис. П2.12). Она состоит из 3 столбцов:

* 1 столбец – номера строк, в которых находятся ошибочные лексемы;
* 2 столбец – Типы ошибок;
* 3 столбец – сами ошибочные лексемы.

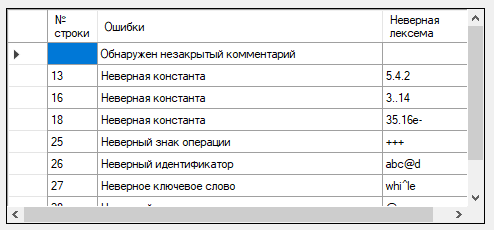


Рис. П2.12. Таблица с ошибочными лексемами

Если пользователь хочет распознать какой-либо другой код, ему необходимо нажать на кнопку «Сброс» (рис. П2.13), чтобы очистить все текстовые блоки и таблицы (рис. П2.14).

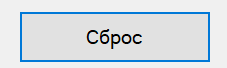


Рис. П2.13. Кнопка «Сброс»

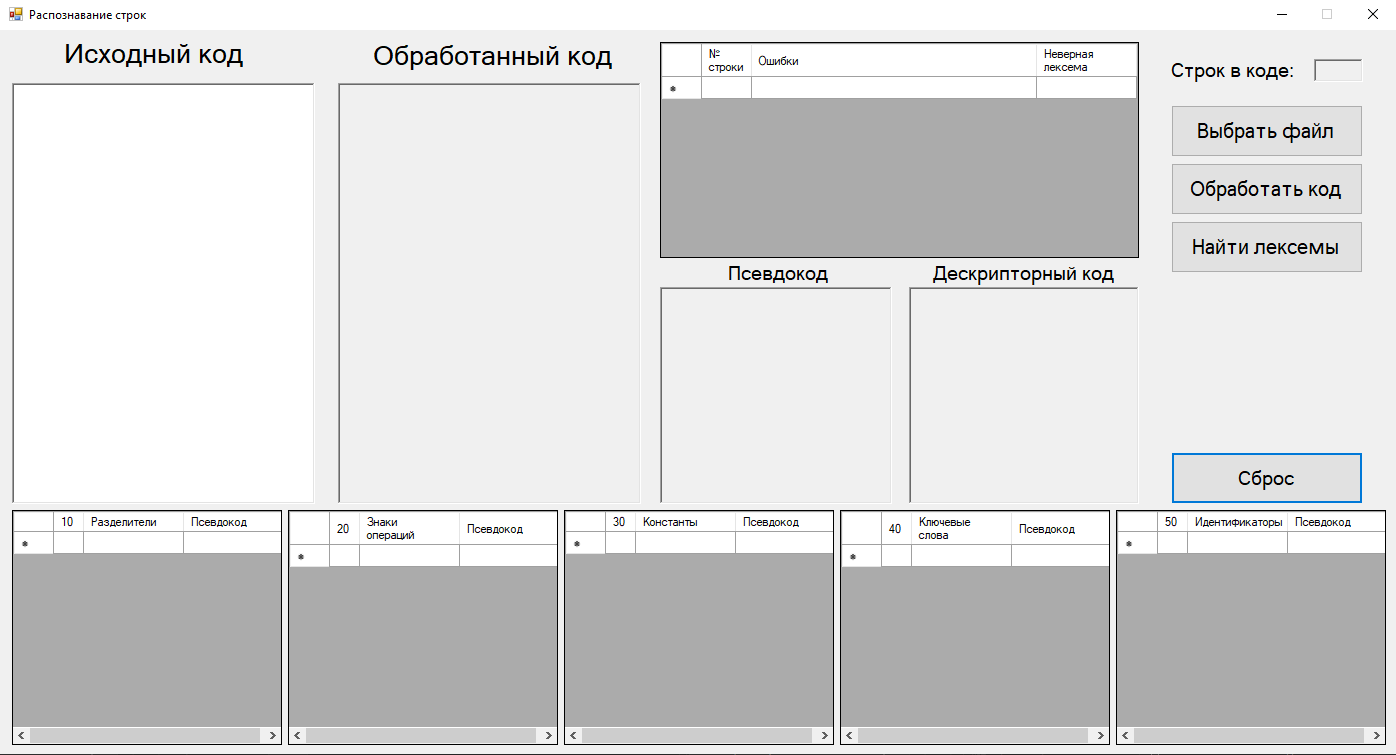


Рис. П2.14. Очищенные текстовые блоки и таблицы после сброса

1. Сообщения пользователю

Если пользователь вызовет диалоговое окно выбора файла, а затем закроет его, не выбрав файл, будет выведено сообщение «Ошибка открытия файла!» (рис. П2.14).

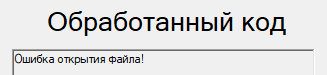


Рис. П2.14. Сообщение «Ошибка открытия файла!»

Если пользователь нажмет на кнопку «Обработать код», при этом предварительно не введя его в текстовый блок «Исходный код», будет выведено сообщение «Код не введен!» (рис. П2.15).

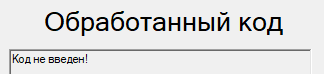


Рис. П2.15. Сообщение «Код не введен!»

# Приложение 3

Текст программы

#pragma once

#include <iostream>

#include <fstream>

#include <sstream>

#include <string>

#include <iomanip>

#include <map>

#include <stdlib.h>

#include <string.h>

#include <array>

#include <msclr\marshal.h>

#include <msclr\marshal\_cppstd.h>

#include <msclr\marshal\_windows.h>

#include <msclr\marshal\_atl.h>

namespace Analizator {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

using namespace System::IO;

using namespace System::Text;

using namespace System::Collections::Generic;

using namespace System::Runtime::InteropServices;

using namespace msclr::interop;

using namespace std;

/// <summary>

/// Сводка для MyForm

/// </summary>

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

{

public:

MyForm(void)

{

InitializeComponent();

//

//TODO: добавьте код конструктора

//

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~MyForm()

{

if (components)

{

delete components;

}

}

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

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

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

private: System::Windows::Forms::RichTextBox^ richTextBox1;

private: System::Windows::Forms::RichTextBox^ richTextBox2;

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

private: System::Windows::Forms::Button^ button2;

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

private: System::Windows::Forms::RichTextBox^ richTextBox3;

private: System::Windows::Forms::DataGridView^ dataGridView1;

private: System::Windows::Forms::Button^ button3;

private: System::Windows::Forms::DataGridView^ dataGridView2;

private: System::Windows::Forms::DataGridView^ dataGridView3;

private: System::Windows::Forms::DataGridView^ dataGridView4;

private: System::Windows::Forms::DataGridView^ dataGridView5;

private: System::Windows::Forms::Button^ button4;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ id10;

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

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Псевдокод10;

private: System::Windows::Forms::RichTextBox^ richTextBox4;

private: System::Windows::Forms::RichTextBox^ richTextBox5;

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

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

private: System::Windows::Forms::DataGridViewTextBoxColumn^ id20;

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

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Псевдокод20;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ id30;

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

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Псевдокод30;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ id40;

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

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Псевдокод40;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ id50;

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

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Псевдокод50;

private: System::Windows::Forms::DataGridView^ dataGridView6;

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

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

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

protected:

private:

/// <summary>

/// Обязательная переменная конструктора.

/// </summary>

System::ComponentModel::Container^ components;

#pragma region Windows Form Designer generated code

/// <summary>

/// Требуемый метод для поддержки конструктора — не изменяйте

/// содержимое этого метода с помощью редактора кода.

/// </summary>

void InitializeComponent(void)

{

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

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

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

this->richTextBox1 = (gcnew System::Windows::Forms::RichTextBox());

this->richTextBox2 = (gcnew System::Windows::Forms::RichTextBox());

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

this->button2 = (gcnew System::Windows::Forms::Button());

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

this->richTextBox3 = (gcnew System::Windows::Forms::RichTextBox());

this->dataGridView1 = (gcnew System::Windows::Forms::DataGridView());

this->id10 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

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

this->Псевдокод10 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->button3 = (gcnew System::Windows::Forms::Button());

this->dataGridView2 = (gcnew System::Windows::Forms::DataGridView());

this->id20 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

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

this->Псевдокод20 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->dataGridView3 = (gcnew System::Windows::Forms::DataGridView());

this->id30 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

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

this->Псевдокод30 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->dataGridView4 = (gcnew System::Windows::Forms::DataGridView());

this->id40 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

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

this->Псевдокод40 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->dataGridView5 = (gcnew System::Windows::Forms::DataGridView());

this->id50 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

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

this->Псевдокод50 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->button4 = (gcnew System::Windows::Forms::Button());

this->richTextBox4 = (gcnew System::Windows::Forms::RichTextBox());

this->richTextBox5 = (gcnew System::Windows::Forms::RichTextBox());

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

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

this->dataGridView6 = (gcnew System::Windows::Forms::DataGridView());

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

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

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

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView1))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView2))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView3))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView4))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView5))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView6))->BeginInit();

this->SuspendLayout();

//

// label1

//

this->label1->AutoSize = true;

this->label1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 20));

this->label1->Location = System::Drawing::Point(58, 7);

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

this->label1->Size = System::Drawing::Size(193, 31);

this->label1->TabIndex = 2;

this->label1->Text = L"Исходный код";

//

// label2

//

this->label2->AutoSize = true;

this->label2->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 20));

this->label2->Location = System::Drawing::Point(367, 9);

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

this->label2->Size = System::Drawing::Size(253, 31);

this->label2->TabIndex = 3;

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

//

// button1

//

this->button1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 15));

this->button1->Location = System::Drawing::Point(1171, 133);

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

this->button1->Size = System::Drawing::Size(192, 52);

this->button1->TabIndex = 4;

this->button1->Text = L"Обработать код";

this->button1->UseVisualStyleBackColor = true;

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

//

// richTextBox1

//

this->richTextBox1->Location = System::Drawing::Point(12, 53);

this->richTextBox1->Name = L"richTextBox1";

this->richTextBox1->Size = System::Drawing::Size(303, 421);

this->richTextBox1->TabIndex = 5;

this->richTextBox1->Text = L"";

//

// richTextBox2

//

this->richTextBox2->Location = System::Drawing::Point(338, 53);

this->richTextBox2->Name = L"richTextBox2";

this->richTextBox2->ReadOnly = true;

this->richTextBox2->Size = System::Drawing::Size(303, 421);

this->richTextBox2->TabIndex = 6;

this->richTextBox2->Text = L"";

//

// openFileDialog1

//

this->openFileDialog1->FileName = L"openFileDialog1";

//

// button2

//

this->button2->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 15));

this->button2->Location = System::Drawing::Point(1171, 75);

this->button2->Name = L"button2";

this->button2->Size = System::Drawing::Size(192, 52);

this->button2->TabIndex = 7;

this->button2->Text = L"Выбрать файл";

this->button2->UseVisualStyleBackColor = true;

this->button2->Click += gcnew System::EventHandler(this, &MyForm::button2\_Click);

//

// label3

//

this->label3->AutoSize = true;

this->label3->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 14));

this->label3->Location = System::Drawing::Point(1167, 28);

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

this->label3->Size = System::Drawing::Size(133, 24);

this->label3->TabIndex = 8;

this->label3->Text = L"Строк в коде:";

//

// richTextBox3

//

this->richTextBox3->Location = System::Drawing::Point(1314, 29);

this->richTextBox3->Name = L"richTextBox3";

this->richTextBox3->ReadOnly = true;

this->richTextBox3->Size = System::Drawing::Size(49, 23);

this->richTextBox3->TabIndex = 9;

this->richTextBox3->Text = L"";

//

// dataGridView1

//

this->dataGridView1->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->dataGridView1->Columns->AddRange(gcnew cli::array< System::Windows::Forms::DataGridViewColumn^ >(3) {

this->id10, this->Разделители,

this->Псевдокод10

});

this->dataGridView1->Location = System::Drawing::Point(12, 480);

this->dataGridView1->Name = L"dataGridView1";

this->dataGridView1->Size = System::Drawing::Size(270, 235);

this->dataGridView1->TabIndex = 10;

//

// id10

//

this->id10->HeaderText = L"10";

this->id10->Name = L"id10";

this->id10->Width = 30;

//

// Разделители

//

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

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

this->Разделители->ReadOnly = true;

//

// Псевдокод10

//

this->Псевдокод10->HeaderText = L"Псевдокод";

this->Псевдокод10->Name = L"Псевдокод10";

//

// button3

//

this->button3->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 15));

this->button3->Location = System::Drawing::Point(1171, 191);

this->button3->Name = L"button3";

this->button3->Size = System::Drawing::Size(192, 52);

this->button3->TabIndex = 12;

this->button3->Text = L"Найти лексемы";

this->button3->UseVisualStyleBackColor = true;

this->button3->Click += gcnew System::EventHandler(this, &MyForm::button3\_Click);

//

// dataGridView2

//

this->dataGridView2->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->dataGridView2->Columns->AddRange(gcnew cli::array< System::Windows::Forms::DataGridViewColumn^ >(3) {

this->id20, this->ЗнакиОпераций,

this->Псевдокод20

});

this->dataGridView2->Location = System::Drawing::Point(288, 480);

this->dataGridView2->Name = L"dataGridView2";

this->dataGridView2->Size = System::Drawing::Size(270, 235);

this->dataGridView2->TabIndex = 13;

//

// id20

//

this->id20->HeaderText = L"20";

this->id20->Name = L"id20";

this->id20->Width = 30;

//

// ЗнакиОпераций

//

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

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

this->ЗнакиОпераций->ReadOnly = true;

//

// Псевдокод20

//

this->Псевдокод20->HeaderText = L"Псевдокод";

this->Псевдокод20->Name = L"Псевдокод20";

//

// dataGridView3

//

this->dataGridView3->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->dataGridView3->Columns->AddRange(gcnew cli::array< System::Windows::Forms::DataGridViewColumn^ >(3) {

this->id30, this->Константы,

this->Псевдокод30

});

this->dataGridView3->Location = System::Drawing::Point(564, 480);

this->dataGridView3->Name = L"dataGridView3";

this->dataGridView3->Size = System::Drawing::Size(270, 235);

this->dataGridView3->TabIndex = 14;

//

// id30

//

this->id30->HeaderText = L"30";

this->id30->Name = L"id30";

this->id30->Width = 30;

//

// Константы

//

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

this->Константы->Name = L"Константы";

this->Константы->ReadOnly = true;

//

// Псевдокод30

//

this->Псевдокод30->HeaderText = L"Псевдокод";

this->Псевдокод30->Name = L"Псевдокод30";

//

// dataGridView4

//

this->dataGridView4->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->dataGridView4->Columns->AddRange(gcnew cli::array< System::Windows::Forms::DataGridViewColumn^ >(3) {

this->id40, this->КлючевыеСлова,

this->Псевдокод40

});

this->dataGridView4->Location = System::Drawing::Point(840, 480);

this->dataGridView4->Name = L"dataGridView4";

this->dataGridView4->Size = System::Drawing::Size(270, 235);

this->dataGridView4->TabIndex = 15;

//

// id40

//

this->id40->HeaderText = L"40";

this->id40->Name = L"id40";

this->id40->Width = 30;

//

// КлючевыеСлова

//

this->КлючевыеСлова->HeaderText = L"Ключевые слова";

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

this->КлючевыеСлова->ReadOnly = true;

//

// Псевдокод40

//

this->Псевдокод40->HeaderText = L"Псевдокод";

this->Псевдокод40->Name = L"Псевдокод40";

//

// dataGridView5

//

this->dataGridView5->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->dataGridView5->Columns->AddRange(gcnew cli::array< System::Windows::Forms::DataGridViewColumn^ >(3) {

this->id50, this->Идентификаторы,

this->Псевдокод50

});

this->dataGridView5->Location = System::Drawing::Point(1116, 480);

this->dataGridView5->Name = L"dataGridView5";

this->dataGridView5->Size = System::Drawing::Size(270, 235);

this->dataGridView5->TabIndex = 16;

//

// id50

//

this->id50->HeaderText = L"50";

this->id50->Name = L"id50";

this->id50->Width = 30;

//

// Идентификаторы

//

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

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

this->Идентификаторы->ReadOnly = true;

//

// Псевдокод50

//

this->Псевдокод50->HeaderText = L"Псевдокод";

this->Псевдокод50->Name = L"Псевдокод50";

//

// button4

//

this->button4->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 14));

this->button4->Location = System::Drawing::Point(1171, 422);

this->button4->Name = L"button4";

this->button4->Size = System::Drawing::Size(192, 52);

this->button4->TabIndex = 18;

this->button4->Text = L"Сброс";

this->button4->UseVisualStyleBackColor = true;

this->button4->Click += gcnew System::EventHandler(this, &MyForm::button4\_Click);

//

// richTextBox4

//

this->richTextBox4->Location = System::Drawing::Point(660, 257);

this->richTextBox4->Name = L"richTextBox4";

this->richTextBox4->ReadOnly = true;

this->richTextBox4->Size = System::Drawing::Size(232, 217);

this->richTextBox4->TabIndex = 19;

this->richTextBox4->Text = L"";

//

// richTextBox5

//

this->richTextBox5->Location = System::Drawing::Point(909, 257);

this->richTextBox5->Name = L"richTextBox5";

this->richTextBox5->ReadOnly = true;

this->richTextBox5->Size = System::Drawing::Size(230, 217);

this->richTextBox5->TabIndex = 20;

this->richTextBox5->Text = L"";

//

// label4

//

this->label4->AutoSize = true;

this->label4->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 14));

this->label4->Location = System::Drawing::Point(724, 231);

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

this->label4->Size = System::Drawing::Size(110, 24);

this->label4->TabIndex = 21;

this->label4->Text = L"Псевдокод";

//

// label5

//

this->label5->AutoSize = true;

this->label5->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 14));

this->label5->Location = System::Drawing::Point(929, 231);

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

this->label5->Size = System::Drawing::Size(191, 24);

this->label5->TabIndex = 22;

this->label5->Text = L"Дескрипторный код";

//

// dataGridView6

//

this->dataGridView6->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->dataGridView6->Columns->AddRange(gcnew cli::array< System::Windows::Forms::DataGridViewColumn^ >(3) {

this->НомерСтроки,

this->dataGridViewTextBoxColumn1, this->НевернаяЛексема

});

this->dataGridView6->Location = System::Drawing::Point(660, 12);

this->dataGridView6->Name = L"dataGridView6";

this->dataGridView6->Size = System::Drawing::Size(479, 216);

this->dataGridView6->TabIndex = 23;

//

// НомерСтроки

//

this->НомерСтроки->HeaderText = L"№ строки";

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

this->НомерСтроки->Width = 50;

//

// dataGridViewTextBoxColumn1

//

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

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

this->dataGridViewTextBoxColumn1->ReadOnly = true;

this->dataGridViewTextBoxColumn1->Width = 285;

//

// НевернаяЛексема

//

this->НевернаяЛексема->HeaderText = L"Неверная лексема";

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

//

// MyForm

//

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

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

this->ClientSize = System::Drawing::Size(1396, 724);

this->Controls->Add(this->dataGridView6);

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

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

this->Controls->Add(this->richTextBox5);

this->Controls->Add(this->richTextBox4);

this->Controls->Add(this->button4);

this->Controls->Add(this->dataGridView5);

this->Controls->Add(this->dataGridView4);

this->Controls->Add(this->dataGridView3);

this->Controls->Add(this->dataGridView2);

this->Controls->Add(this->button3);

this->Controls->Add(this->dataGridView1);

this->Controls->Add(this->richTextBox3);

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

this->Controls->Add(this->button2);

this->Controls->Add(this->richTextBox2);

this->Controls->Add(this->richTextBox1);

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

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

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

this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::FixedSingle;

this->MaximizeBox = false;

this->Name = L"MyForm";

this->Text = L"Распознавание строк";

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView1))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView2))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView3))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView4))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView5))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView6))->EndInit();

this->ResumeLayout(false);

this->PerformLayout();

}

bool search(DataGridView^ dataGridView, String^ symbol) {

for (int i = 0; i < dataGridView->RowCount; i++) {

if (dataGridView->Rows[i]->Cells->Count > 0) {

Object^ Cell = dataGridView->Rows[i]->Cells[1]->Value;

if (Cell != nullptr) {

String^ searchSymbol = Cell->ToString();

if (searchSymbol == symbol) {

Object^ num = dataGridView->Rows[i]->Cells[0]->Value;

id = num->ToString();

num = dataGridView->Rows[i]->Cells[2]->Value;

psevdo = num->ToString();

return true;

}

}

}

}

if (symbol == "") return true;

else return false;

}

int Automat1(char c, int s) {

switch (s) {

case 0: { //начало новой строки, увеличивается счетчик

if (c == '\n') return 11;

else if (c == '/') return 4;

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

else return 1;

}

break;

case 1: { //обнаружен другой символ кода

if (c == ' ') return 2;

else if (c == '\n') return 0;

else if (c == '/') return 4;

else return 1;

}

break;

case 2: { //обнаружен 1 пробел

if (c == ' ') return 3;

else if (c == '\n') return 0;

else if (c == '/') return 4;

else return 1;

}

break;

case 3: { //обнаружен 2 пробел подряд, он пропускается

if (c == ' ') return 3;

else if (c == '\n') return 0;

else if (c == '/') return 4;

else return 1;

}

break;

case 4: { //обнаружен 1 слеш

if (c == ' ') return 2;

else if (c == '\n') return 0;

else if (c == '/') return 5;

else if (c == '\*') return 7;

else return 1;

}

break;

case 5: { //обнаружен 2 слеш подряд, начался однострочный комментарий

if (c == '\n') return 0;

else return 6;

}

break;

case 6: { //пропуск символов однострочного комментария

if (c == '\n') return 0;

else return 6;

}

break;

case 7: { //обнаружена звездочка после слеша, начался многострочный комментарий

if (c == '\*') return 9;

else return 8;

}

break;

case 8: { //пропуск символов многострочного комментария

if (c == '\*') return 9;

else return 8;

}

break;

case 9: { //обнаружена звездочка в многострочном комментарии

if (c == '\*') return 9;

else if (c == '/') return 10;

else return 8;

}

break;

case 10: { //обнаружен слеш после звездочки в многострочном комментарии, комментарий закончен

if (c == '\n') return 0;

else if (c == '/') return 4;

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

else return 1;

}

break;

case 11: { //обнаружен 2 перенос строки подряд, он пропускается

if (c == '\n') return 11;

else if (c == '/') return 4;

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

else return 1;

}

}

}

void Automat2(char c, int& s, string& str, int& i, bool& f, bool& err) {

switch (s) {

case 0: {

switch (c) {

//пробел, перенос

case ' ': break;

case '\n': count\_str++;

break;

//запрещенные символы

case '@':

case '`':

case '№':

case '$':

case '^': {

str += c;

s = 999;

err = true;

}

break;

//разделители

case '{':

case '}':

case '(':

case ')':

case '[':

case ']':

case ';':

case ',':

case ':':

case '\'':

case '"': {

str += c;

s = 1;

f = true;

}

break;

//знаки

case '+': {

str += c;

s = 2;

}

break;

case '-': {

str += c;

s = 3;

}

break;

case '<': {

str += c;

s = 4;

}

break;

case '>': {

str += c;

s = 5;

}

break;

case '=':

case '\*':

case '/':

case '%':

case '!': {

str += c;

s = 6;

}

break;

case '&': {

str += c;

s = 8;

}

break;

case '|': {

str += c;

s = 9;

}

break;

//константы

case '0':

case '1':

case '2':

case '3':

case '4':

case '5':

case '6':

case '7':

case '8':

case '9': {

str += c;

s = 10;

}

break;

//ключевые слова

case 'a': {

str += c;

s = 14;

}

break;

case 'b': {

str += c;

s = 18;

}

break;

case 'c': {

str += c;

s = 24;

}

break;

case 'd': {

str += c;

s = 40;

}

break;

case 'e': {

str += c;

s = 53;

}

break;

case 'f': {

str += c;

s = 65;

}

break;

case 'g': {

str += c;

s = 77;

}

break;

case 'i': {

str += c;

s = 80;

}

break;

case 'l': {

str += c;

s = 85;

}

break;

case 'm': {

str += c;

s = 88;

}

break;

case 'n': {

str += c;

s = 94;

}

break;

case 'o': {

str += c;

s = 108;

}

break;

case 'p': {

str += c;

s = 115;

}

break;

case 'r': {

str += c;

s = 131;

}

break;

case 's': {

str += c;

s = 136;

}

break;

case 't': {

str += c;

s = 158;

}

break;

case 'u': {

str += c;

s = 171;

}

break;

case 'v': {

str += c;

s = 183;

}

break;

case 'w': {

str += c;

s = 191;

}

break;

case 'h':

case 'j':

case 'k':

case 'q':

case 'x':

case 'y':

case 'z':

case '\_': {

str += c;

s = 999;

}

}

if (c >= 'A' && c <= 'Z') {

str += c;

s = 999;

}

}

break;

case 1:

break;

case 2: {

switch (c) {

case '+':

case '=': {

str += c;

s = 7;

}

break;

default: {

i--;

f = true;

}

break;

}

}

break;

case 3: {

switch (c) {

case '-':

case '=': {

str += c;

s = 7;

}

break;

default: {

i--;

f = true;

}

break;

}

}

break;

case 4: {

switch (c) {

case '<':

case '=': {

str += c;

s = 7;

}

break;

default: {

i--;

f = true;

}

break;

}

}

break;

case 5: {

switch (c) {

case '>':

case '=': {

str += c;

s = 7;

}

break;

default: {

i--;

f = true;

}

break;

}

}

break;

case 6: {

switch (c) {

case '=': {

str += c;

s = 7;

}

break;

default: {

i--;

f = true;

}

break;

}

}

break;

case 7: {

if (c == str[str.length() - 1]) {

str += c;

err = true;

}

else {

i--;

f = true;

}

}

break;

case 8: {

switch (c) {

case '&': {

str += c;

s = 7;

f = true;

}

break;

default: {

i--;

f = true;

}

break;

}

}

break;

case 9: {

switch (c) {

case '|': {

str += c;

s = 7;

f = true;

}

break;

default: {

i--;

f = true;

}

break;

}

}

break;

case 10: {

switch (c) {

case '0':

case '1':

case '2':

case '3':

case '4':

case '5':

case '6':

case '7':

case '8':

case '9': {

str += c;

}

break;

case '.': {

str += c;

s = 11;

}

break;

case 'e':

case 'E': {

str += c;

s = 12;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || c == '\_') {

str += c;

err = true;

s = 999;

}

else {

i--;

f = true;

}

}

break;

}

}

break;

case 11: {

switch (c) {

case '0':

case '1':

case '2':

case '3':

case '4':

case '5':

case '6':

case '7':

case '8':

case '9': {

str += c;

}

break;

case 'e':

case 'E': {

str += c;

s = 12;

}

break;

default: {

if (c == '.') {

str += c;

err = true;

}

else {

if (str[str.length() - 1] == '.') err = true;

i--;

f = true;

}

}

break;

}

}

break;

case 12: {

switch (c) {

case '+':

case '-': {

str += c;

s = 13;

}

break;

default: {

i--;

f = true;

}

break;

}

}

break;

case 13: {

switch (c) {

case '0':

case '1':

case '2':

case '3':

case '4':

case '5':

case '6':

case '7':

case '8':

case '9': {

str += c;

}

break;

default: {

if (c == 'e' || c == 'E') {

str += c;

err = true;

}

else {

i--;

f = true;

}

}

break;

}

}

break;

case 14: {

switch (c) {

case 'u': {

str += c;

s = 15;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 15: {

switch (c) {

case 't': {

str += c;

s = 16;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 16: {

switch (c) {

case 'o': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 17: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else {

i--;

f = true;

}

}

break;

case 18: {

switch (c) {

case 'o': {

str += c;

s = 19;

}

break;

case 'r': {

str += c;

s = 21;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 19: {

switch (c) {

case 'o': {

str += c;

s = 20;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 20: {

switch (c) {

case 'l': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 21: {

switch (c) {

case 'e': {

str += c;

s = 22;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 22: {

switch (c) {

case 'a': {

str += c;

s = 23;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 23: {

switch (c) {

case 'k': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 24: {

switch (c) {

case 'a': {

str += c;

s = 25;

}

break;

case 'h': {

str += c;

s = 29;

}

break;

case 'l': {

str += c;

s = 31;

}

break;

case 'o': {

str += c;

s = 34;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 25: {

switch (c) {

case 's': {

str += c;

s = 26;

}

break;

case 't': {

str += c;

s = 27;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 26: {

switch (c) {

case 'e': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 27: {

switch (c) {

case 'c': {

str += c;

s = 28;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 28: {

switch (c) {

case 'h': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 29: {

switch (c) {

case 'a': {

str += c;

s = 30;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 30: {

switch (c) {

case 'r': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 31: {

switch (c) {

case 'a': {

str += c;

s = 32;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 32: {

switch (c) {

case 's': {

str += c;

s = 33;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 33: {

switch (c) {

case 's': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 34: {

switch (c) {

case 'n': {

str += c;

s = 35;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 35: {

switch (c) {

case 'c': {

str += c;

s = 36;

}

break;

case 's': {

str += c;

s = 39;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 36: {

switch (c) {

case 'e': {

str += c;

s = 37;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 37: {

switch (c) {

case 'p': {

str += c;

s = 38;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 38: {

switch (c) {

case 't': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 39: {

switch (c) {

case 't': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 40: {

switch (c) {

case 'e': {

str += c;

s = 41;

}

break;

case 'o': {

str += c;

s = 49;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 41: {

switch (c) {

case 'f': {

str += c;

s = 42;

}

break;

case 'l': {

str += c;

s = 46;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 42: {

switch (c) {

case 'a': {

str += c;

s = 43;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 43: {

switch (c) {

case 'u': {

str += c;

s = 44;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 44: {

switch (c) {

case 'l': {

str += c;

s = 45;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 45: {

switch (c) {

case 't': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 46: {

switch (c) {

case 'e': {

str += c;

s = 47;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 47: {

switch (c) {

case 't': {

str += c;

s = 48;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 48: {

switch (c) {

case 'e': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 49: {

switch (c) {

case 'u': {

str += c;

s = 50;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 17;

i--;

f = true;

}

}

break;

}

}

break;

case 50: {

switch (c) {

case 'b': {

str += c;

s = 51;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 51: {

switch (c) {

case 'l': {

str += c;

s = 52;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 52: {

switch (c) {

case 'e': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 53: {

switch (c) {

case 'l': {

str += c;

s = 54;

}

break;

case 'n': {

str += c;

s = 56;

}

break;

case 'x': {

str += c;

s = 58;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 54: {

switch (c) {

case 's': {

str += c;

s = 55;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 55: {

switch (c) {

case 'e': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 56: {

switch (c) {

case 'u': {

str += c;

s = 57;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 57: {

switch (c) {

case 'm': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 58: {

switch (c) {

case 'p': {

str += c;

s = 59;

}

break;

case 't': {

str += c;

s = 62;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 59: {

switch (c) {

case 'o': {

str += c;

s = 60;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 60: {

switch (c) {

case 'r': {

str += c;

s = 61;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 61: {

switch (c) {

case 't': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 62: {

switch (c) {

case 'e': {

str += c;

s = 63;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 63: {

switch (c) {

case 'r': {

str += c;

s = 64;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 64: {

switch (c) {

case 'n': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 65: {

switch (c) {

case 'a': {

str += c;

s = 66;

}

break;

case 'l': {

str += c;

s = 69;

}

break;

case 'o': {

str += c;

s = 72;

}

break;

case 'r': {

str += c;

s = 73;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 66: {

switch (c) {

case 'l': {

str += c;

s = 67;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 67: {

switch (c) {

case 's': {

str += c;

s = 68;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 68: {

switch (c) {

case 'e': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 69: {

switch (c) {

case 'o': {

str += c;

s = 70;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 70: {

switch (c) {

case 'a': {

str += c;

s = 71;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 71: {

switch (c) {

case 't': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 72: {

switch (c) {

case 'r': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 73: {

switch (c) {

case 'i': {

str += c;

s = 74;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 74: {

switch (c) {

case 'e': {

str += c;

s = 75;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 75: {

switch (c) {

case 'n': {

str += c;

s = 76;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 76: {

switch (c) {

case 'd': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 77: {

switch (c) {

case 'o': {

str += c;

s = 78;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 78: {

switch (c) {

case 't': {

str += c;

s = 79;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 79: {

switch (c) {

case 'o': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 80: {

switch (c) {

case 'f': {

str += c;

s = 17;

}

break;

case 'n': {

str += c;

s = 81;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 81: {

switch (c) {

case 'l': {

str += c;

s = 82;

}

break;

case 't': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 82: {

switch (c) {

case 'i': {

str += c;

s = 83;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 83: {

switch (c) {

case 'n': {

str += c;

s = 84;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 84: {

switch (c) {

case 'e': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 85: {

switch (c) {

case 'o': {

str += c;

s = 86;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 86: {

switch (c) {

case 'n': {

str += c;

s = 87;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 87: {

switch (c) {

case 'g': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 88: {

switch (c) {

case 'u': {

str += c;

s = 89;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 89: {

switch (c) {

case 't': {

str += c;

s = 90;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 90: {

switch (c) {

case 'a': {

str += c;

s = 91;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 91: {

switch (c) {

case 'b': {

str += c;

s = 92;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 92: {

switch (c) {

case 'l': {

str += c;

s = 93;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 93: {

switch (c) {

case 'e': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 94: {

switch (c) {

case 'a': {

str += c;

s = 95;

}

break;

case 'e': {

str += c;

s = 102;

}

break;

case 'u': {

str += c;

s = 103;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 95: {

switch (c) {

case 'm': {

str += c;

s = 96;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 96: {

switch (c) {

case 'e': {

str += c;

s = 97;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 97: {

switch (c) {

case 's': {

str += c;

s = 98;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 98: {

switch (c) {

case 'p': {

str += c;

s = 99;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 99: {

switch (c) {

case 'a': {

str += c;

s = 100;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 100: {

switch (c) {

case 'c': {

str += c;

s = 101;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 101: {

switch (c) {

case 'e': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 102: {

switch (c) {

case 'w': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 103: {

switch (c) {

case 'l': {

str += c;

s = 104;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 104: {

switch (c) {

case 'l': {

str += c;

s = 105;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 105: {

switch (c) {

case 'p': {

str += c;

s = 106;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 106: {

switch (c) {

case 't': {

str += c;

s = 107;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 107: {

switch (c) {

case 'r': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 108: {

switch (c) {

case 'p': {

str += c;

s = 109;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 109: {

switch (c) {

case 'e': {

str += c;

s = 110;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 110: {

switch (c) {

case 'r': {

str += c;

s = 111;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 111: {

switch (c) {

case 'a': {

str += c;

s = 112;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 112: {

switch (c) {

case 't': {

str += c;

s = 113;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 113: {

switch (c) {

case 'o': {

str += c;

s = 114;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 114: {

switch (c) {

case 'r': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 115: {

switch (c) {

case 'r': {

str += c;

s = 116;

}

break;

case 'u': {

str += c;

s = 127;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 116: {

switch (c) {

case 'i': {

str += c;

s = 117;

}

break;

case 'o': {

str += c;

s = 121;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 117: {

switch (c) {

case 'v': {

str += c;

s = 118;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 118: {

switch (c) {

case 'a': {

str += c;

s = 119;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 119: {

switch (c) {

case 't': {

str += c;

s = 120;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 120: {

switch (c) {

case 'e': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 121: {

switch (c) {

case 't': {

str += c;

s = 122;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 122: {

switch (c) {

case 'e': {

str += c;

s = 123;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 123: {

switch (c) {

case 'c': {

str += c;

s = 124;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 124: {

switch (c) {

case 't': {

str += c;

s = 125;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 125: {

switch (c) {

case 'e': {

str += c;

s = 126;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 126: {

switch (c) {

case 'd': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 127: {

switch (c) {

case 'b': {

str += c;

s = 128;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 128: {

switch (c) {

case 'l': {

str += c;

s = 129;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 129: {

switch (c) {

case 'i': {

str += c;

s = 130;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 130: {

switch (c) {

case 'c': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 131: {

switch (c) {

case 'e': {

str += c;

s = 132;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 132: {

switch (c) {

case 't': {

str += c;

s = 133;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 133: {

switch (c) {

case 'u': {

str += c;

s = 134;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 134: {

switch (c) {

case 'r': {

str += c;

s = 135;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 135: {

switch (c) {

case 'n': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 136: {

switch (c) {

case 'h': {

str += c;

s = 137;

}

break;

case 'i': {

str += c;

s = 140;

}

break;

case 't': {

str += c;

s = 147;

}

break;

case 'w': {

str += c;

s = 154;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 137: {

switch (c) {

case 'o': {

str += c;

s = 138;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 138: {

switch (c) {

case 'r': {

str += c;

s = 139;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 139: {

switch (c) {

case 't': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 140: {

switch (c) {

case 'g': {

str += c;

s = 141;

}

break;

case 'z': {

str += c;

s = 144;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 141: {

switch (c) {

case 'n': {

str += c;

s = 142;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 142: {

switch (c) {

case 'e': {

str += c;

s = 143;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 143: {

switch (c) {

case 'd': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 144: {

switch (c) {

case 'e': {

str += c;

s = 145;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 145: {

switch (c) {

case 'o': {

str += c;

s = 146;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 146: {

switch (c) {

case 'f': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 147: {

switch (c) {

case 'a': {

str += c;

s = 148;

}

break;

case 'r': {

str += c;

s = 151;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 148: {

switch (c) {

case 't': {

str += c;

s = 149;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 149: {

switch (c) {

case 'i': {

str += c;

s = 150;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 150: {

switch (c) {

case 'c': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 151: {

switch (c) {

case 'u': {

str += c;

s = 152;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 152: {

switch (c) {

case 'c': {

str += c;

s = 153;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 153: {

switch (c) {

case 't': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 154: {

switch (c) {

case 'i': {

str += c;

s = 155;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 155: {

switch (c) {

case 't': {

str += c;

s = 156;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 156: {

switch (c) {

case 'c': {

str += c;

s = 157;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 157: {

switch (c) {

case 'h': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 158: {

switch (c) {

case 'e': {

str += c;

s = 159;

}

break;

case 'h': {

str += c;

s = 165;

}

break;

case 'r': {

str += c;

s = 169;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 159: {

switch (c) {

case 'm': {

str += c;

s = 160;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 160: {

switch (c) {

case 'p': {

str += c;

s = 161;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 161: {

switch (c) {

case 'l': {

str += c;

s = 162;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 162: {

switch (c) {

case 'a': {

str += c;

s = 163;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 163: {

switch (c) {

case 't': {

str += c;

s = 164;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 164: {

switch (c) {

case 'e': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 165: {

switch (c) {

case 'i': {

str += c;

s = 166;

}

break;

case 'r': {

str += c;

s = 167;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 166: {

switch (c) {

case 's': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 167: {

switch (c) {

case 'o': {

str += c;

s = 168;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 168: {

switch (c) {

case 'w': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 169: {

switch (c) {

case 'u': {

str += c;

s = 170;

}

break;

case 'y': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 170: {

switch (c) {

case 'e': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 171: {

switch (c) {

case 'n': {

str += c;

s = 172;

}

break;

case 's': {

str += c;

s = 180;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 172: {

switch (c) {

case 'i': {

str += c;

s = 173;

}

break;

case 's': {

str += c;

s = 175;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 173: {

switch (c) {

case 'o': {

str += c;

s = 174;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 174: {

switch (c) {

case 'n': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 175: {

switch (c) {

case 'i': {

str += c;

s = 176;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 176: {

switch (c) {

case 'g': {

str += c;

s = 177;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 177: {

switch (c) {

case 'n': {

str += c;

s = 178;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 178: {

switch (c) {

case 'e': {

str += c;

s = 179;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 179: {

switch (c) {

case 'd': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 180: {

switch (c) {

case 'i': {

str += c;

s = 181;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 181: {

switch (c) {

case 'n': {

str += c;

s = 182;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 182: {

switch (c) {

case 'g': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 183: {

switch (c) {

case 'i': {

str += c;

s = 184;

}

break;

case 'o': {

str += c;

s = 189;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 184: {

switch (c) {

case 'r': {

str += c;

s = 185;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 185: {

switch (c) {

case 't': {

str += c;

s = 186;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 186: {

switch (c) {

case 'u': {

str += c;

s = 187;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 187: {

switch (c) {

case 'a': {

str += c;

s = 188;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 188: {

switch (c) {

case 'l': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 189: {

switch (c) {

case 'i': {

str += c;

s = 190;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 190: {

switch (c) {

case 'd': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 191: {

switch (c) {

case 'h': {

str += c;

s = 192;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 192: {

switch (c) {

case 'i': {

str += c;

s = 193;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 193: {

switch (c) {

case 'l': {

str += c;

s = 194;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

case 194: {

switch (c) {

case 'e': {

str += c;

s = 17;

}

break;

default: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

s = 999;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

s = 999;

i--;

f = true;

}

}

break;

}

}

break;

//идентификатор

case 999: {

if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '\_') {

str += c;

}

else if (c == '@' || c == '№' || c == '$' || c == '^' || c == '`' || c == '~') {

str += c;

err = true;

}

else {

i--;

f = true;

}

}

}

}

public: String^ prog = "";

public: String^ id = "";

public: String^ psevdo = "";

public: int count\_r = 0, count\_z = 0, count\_const = 0, count\_key = 0, count\_id = 0, count\_str = 1;

#pragma endregion

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) {

setlocale(LC\_ALL, "Russian");

richTextBox2->Text = "";

if (richTextBox1->Text == "") richTextBox2->Text = "Код не введен!";

int count = 0;

char c;

String^ str1 = "";

String^ str00 = richTextBox1->Text;

str00 += "\n";

string str = "";

string str0 = marshal\_as<string>(str00);

int s = 0;

for (int i = 0; i < str0.length(); i++) {

c = str0[i];

s = Automat1(c, s);

if (s == 0 || s == 1 || s == 2 || s == 4) str += c; //запись символа кода в строку

if (s == 5 || s == 7) str.erase(str.length() - 1, 1); //стирание предыдущего символа (при // и /\*)

if (s == 0) {

if (str != "" && str != "\n") {

count++; //увеличение счетчика при переносе строки

str1 = count + "\t" + marshal\_as<String^>(str);

prog += marshal\_as<String^>(str);

richTextBox2->Text += str1;

}

str = "";

}

}

if (s == 8 || s == 9) {

String^ symbol = "Обнаружен незакрытый комментарий";

cli::array<String^>^ row = gcnew cli::array<String^> { "", symbol, "" };

dataGridView6->Rows->Add(row);

}

str1 = count + "";

richTextBox3->Text = str1;

}

private: System::Void button2\_Click(System::Object^ sender, System::EventArgs^ e) {

prog = "";

setlocale(LC\_ALL, "Russian");

OpenFileDialog^ openFile = gcnew OpenFileDialog();

openFile->Filter = "Текстовые файлы (\*.txt)|\*.txt";

String^ fileName;

if (System::Windows::Forms::DialogResult::OK == openFile->ShowDialog()) fileName = openFile->FileName;

try {

StreamReader^ file = gcnew StreamReader(fileName, Encoding::GetEncoding("windows-1251"));

richTextBox1->Text = file->ReadToEnd();

}

catch (Exception^ e)

{

richTextBox2->Text = "Ошибка открытия файла!";

}

}

private: System::Void button3\_Click(System::Object^ sender, System::EventArgs^ e) {

String^ str0 = prog;

String^ symbol;

String^ count;

String^ pk;

string str00 = marshal\_as<string>(str0), str = "";

bool f = false, err = false;

char c;

int s = 0;

for (int i = 0; i < str00.length(); i++) {

c = str00[i];

Automat2(c, s, str, i, f, err);

if (c == ' ' || c == '\n') richTextBox4->Text += Char::ToString(c);

if (s == 1) { //разделители

if (f) {

symbol += marshal\_as<String^>(str);

if (!search(dataGridView1, symbol)) {

count\_r++;

count = Convert::ToString(count\_r);

pk = symbol;

cli::array<String^>^ row = gcnew cli::array<String^> { count, symbol, pk };

dataGridView1->Rows->Add(row);

richTextBox4->Text += pk;

richTextBox5->Text += "(10, " + count + "), ";

}

else {

richTextBox4->Text += psevdo;

richTextBox5->Text += "(10, " + id + "), ";

}

symbol = "";

str = "";

f = false;

s = 0;

count = "";

}

}

if (s >= 2 && s <= 9) { //знаки

if (f && !err) {

for (int i = 0; i < str.length(); i++)

symbol += Char::ToString(str[i]);

if (!search(dataGridView2, symbol)) {

count\_z++;

count = Convert::ToString(count\_z);

pk = symbol;

cli::array<String^>^ row = gcnew cli::array<String^> { count, symbol, pk };

dataGridView2->Rows->Add(row);

richTextBox4->Text += pk;

richTextBox5->Text += "(20, " + count + "), ";

}

else {

richTextBox4->Text += psevdo;

richTextBox5->Text += "(20, " + id + "), ";

}

symbol = "";

str = "";

f = false;

s = 0;

count = "";

}

if (f && err) {

symbol = "Неверный знак операции";

cli::array<String^>^ row = gcnew cli::array<String^> { Convert::ToString(count\_str), symbol, marshal\_as<String^>(str) };

dataGridView6->Rows->Add(row);

symbol = "";

str = "";

f = false;

err = false;

s = 0;

}

}

if (s == 10 || s == 11 || s == 13) { //константы

if (f && !err) {

for (int i = 0; i < str.length(); i++)

symbol += Char::ToString(str[i]);

if (!search(dataGridView3, symbol)) {

count\_const++;

count = Convert::ToString(count\_const);

pk = symbol;

cli::array<String^>^ row = gcnew cli::array<String^> { count, symbol, pk };

dataGridView3->Rows->Add(row);

richTextBox4->Text += pk;

richTextBox5->Text += "(30, " + count + "), ";

}

else {

richTextBox4->Text += psevdo;

richTextBox5->Text += "(30, " + id + "), ";

}

str = "";

f = false;

s = 0;

count = "";

symbol = "";

}

if (f && err) {

symbol = "Неверная константа";

cli::array<String^>^ row = gcnew cli::array<String^> { Convert::ToString(count\_str), symbol, marshal\_as<String^>(str) };

dataGridView6->Rows->Add(row);

symbol = "";

str = "";

f = false;

err = false;

s = 0;

}

}

if (s == 17) { //ключевые слова

if (f && !err) {

for (int i = 0; i < str.length(); i++)

symbol += Char::ToString(str[i]);

if (!search(dataGridView4, symbol)) {

count\_key++;

count = Convert::ToString(count\_key);

pk = symbol;

cli::array<String^>^ row = gcnew cli::array<String^> { count, symbol, pk };

dataGridView4->Rows->Add(row);

richTextBox4->Text += pk;

richTextBox5->Text += "(40, " + count + "), ";

}

else {

richTextBox4->Text += psevdo;

richTextBox5->Text += "(40, " + id + "), ";

}

symbol = "";

str = "";

f = false;

s = 0;

count = "";

pk = "";

}

if (f && err) {

symbol = "Неверное ключевое слово";

cli::array<String^>^ row = gcnew cli::array<String^> { Convert::ToString(count\_str), symbol, marshal\_as<String^>(str) };

dataGridView6->Rows->Add(row);

symbol = "";

str = "";

f = false;

err = false;

s = 0;

}

}

if (s == 999) { //идентификаторы

if (f && !err) {

for (int i = 0; i < str.length(); i++)

symbol += Char::ToString(str[i]);

if (!search(dataGridView5, symbol)) {

count\_id++;

count = Convert::ToString(count\_id);

pk = "id" + count;

cli::array<String^>^ row = gcnew cli::array<String^> { count, symbol, pk };

dataGridView5->Rows->Add(row);

richTextBox4->Text += pk;

richTextBox5->Text += "(50, " + count + "), ";

}

else {

richTextBox4->Text += psevdo;

richTextBox5->Text += "(50, " + id + "), ";

}

symbol = "";

str = "";

f = false;

err = false;

s = 0;

count = "";

}

if (f && err) {

if (str.length() == 1) symbol = "Неверный символ";

else symbol = "Неверный идентификатор";

cli::array<String^>^ row = gcnew cli::array<String^> { Convert::ToString(count\_str), symbol, marshal\_as<String^>(str) };

dataGridView6->Rows->Add(row);

symbol = "";

str = "";

f = false;

err = false;

s = 0;

}

}

}

if (richTextBox5->Text != "") richTextBox5->Text = richTextBox5->Text->Substring(0, richTextBox5->Text->Length - 1);

string pcode = marshal\_as<string>(richTextBox4->Text);

s = 0;

for (int i = 0; i < pcode.length(); i++) {

Automat1(pcode[i], s);

}

richTextBox4->Text = marshal\_as<String^>(pcode);

}

private: System::Void button4\_Click(System::Object^ sender, System::EventArgs^ e) {

richTextBox1->Clear();

richTextBox2->Clear();

richTextBox3->Clear();

richTextBox4->Clear();

richTextBox5->Clear();

dataGridView1->Rows->Clear();

dataGridView2->Rows->Clear();

dataGridView3->Rows->Clear();

dataGridView4->Rows->Clear();

dataGridView5->Rows->Clear();

dataGridView6->Rows->Clear();

prog = "";

count\_r = 0, count\_z = 0, count\_const = 0, count\_key = 0, count\_id = 0, count\_str = 1;

}

};

}

# Приложение 4