ГОСУДАРСТВЕННОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Программное обеспечение информационных технологий»

ТЕОРИЯ ФОРМАЛЬНЫХ ЯЗЫКОВ

Методические рекомендации к лабораторным работам для студентов направления подготовки 09.03.04 «Программная инженерия» дневной формы обучения

Могилев 2018

УДК 004.43
ББК 32.973.26-018.1
Т 33

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Программное обеспечение информационных технологий» «18» мая 2018 г., протокол № 10

Составитель канд. техн. наук, доц. Э. И. Ясюкович

Рецензент канд. техн. наук, доц. В. В. Кутузов

Методические рекомендации содержат основные базовые теоретические сведения, некоторые приемы реализации задач, а также практические задания для выполнения лабораторных работ по всем темам курса «Теория формальных языков».

Учебно-методическое издание

ТЕОРИЯ ФОРМАЛЬНЫХ ЯЗЫКОВ

Ответственный за выпуск	К. В. Овсянников
Технический редактор	А. А. Подошевко
Компьютерная верстка	Н. П. Полевничая

Подписано в печать . Формат 60×84/16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл. печ. л. . Уч.-изд. л. . Тираж 31 экз. Заказ №

Издатель и полиграфическое исполнение:
Государственное учреждение высшего профессионального образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 24.01.2014.
Пр. Мира, 43, 212000, Могилев.

Содержание

Введение

Классическая теория формальных языков, грамматик и автоматов является одной из важнейших составных частей инженерного образования по информационным технологиям. Она дала новый стимул к развитию математической лингвистики и методов искусственного интеллекта, связанных с естественными и искусственными языками, а ее элементы успешно применяются при анализе и выполнении манипуляций с текстами и разработке трансляторов.

Дисциплина «Теория формальных языков» включает такие разделы, как конечные автоматы, нотация Бэкуса-Наура, машины Тьюринга, формальные языки и грамматики, регулярные выражения.

Цель методических рекомендаций заключается в закреплении студентами знаний по дисциплине «Теория формальных языков» и в приобретении практических навыков использования основных ее положений для решения прикладных задач.

Основы синтаксических методов данной теории были заложены Н. Хомским в 40–50-е гг. XX в. в его лингвистических работах, посвященных изучению естественных языков, и нашли широкое практическое применение в области разработки и реализации языков программирования.

В методических рекомендациях каждая лабораторная работа содержит краткие теоретические сведения для самостоятельной подготовки к ее выполнению, порядок выполнения, содержание отчета, варианты заданий и контрольные вопросы. Цель каждой работы соответствует названию, а ее выполнение производится в следующем порядке:

1) ознакомиться с основными теоретическими положениями;
2) выбрать из таблицы «Варианты заданий для выполнения работы» согласно варианту, указанному преподавателем, задание и исходные данные;
3) выполнить задание и оформить отчет.

1 Лабораторная работа № 1. Построение таблиц истинности алгебры высказываний

В математической логике конкретное содержание высказывания не рассматривается, важно только, истинно оно или ложно. Поэтому высказывание можно представить некоторой переменной величиной, значением которой может быть только 1, если оно истинно, или 0, если ложно. Простые высказывания называются логическими переменными и для простоты записи их обозначают латинскими буквами: A, B, C … . Сложные высказывания – это логические функции, которые также могут принимать значения только 0 или 1 и обозначаются строчными буквами латинского алфавита.

В алгебре логики все логические функции путём логических преобразований могут быть сведены к трём базовым логическим операциям: конъюнкция, дизъюнкция и инверсия. Функции двух переменных называют простейшими. В таблице 1.1 приведены названия, обозначения и результаты наиболее часто используемых простейших логических функций.

Таблица 1.1 – Основные логические функции

x_1	x_2	Конъюнкция &; ∧	Дизъюнкция ∨	Инверсия $\bar{x}_1$	Импликация →; ⊃	Эквивалентность ↔; ~	Штрих Шеффера $x_1 \mid x_2$	Сложение по модулю 2, ⊕	Стрелка Пирса ↑ и функция Вебба ↓
0	0	0	0	1	1	1	1	0	1
0	1	0	1	1	1	0	1	1	0
1	0	0	1	0	0	0	1	1	0
1	1	1	1	0	1	1	0	0	0

Функцией алгебры логики от n переменных называют однозначное отображение множества всевозможных наборов значений n булевых переменных в множество {0, 1}. Такую функцию можно представить в виде таблицы из n + 1 столбцов и 2^n строк, называемой таблицей истинности. Наборы значений переменных в таблице истинности располагают в порядке возрастания – лексикографическом порядке. Это первый способ описания логической функции.

Второй способ описания булевой функции состоит в перечислении наборов значений, на которых функция равна 1 – множество T_1 – или 0 – множество T_0.

Третий способ описания логической функции – это представление ее в виде вектора. Так как порядок перечисления наборов входных переменных установлен, то достаточно указать только столбец функции, например в виде 01101001.

Порядок выполнения работы

Выполнить задания, приведенные в таблице 1.2.

Таблица 1.2 – Варианты заданий

Вариант	Задание 1 (установить, равносильны ли два высказывания)	Задание 2 (упростить логические выражения)
1	$A \wedge B$ и $\overline{\overline{A} \vee B}$	$X \wedge Y \wedge Z \vee \overline{X \wedge Y \wedge Z} \vee X \wedge \overline{Y}$
2	$B \wedge A$ и $\overline{B \vee \overline{A}}$	$(A \vee C) \wedge (A \vee B) \wedge (A \vee C)$
3	$A \vee B$ и $A \wedge B$	$\left((X \vee Y) \wedge \overline{X}\right) \vee (\overline{(X \vee Y)} \wedge \overline{X})$
4	$B \vee A$ и $\overline{\overline{B} \wedge A}$	$X \wedge Y \wedge Z \vee X \wedge Y \wedge \overline{Z} \vee \overline{X} \wedge Y \wedge Z$
5	$\overline{A} \wedge B$ и $\overline{A \vee B}$	$(A \vee C) \wedge (\overline{A} \vee B) \wedge (A \vee \overline{C})$
6	$B \wedge A$ и $\overline{B} \vee \overline{A}$	$X \wedge Y \wedge Z \vee X \wedge \overline{Y \wedge Z} \vee X \wedge \overline{Y}$
7	$A \vee \overline{B}$ и $\overline{A} \wedge \overline{B}$	$(A \vee \overline{B}) \wedge (\overline{A} \vee B) \vee \overline{A \wedge B}$
8	$\overline{B \vee A}$ и $\overline{\overline{B} \wedge \overline{A}}$	$\overline{(X \wedge \overline{Y} \vee Z) \wedge Y \vee \overline{Z}}$
9	$\overline{A \wedge B}$ и $A \vee B$	$X \wedge Y \wedge Z \vee X \wedge Y \wedge \overline{Z} \vee \overline{X} \wedge Y \wedge Z$
10	$B \wedge \overline{A}$ и $\overline{B \vee A}$	$(X \vee Y \vee Z) \wedge (X \vee \overline{Y} \vee \overline{Z})$
11	$A \vee B$ и $\overline{A \wedge \overline{B}}$	$(\overline{A} \wedge B) \wedge (B \vee C) \wedge (A \vee (B \wedge C))$
12	$B \wedge A$ и $\overline{B} \vee \overline{A}$	$\left(\overline{\overline{X} \wedge \overline{Y}} \vee \overline{X}\right) \wedge \overline{(X \vee \overline{\overline{X} \wedge Y})}$

Контрольные вопросы

1 Что такое математическая логика?
2 Какие операции используются в алгебре логики?
3 Как называются и обозначаются простые высказывания?
4 Назовите базовые логические операции.
5 Какие логические функции Вы знаете?
6 Что такое таблица истинности?
7 Что такое функция алгебры логики от n переменных?
8 Что такое лексикографический порядок в таблице истинности?
9 Какие способы описания булевых функций Вы знаете?
10 Назовите правила преобразования логических выражений.

2 Лабораторная работа № 2. Изучение операций с множествами

Под множеством понимают совокупность объектов, рассматриваемых как единое целое и обладающих общим характеристическим свойством. Например, множество целых чисел: A = {x | x – целое}. Один из основателей теории множеств Г. Кантор дал такое определение множеству: «Множество есть многое, мыслимое как целое». Множество является конечным, если конечно число его элементов, т. е. если существует натуральное число N, являющееся количеством элементов множества.

Пустое множество – это множество, не содержащее ни одного элемента, обозначается символом ∅ или ε. Множество X является подмножеством множества Y, если любой элемент множества X принадлежит множеству Y.

Множества обозначают прописными буквами латинского алфавита, например, A, B, C, а его элементы – строчными, например, a, b, c.

Множество может состоять из элементов, которые сами являются множествами. Следует различать элемент «a» и множество, состоящее из единственного элемента «a», например, множество A = {1, 2} состоит из двух элементов «1» и «2»; но множество {A} состоит из одного элемента «A».

Если элемент «a» принадлежит множеству A, то это записывается следующим образом: $a \in A$, а если не принадлежит, то $a \notin A$.

Например, если A_1 – множество простых чисел, A_2 – множество целых чисел и задано a = 4, то $a \in A_2$, $a \notin A_1$, т. к. простое число – это натуральное число, которое имеет ровно два различных делителя (только 1 и само себя).

Если все элементы множества A являются элементами множества B и наоборот, т. е. множества A и B совпадают, т. е. A = B.

Если каждый элемент множества A является элементом множества B, то говорят, что множество A является подмножеством множества B, и записывают $A \subseteq B$ или $B \supseteq A$. Отметим, что по определению само множество A является своим подмножеством, т. е. $A \subseteq A$.

Если $A \subseteq B$ и $B \subseteq A$, то по ранее введенному определению A = B. Если же $A \subseteq B$ и $A \neq B$, то A есть собственное подмножество множества B: $A \subset B$, а если A не является собственным подмножеством B, то записывают $A \not\subset B$.

Например, если A – множество четных чисел, B – множество целых чисел, C – множество нечетных чисел, то $A \subset B$, $C \subset B$, $A \not\subset C$, $B \not\subset A$.

Не следует смешивать отношение принадлежности "$\in$" и отношение включения "$\subseteq$". Например, если A = {2} – множество, состоящее из одного элемента, B = {{2}, {4}} – множество, состоящее из двух элементов, каждое из которых является одноэлементным множеством, то имеют место следующие соотношения: $2 \in \{2\}$; $\{2\} \subset \{\{2\}, \{4\}\}$; $2 \notin \{\{2\}, \{4\}\}$.

Принято считать, что пустое множество является подмножеством любого множества: $\varnothing \subseteq A$, где A – любое множество. Таким образом, всякое множество содержит в качестве своих подмножеств пустое множество и само себя.

Операции над множествами. Объединением множеств A и B называется множество, обозначаемое $A \cup B$, состоящее из тех, и только тех элементов, которые принадлежат хотя бы одному из множеств A и B.

Пересечением множеств A и B называется множество, обозначаемое $A \cap B$, состоящее из тех, и только тех элементов, которые принадлежат как множеству A, так и множеству B.

Разностью множеств A и B называется множество, обозначаемое $A \setminus B$, элементами которого являются все те элементы множества A, которые не принадлежат множеству B, и только они.

Симметричной разностью множеств A и B называется множество $A \Delta B$, являющееся объединением разностей множеств $A \setminus B$ и $B \setminus A$,

т. е. $A \Delta B = (A \setminus B) \cup (B \setminus A)$. Например, если A={1, 2, 3, 4}, B={3, 4, 5, 6}, то $A \Delta B = \{1, 2\} \cup \{5, 6\} = \{1, 2, 5, 6\}$.

Дополнение множества A. Если $A \subseteq U$ (где U – некоторое универсальное множество, которое содержит в качестве подмножеств все рассматриваемые выше множества), то дополнением множества A в множестве U называется множество, обозначаемое $\bar{A}$, состоящее из всех тех элементов множества U, которые не принадлежат множеству A.

Декартовым произведением множеств A и B называется множество пар, первая компонента которых принадлежит множеству A, вторая – множеству B. Обозначают $A \times B$. Таким образом, $A \times B = \{(x; y) \mid x \in A, y \in B\}$. Перечислим элементы, принадлежащие множеству $A \times B$, если A = {a, b, c, d}, B = A = {a, b, c, d}. Декартово произведение $A \times B$ = {(a, a), (a, b), (a, c), (a, d); (b, a), (b, b), (b, c), (b, d); (c, a), (c, b), (c, c), (c, d); (d, a), (d, b) ,(d, c), (d, d)}.

Геометрическое моделирование множеств. Диаграммы (круги) Венна. Для наглядного представления множеств и отношений между ними используется диаграммы Венна, называемые также кругами Эйлера или диаграммами Эйлера-Венна.

Универсальное множество изображают в виде прямоугольника, а множества, входящие в универсальное множество, – в виде кругов внутри прямоугольника; элементу множества соответствует точка внутри круга (рисунок 2.1, *а*).

С помощью диаграмм Венна удобно иллюстрировать операции над множествами. На рисунке 2.1, *б* приведена диаграмма Венна операции объединения множеств $A \cup B$, на рисунке 2.1, *в* – пересечения, на рисунке 2.1, *г* – разности, на рисунке 2.1, *д* – декартового произведения, на рисунке 2.1, *е* – дополнения множества A.

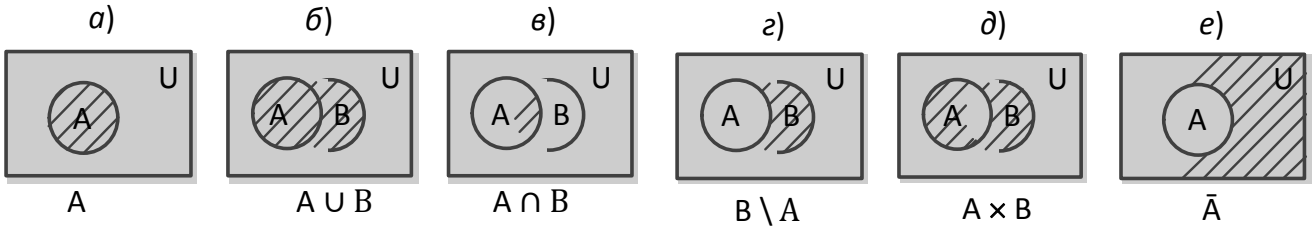


Рисунок 2.1 – Диаграммы Эйлера-Венна

Порядок выполнения работы

Выполнить задания, приведенные в таблице 2.1.

Таблица 2.1 – Варианты заданий

Вариант	Задание 1 (найти результат выполнения операций $\overline{A}$; $B \cup \overline{C}$; $B \setminus A$; $C \cap A$)	Задание 2 (найти результат $B \cup \overline{C}$, $B \setminus A$, $C \cup A$)	Задание 3 (доказать равенство множеств, построить диаграмму Эйлера-Венна)
1	$A = \{x \in R \mid x^2 + 5x - 6 < 0\}$, $B = Z$, $C = \{x \in R \mid \lvert 2x - 1\rvert \geq 0\}$	A: математика B: электрика C: сюръекция	$A \setminus (B \cap C) = (A \setminus B) \cup (A \setminus C)$
2	$A = \{x \in R \mid x^2 + 2x - 1 \geq 0\}$, $B = N$, $C = \{x \in R \mid \lvert 2 + x\rvert < 3\}$	A: логарифм B: физика C: инъекция	$(A \setminus B) \setminus C = A \setminus (B \cup C)$
3	$A = \{x \in R \mid x^2 - x - 12 \leq 0\}$, $B = N$, $C = \{x \in R \mid \lvert 3 + x\rvert > 0\}$	A: трапеция B: энергетика C: компьютер	$(A \cup B) \setminus (A \cap B) = (A \cap \overline{B}) \cup B \cap \overline{A}$
4	$A = \{x \in R \mid x^2 - 2x - 15 > 0\}$, $B = Z$, $C = \{x \in R \mid \lvert 4 - 2x\rvert < 1\}$	A: математика B: транзистор C: множество	$\overline{A \setminus B} = \overline{A} \cup (A \cap B)$
5	$A = \{x \in R \mid x^2 - 4x + 4 > 0\}$, $B = Z$, $C = \{x \in R \mid \lvert x - 1\rvert \leq 3\}$	A: геометрия B: физика C: множество	$(A \setminus B) \setminus C = (A \setminus B) \setminus (B \setminus C)$
6	$A = \{x \in R \mid x^2 + 5x + 6 \leq 0\}$, $B = Z$, $C = \{x \in R \mid \lvert 2x + 3\rvert > 2\}$	A: определение B: транспорт C: отображение	$\overline{A} \cap (B \cup C) = (\overline{A} \cap B) \cup (\overline{A} \cap C)$
7	$A = \{x \in R \mid x^2 - 7x + 6 > 0\}$, $B = N$, $C = \{x \in R \mid \lvert 1 - x\rvert < 2\}$	A :компьютер B: физика C: многочлен	$A \setminus (B \cup C) = (A \setminus B) \cap (A \setminus C)$
8	$A = \{x \in R \mid x^2 - 5x + 6 \leq 0\}$, $B = Z$, $C = \{x \in R \mid \lvert x - 3\rvert \leq 1\}$	A: математика B: движение C: произведение	$A \setminus B = A \cap (A \Delta B)$
9	$A = \{x \in R \mid x^2 + 7x - 6 \geq 0\}$, $B = Z$, $C = \{x \in R \mid \lvert 1 - x\rvert > 2\}$	A: дизъюнкция B: физика C: равенство	$A \cup B = (A \cap B) \Delta (A \Delta B)$
10	$A = \{x \in R \mid x^2 + 3x - 6 < 1\}$, $B = Z$, $C = \{x \in R \mid \lvert 2x + 1\rvert \geq 0\}$	A: конструктор B: физика C: множество	$(A \cup B) \cap (A \cup \overline{B}) = A$
11	$A = \{x \in R \mid x^2 + 2x + 1 > 0\}$, $B = N$, $C = \{x \in R \mid \lvert 3 - x\rvert < 2\}$	A: математика B: механизм C: сюръекция	$\overline{A \setminus B} = \overline{A} \cup (A \cap B)$
12	$A = \{x \in R \mid x^2 + 2x - 1 > 0\}$, $B = Z$, $C = \{x \in R \mid \lvert 2 + x\rvert < 3\}$	A :производная B: физика C: инъекция	$(A \setminus B) \setminus C = (A \setminus C) \setminus (B \setminus C)$

Контрольные вопросы

1 Что понимается под множеством?
2 Кто является основателем теории множеств?
3 Что такое подмножество и пустое множество?
4 Как обозначаются множества и их элементы?
5 Какие операции над множеством Вы знаете?
6 Что понимается под объединением, пересечением множеств?
7 Дайте определения разности и дополнению множеств.
8 Что понимается под декартовым произведением множеств?
9 Как выполняется геометрическое моделирование множеств?
10 Как изображается универсальное множество на диаграмме Венна?

3 Лабораторная работа № 3. Изучение операций алгебры логики

Логика – наука о законах и правилах мышления. Формальная логика – наука о законах и формах мышления. Математическая логика – область знания, в которой формальная логика изучается математическими методами.

Математическая логика изучает вопросы применения математических методов в задачах построения логических схем, а также при разработке алгоритмов и программ.

В логике известны следующие основные формы мышления: понятие, суждение, умозаключение.

Понятие – это форма мышления, которая выделяет существенные признаки предмета или класса предметов, отличающие его от других.

Суждение – это форма мышления, в которой об объектах, их свойствах и отношениях между объектами что-либо утверждается или отрицается.

Умозаключение – это прием мышления, позволяющий на основе нескольких суждений-посылок получить новое суждение – знание или вывод.

Алгебра логики – это математический аппарат, с помощью которого записывается, вычисляется, упрощается и преобразуется логическое высказывание. Поэтому алгебру логики называют также алгеброй высказываний. В математической логике суждения, называемые высказываниями, являются основным понятием.

Порядок выполнения работы

Выполнить задания, приведенные в таблицах 3.1 и 3.2.

Таблица 3.1 – Варианты заданий 1

Вариант	Построить таблицу истинности	Вариант	Построить таблицу истинности
1	$\bar{x}_3 \wedge (\overline{x_1 \vee \bar{x}_2}) \to (\bar{x}_1 \vee x_2) \leftrightarrow \bar{x}_3$	7	$\bar{x}_1 \uparrow x_2 \to (\bar{x}_1 \vee \bar{x}_2) \oplus (\bar{x}_3 \mid x_2)$
2	$\bar{x}_1 \to (\overline{x_1 \vee x_2}) \vee (\bar{x}_3 \oplus x_2)$	8	$\bar{x}_3 \wedge (x_1 \mid \bar{x}_2) \wedge (\overline{x_2 \vee \bar{x}_1})$
3	$\bar{x}_3 \leftrightarrow (\bar{x}_2 \uparrow x_3) \wedge \bar{x}_1 \to x_2$	9	$(\bar{x}_1 \vee x_2) \oplus (x_1 \uparrow \bar{x}_2) \to \bar{x}_3$
4	$x_1 \vee (\overline{x_1 \vee x_2}) \to (x_3 \mid \bar{x}_2)$	10	$(\bar{x}_3 \mid x_2) \wedge (\bar{x}_1 \oplus x_3) \uparrow \bar{x}_2$
5	$\bar{x}_1 \leftrightarrow (\bar{x}_1 \oplus x_2) \wedge (\overline{x_1 \vee \bar{x}_3})$	11	$(x_1 \wedge \bar{x}_2) \mid (\bar{x}_3 \uparrow x_2) \to \bar{x}_1$
6	$(\overline{x_3 \vee \bar{x}_1}) \leftrightarrow \bar{x}_2 \wedge (\overline{x_1 \uparrow \bar{x}_3})$	12	$\bar{x}_2 \to (\bar{x}_1 \vee \bar{x}_2) \uparrow (\overline{\bar{x}_1 \oplus x_2})$

Таблица 3.2 – Варианты заданий 2

Вариант	Установить, равносильны ли два высказывания	Упростить логические выражения
1	2	3
1	A & B и $\overline{\bar{A} \vee B}$	$A \,\&\, \big((\bar{B} \vee \bar{C}) \vee \bar{B} \,\&\, C\big) \vee \bar{A}$ $X \,\&\, Y \,\&\, Z \vee \overline{\bar{X} \,\&\, Y \,\&\, \bar{Z}} \vee X \,\&\, \bar{Y}$
2	B & A и $\overline{B \vee \bar{A}}$	$(A \vee C) \,\&\, (\bar{A} \vee B) \,\&\, (A \vee \bar{C})$ $\overline{(X \,\&\, \bar{Y} \vee Z) \,\&\, Y \vee \bar{Z}}$

Окончание таблицы 3.2

1	2	3
3	$A \vee B$ и $\overline{A \,\&\, \overline{B}}$	$((X \vee Y) \,\&\, \overline{X}) \vee ((\overline{X \vee Y}) \,\&\, \overline{X})$ $(A \vee \overline{B}) \,\&\, (\overline{A} \vee B) \vee \overline{A \,\&\, B}$
4	$B \vee A$ и $\overline{\overline{B} \,\&\, C}$	$(\overline{A} \,\&\, B) \,\&\, (B \vee C) \,\&\, (A \vee (B \,\&\, C))$ $X \,\&\, Y \,\&\, Z \vee X \,\&\, Y \,\&\, \overline{Z} \vee \overline{X} \,\&\, Y \,\&\, Z$
5	$\overline{A} \,\&\, B$ и $\overline{A \vee B}$	$\left(\overline{\overline{X} \,\&\, \overline{Y}} \vee \overline{X}\right) \,\&\, \left(\overline{X \vee \overline{\overline{X} \,\&\, Y}}\right)$ $(A \vee C) \,\&\, (\overline{A} \vee B) \,\&\, (A \vee \overline{C})$
6	$B \,\&\, A$ и $\overline{B} \vee \overline{A}$	$X \,\&\, Y \,\&\, Z \vee X \,\&\, \overline{Y \,\&\, Z} \vee X \,\&\, \overline{Y}$ $A \,\&\, \left((\overline{B} \vee \overline{C}) \vee \overline{B} \,\&\, C\right) \vee \overline{A}$
7	$A \vee \overline{B}$ и $\overline{A} \,\&\, \overline{B}$	$(A \vee \overline{B}) \,\&\, (\overline{A} \vee B) \vee \overline{A \,\&\, B}$ $(X \vee Y \vee Z) \,\&\, (X \vee \overline{Y} \vee \overline{Z})$
8	$\overline{B \vee A}$ и $\overline{\overline{B} \,\&\, \overline{A}}$	$\overline{(X \,\&\, \overline{Y} \vee Z) \,\&\, Y \vee \overline{Z}}$ $(\overline{A} \,\&\, B) \,\&\, (B \vee C) \,\&\, (A \vee (B \,\&\, C))$
9	$\overline{B \vee A}$ и $\overline{\overline{B} \,\&\, \overline{A}}$	$\overline{(X \,\&\, \overline{Y} \vee Z) \,\&\, Y \vee \overline{Z}}$ $(\overline{A} \,\&\, B) \,\&\, (B \vee C) \,\&\, (A \vee (B \,\&\, C)$
10	$\overline{A \,\&\, B}$ и $A \vee B$	$X \,\&\, Y \,\&\, Z \vee X \,\&\, Y \,\&\, \overline{Z} \vee \overline{X} \,\&\, Y \,\&\, Z$ $(\overline{A} \,\&\, B) \,\&\, (B \vee C) \,\&\, (A \vee (B \,\&\, C))$
11	$B \,\&\, \overline{A}$ и $\overline{B \vee A}$	$(X \vee Y \vee Z) \,\&\, (X \vee \overline{Y} \vee \overline{Z})$ $\left(\overline{\overline{X} \,\&\, \overline{Y}} \vee \overline{X}\right) \,\&\, \left(\overline{X \vee \overline{X} \,\&\, Y}\right)$
12	$A \vee \overline{B}$ и $\overline{A} \,\&\, \overline{B}$	$((X \vee Y) \,\&\, \overline{X}) \vee ((\overline{X \vee Y}) \,\&\, \overline{X})$ $(A \vee \overline{B}) \,\&\, (\overline{A} \vee B) \vee \overline{A \,\&\, B}$

Контрольные вопросы

1 Для чего используется таблица истинности в алгебре логики?
2 Какие вопросы изучает математическая логика?
3 Как построить логическую функцию по заданной таблице истинности?
4 Что понимается под термином «понятие» в алгебре логики?
5 Где используется алгебра логики?
6 Как построить таблицу истинности для набора переменных?
7 Какие операции алгебры логики Вы знаете?
8 Что означает понятие «суждение» в алгебре логики?
9 Что такое двойное отрицание?
10 Что такое умозаключение в алгебре логики?

4 Лабораторная работа № 4. Грамматики и конечные автоматы

В теории формальных языков при изучении регулярных выражений, ориентированных на обработку текстов, используются некоторые положения теории конечных автоматов. В этом случае движок регулярных выражений, получая регулярное выражение, создаёт в оперативной памяти таблицу переходов конечного автомата, который под действием очередного входного символа строки текста переходит из одного состояния в другое, пока не дойдет до конечного состояния.

Широкую популярность в теории автоматов получил цифровой автомат, который содержит комбинационную логическую схему с памятью, позволяющую помнить предысторию его работы. Цифровой автомат может принимать входные сигналы и под их воздействием переходить из одного состояния в другое, сохранять это состояние до прихода следующего входного сигнала и выдавать выходные сигналы. Автомат называется конечным, если конечны множества входных сигналов, состояний и выходных сигналов.

Способы задания автоматов. Существуют три основных способа задания автоматов: табличный, графический и матричный.

Табличный способ. Для автомата Мили табличный способ представляется двумя таблицами: таблицей переходов и таблицей выходов.

Графический способ. Автомат представляется ориентированным связным графом (орграфом), вершины которого соответствуют состояниям автомата, а дуги – переходам из состояния в состояние.

Матричный способ. При решении задач часто более удобным способом задания автомата является матричный. При этом используется матрица соединений автомата, т. е. квадратная матрица $C = |c_{ij}|$. Элемент $c_{ij} = x_k/y_s$ этой матрицы, стоящий на пересечении i-й строки и j-го столбца, в случае автомата Мили соответствует входному сигналу x_k, вызывающему переход из состояния a_i в состояние a_j, и выходному сигналу y_s, выдаваемому при этом переходе.

Порядок выполнения работы

1 Построить таблицы переходов и выходов, а также ориентированный граф переходов и матрицу соединений заданного автомата (таблица 4.1).

2 Написать и отладить программный модуль формирования матрицы выходов заданного автомата.

Таблица 4.1 – Варианты заданий

Вариант	Тип автомата	Количество входных сигналов	Количество состояний	Количество выходов
1	Мили	3	3	2
2	Мура	3	4	2
3	Мили	3	3	5
4	Мура	3	5	3
5	Мили	3	2	5
6	Мура	2	4	3
7	Мили	2	5	3
8	Мура	2	4	2
9	Мили	2	3	3
10	Мура	2	2	5
11	Мили	5	4	2
12	Мура	5	3	3

Контрольные вопросы

1 Что понимается под автоматом?
2 Какова структура цифрового автомата?
3 Приведите математическое описание абстрактного автомата.
4 Как классифицируются абстрактные конечные автоматы?
5 Приведите классификацию синхронных автоматов.
6 Приведите законы функционирования автоматов 1-го и 2-го рода.
7 Приведите закон функционирования автомата Мура.
8 Что такое С-автомат?
9 Какие способы задания автоматов Вы знаете?
10 Прокомментируйте матричный способ задания автомата.

5 Лабораторная работа № 5. Построение конечного автомата по регулярной грамматике

Конечный автомат (КА) в теории формальных грамматик – это пятерка (Q, T, F, H, Z), где Q – конечное множество состояний автомата; T – конечное множество допустимых входных символов; F – функция переходов, отображающая множество Q × T во множество Q; H – конечное множество начальных состояний автомата; Z – множество заключительных состояний автомата, $Z \subseteq Q$ [5].

Известны три способа представления функции переходов конечного автомата: командный, табличный и графический.

Командный способ. Каждую команду КА записывают в форме F(q, t) = p, где $q, p \in Q$, $t \in T$.

Табличный способ. Строки таблицы переходов соответствуют входным символам автомата $t \in T$, а столбцы – состояниям Q. Ячейки таблицы заполняются новыми состояниями, соответствующими значению функции F(q, t). Неопределенным значениям функции переходов соответствуют пустые ячейки таблицы.

Графический способ. Строится диаграмма состояний автомата – неупорядоченный ориентированный помеченный граф. Вершины графа помечены именами состояний автомата. Дуга ведет из состояния q в состояние p и помечается списком всех символов $t \in T$, для которых F(q, t) = p. Вершина, соответствующая входному состоянию автомата, снабжается стрелкой. Заключительное состояние на графе обозначается двумя концентрическими окружностями.

При построении конечного автомата по регулярной грамматике входом является регулярная грамматика $G = (V_T, V_N, P, S)$, а выходом – конечный автомат M = (Q, T, F, H, Z).

Алгоритм построения конечного автомата по регулярной грамматике может быть следующим.

1 Пополнить грамматику правилом $A \rightarrow aN$, где $A \in V_N$, $a \in V_T$; N – новый нетерминал, который вводится для каждого правила вида $A \rightarrow a$, если в грам-

матике нет соответствующего ему правила A→ aB, где B ∈ V_N.

2 Начальный символ грамматики S принять за начальное состояние конечного автомата. Из нетерминалов образовать множество состояний автомата Q = V_N, а из терминалов – множество символов входного алфавита T = V_T.

3 Каждое правило A → aB преобразовать в функцию переходов F(A, a) = B, где A, B∈V_N, a∈V_T.

4 Во множество заключительных состояний включить все вершины, помеченные символами B ∈ V_N из правил вида A → aB, для которых имеются соответствующие правила A → a, где A, B ∈ V_N, a ∈ V_T.

5 Во множество заключительных состояний включить все вершины, помеченные символами B ∈ V_N из правил вида A → aB, для которых имеются соответствующие правила A → a, где A, B ∈ V_N, a ∈ V_T.

6 Если в грамматике имеется правило S → ε , где S – начальный символ грамматики, то поместить S во множество заключительных состояний.

Порядок выполнения работы

Разработать программное средство, реализующее следующие функции:

– ввод произвольной формальной грамматики и проверка ее на принадлежность к классу регулярных грамматик;

– построение по заданной регулярной грамматике конечного автомата;

– вывод графа результирующего конечного автомата на экран.

Варианты индивидуальных заданий приведены в таблице 5.1.

Таблица 5.1 – Варианты индивидуальных заданий

Вариант	Регулярная грамматика
1	G = ({S, C, D}, {0, 1}, P, S), где P: 1) S → 1C | 0D; 2) C → 0D | 0S | 1; 3) D → 1C | 1S | 0
2	G = ({S, A, B, C}, {a, b, c}, P, S), где P: 1) S → aA | bB | aC; 2) A → bA | bB | c; 3) B → aA | cC | b; 4) C → bB | bC |a
3	G = ({K, L, M, N}, {a, b, +, -, ⊥}, P, K), где P: 1) K → aL | bM; 2) L → –N | –M; 3) M → +N; 4) N → aL | bM | ⊥
4	G = ({X, Y, Z, W, V}, {0, 1, ~, #, &}, P, X), где P: 1) X → 0Y | 1Z | ε; 3) Z → 1Y| 1W| 0V; 5) V→&Z 2) Y → 0Z| ~W| #; 4) W → 0W| 1W| #;
5	G = ({K, L, M, N, Q, P, R, S}, {0, 1, *, $, /}, V, K), где V: 1) K → 1L| 0N; 3) N → 1R| 1M| *S; 5) P → *L | $; 7) S → 0R; 2) L → 0M| 0P| /Q; 4) Q → 1P; 6) M → $; 8) R → /N | $
6	G = ({E, A, B, C, D}, {0, 1, a, b, c}, P, E), где P: 1) E → 0A | ε; 2) A → aB | aD; 3) B → bB | 1C | c; 4) D → aD | 0C | c
7	G = ({X, Y, Z, V, W}, {0, 1, x, y, z}, P, X), гдеP: 1) X → yY | zZ; 2) Y → 1V; 3) Z → 0W | 0Y 4) V → xZ | xW | 1; 5) W → 1Y | 0
8	G = ({S, A, B, C, D}, {a, b, c, d, ⊥}, P, S), где P: 1) S → aA | bB; 2) A → cC | ⊥; 3) C → cC | cA; 4) B → dD | ⊥; 5) D → dD|dB

Окончание таблицы 5.1

Вариант	Регулярная грамматика
9	G = ({K, L, M, N, P}, {0, 1, &, %, a, b}, C, K), где C: 1) K → 1M | ε; 3) L → 1L | 0L | %P; 5) P → 1P | aP | 0 2) M → 0L | &N | &P; 4) N → aN | bN | %P;
10	G = ({I, J, K, M, N}, {0, 1, ~, !}, P, I), где P: 1) I → 0J| 1K| 0M; 3) K → ~M| 0J| 0N; 5) N → 0I| 1I| 2) J → ~K| 0M; 4) M → 1K| !;
11	G = ({S, A, B, C, D, E}, {a, b, c, d, e, $, ⊥}, P, S), гдеP: 1) S → aA | bB | cC; 3) B → #D | $E; 5) C → cE; 2) A → dD; 4) D → dD | dB | ⊥; 6) E → eE | eB | ⊥
12	G = ({X, Y, Z, V}, {(,), y, z, v}, P, X), где P: 1) X → (Y| ε; 2) Y → yY| zY| zZ; 3) Z → zZ| vZ| vV; 4) V → vV|)

Контрольные вопросы

1 Что такое регулярная грамматика?
2 Дайте математическое определение детерминированного КА.
3 Дайте математическое определение недетерминированного КА.
4 Что понимается под функцией перехода детерминированного КА?
5 Что понимается под функцией перехода недетерминированного КА?
6 Назовите способы представления функции переходов КА.
7 Прокомментируйте табличный способ представления КА.
8 Прокомментируйте графический способ представления КА.
9 Прокомментируйте алгоритм построения КА по регулярной грамматике.
10 Прокомментируйте алгоритм преобразования недетерминированного КА в детерминированный.

6 Лабораторная работа № 6. Минимизация конечного автомата

Задача минимизации конечного автомата, распознающего заданный язык, разрешима конструктивным методом.

Алгоритм минимизации конечного автомата можно определить в следующем порядке:

1) поиск и удаление всех недостижимых состояний;

2) поиск такого разбиения множества состояний автомата, при котором каждое подмножество содержит неразличимые состояния, т. е. если s и t принадлежат некоторому подмножеству, то все a из S d(s, a) и d(t, a) также принадлежат этому подмножеству. Для этого мы множество состояний разбивают на два подмножества: F и S – F;

3) попытка разбиения каждого из подмножеств с соблюдением указанного выше условия. Если возникает ситуация, при которой не удается разбить никакое множество состояний, то процесс разбиения заканчивается;

4) в результате должен получиться некоторый набор множеств состояний $S_1 \ldots S_k$, каждое из которых содержит только неразличимые состояния;

5) внесение в множество состояний минимизированного автомата по одному представителю каждого из множеств S_i.

На этом процесс минимизации завершается.

Алгоритм минимизации целесообразно использовать и при распознавании эквивалентности двух заданных конечных автоматов.

Если необходимо выяснить, эквивалентны ли автоматы M_1 и M2, то достаточно минимизировать каждый из них. Если минимальные автоматы M'_1 и M'_2 имеют множества состояний с разным числом вершин, то исходные автоматы заведомо не эквивалентны.

Конечный автомат M = (Q, T, F, H, Z) может содержать лишние состояния двух типов: недостижимые и эквивалентные состояния.

Конечный автомат, не содержащий недостижимых и эквивалентных состояний, называется приведенным или минимальным конечным автоматом.

Одним из способов минимизации конечного автомата является устранение его недостижимых состояний, которое может быть выполнено с использованием одного из двух алгоритмов: устранения недостижимых состояний или объединения эквивалентных состояний.

Устранение недостижимых состояний автомата выполняется в следующем порядке.

1) поместить начальное состояние автомата в список достижимых состояний $Q_д$, т. е. $Q_д^0 = H$;

2) пополнить список группы достижимых состояний группой состояний-приемников, отсутствующих в этом списке;

3) повторить п. 2, пока список достижимых состояний не перестанет меняться;

4) исключить из множества состояний конечного автомата все состояния, отсутствующие в списке достижимых состояний.

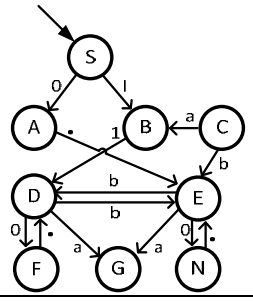
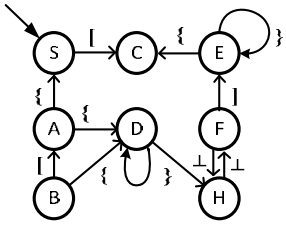
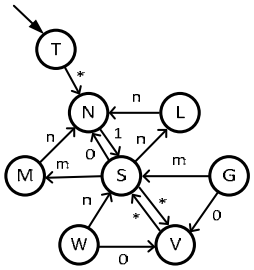
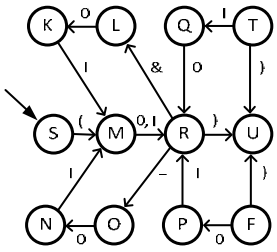
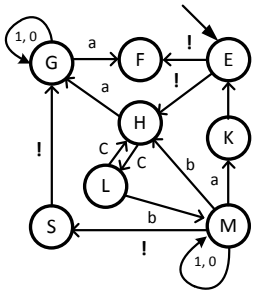
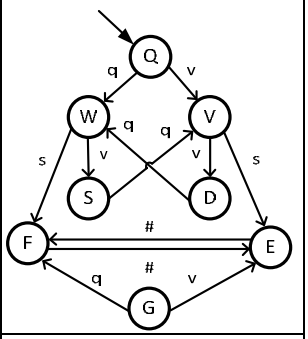
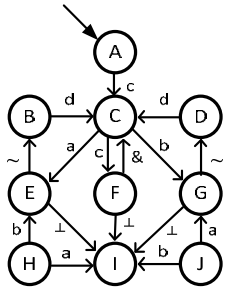
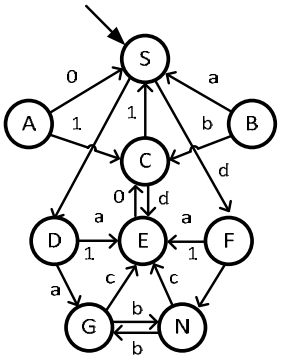
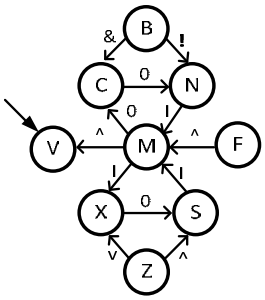
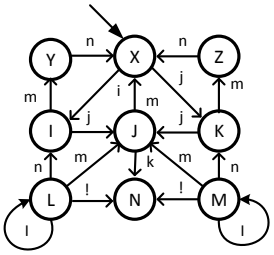
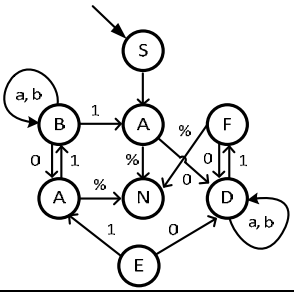
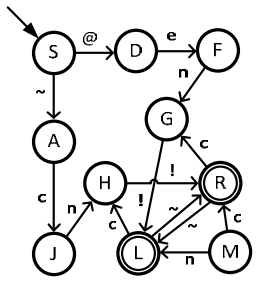
Порядок выполнения работы

Разработать программное средство, реализующее следующие функции:

– ввод исходного конечного автомата и вывод на экран его графа;
– устранение недостижимых состояний конечного автомата;
– исключение эквивалентных состояний конечного автомата;
– вывод на экран графа минимального конечного автомата.

Варианты индивидуальных заданий к лабораторной работе представлены в таблице 6.1.

Таблица 6.1 – Варианты заданий

Вариант	Схема автомата	Вариант	Схема автомата	Вариант	Схема автомата
1		5		9	
2		6		10	
3		7		11	
4		8		12	

Контрольные вопросы

1 Какие состояния конечного автомата называются n-эквивалентными?
2 Прокомментируйте граф переходов конечного автомата.
3 Какие состояния конечного автомата называются недостижимыми?
4 Какой конечный автомат называется приведенным или минимальным?
5 Прокомментируйте алгоритм объединения эквивалентных состояний КА.
6 Сформулируйте алгоритм устранения недостижимых состояний КА.

7 Сформулируйте последовательность устранения недостижимых состояний КА.

8 Какой автомат называют минимальным конечным автоматом?

9 Прокомментируйте процесс минимизации конечного автомата.

10 Какое состояние конечного автомата называется недостижимым?

7 Лабораторная работа № 7. Изучение синтаксических правил БНФ

Для строгого и точного описания синтаксиса языка удобно использовать какой-либо метаязык, наиболее распространенным среди которых является язык формул Бэкуса-Наура, представляющий собой естественный способ описания синтаксиса с помощью металингвистических форм – Бэкуса нормальных форм (БНФ).

Металингвистическая форма определяет одну металингвистическую переменную и состоит из двух частей – левой и правой. В левой части этой формы записывается определяемая металингвистическая переменная, которая заключается в угловые скобки « < » и « > », например, <двоичное число>, <метка>, <арифметическое выражение>. Правая же ее часть содержит все варианты определения конструкции, задаваемой этой металингвистической формой, состоящие из металингвистических переменных и терминальных символов.

Каждый вариант представляет собой цепочку основных символов определяемого языка и металингвистических переменных. Варианты разделяются металингвистической связкой "|", имеющей смысл «или». Левая и правая части формы разделяются метасимволом "::=", проставляемым вместо символа ″→″, означающим «по определению есть». Например, следующие металингвистические формы определяют множество целых чисел:

<целое число> ::= <целое без знака> |+<целое без знака> | –<целое без знака>
<целое без знака> ::= <цифра> | <целое без знака ><цифра>
<цифра> ::=0|1|2|3|4|5|6|7|8|9

БНФ являются широко используемой формой записи правил грамматик, а основным классом их объектов являются имена конструкций описываемого языка или так называемые металингвистические переменные – цепочки основных символов языка.

На практике для описания синтаксиса языков программирования часто используют расширения БНФ, позволяющие более естественно представлять альтернативные, необязательные и повторяющиеся части металингвистических формул. В расширенной БРФ (РБНФ) определение множества целых чисел можно записать в виде

<целое число> ::= [+ | –] <целое без знака>
<целое без знака> ::<цифра> { <цифра> }
<цифра> ::=0|1|2|3|4|5|6|7|8|9

Главное преимущество РБНФ перед БНФ – это возможность описывать простые повторяющиеся конструкции неопределённой длины (списки, строки, последовательности и так далее) без рекурсивных правил, что делает запись в РБНФ одновременно и короче, и удобнее для восприятия.

Однако преимущества РБНФ перед БНФ привели к увеличению сложности автоматической интерпретации его описаний. Например, для описания целого числа можно использовать следующие правила:

```
цифра не ноль   = "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9";
цифра           = "0" | цифра не ноль;
натуральное     = цифра не ноль, {цифра};
целое           = "0" | ["-"], натуральное.
```

Порядок выполнения работы

Разработать процедуру для проверки, какому типу операторов принадлежит заданный элемент синтаксиса (таблица 7.1).

Таблица 7.1 – Варианты заданий

Вариант	Задание 1 (построить синтаксическую диаграмму для элементов синтаксиса)	Вариант	Задание 1 (построить синтаксическую диаграмму для элементов синтаксиса)
1	Идентификатор; оператор if	7	Арифметическое выражение; оператор if
2	Идентификатор; оператор цикла do	8	Арифметическое выражение; оператор цикла for
3	Идентификатор; оператор выбора	9	Арифметическое выражение; оператор выбора
4	Константа со знаком; оператор if	10	Логическое выражение; оператор if
5	Константа со знаком; оператор цикла for	11	Логическое выражение; оператор while
6	Константа со знаком; оператор выбора	12	Логическое выражение; оператор выбора

Контрольные вопросы

1 Что такое формальный язык? Каков смысл его создания?

2 Поясните термин «формальная грамматика». Раскройте каждый компонент математического определения формальной грамматики.

3 Какие типы формальных грамматик Вы знаете?

4 Что такое цепочка вывода?

5 Прокомментируйте металингвистическую форму определения множества целых чисел.

6 Что такое синтаксис и семантика языка?

7 Чем отличается РБНФ от БНФ?

8 Что такое металингвистическая форма?

9 Для чего используются синтаксические диаграммы Вирта?

10 Какие элементы синтаксической диаграммы Вы знаете?

8 Лабораторная работа № 8. Выполнение арифметических операций на машине Тьюринга

Машина Тьюринга (МТ) – это универсальная учебная машина, созданная для уточнения понятия «алгоритм».

Первым идею универсальной машины предложил Алан Тьюринг в 1936 г. Для уточнения понятия алгоритма им был разработан абстрактный универсальный исполнитель, представляющий собой логическую вычислительную конструкцию, но не реальную вычислительную машину. Впоследствии придуманная Тьюрингом вычислительная конструкция была названа машиной Тьюринга.

Машина Тьюринга (рисунок 8.1) состоит из следующих элементов:

– бесконечная в обе стороны лента, разделенная на ячейки;

– каретка, содержащая читающую и записывающую головки (Г);

– программируемый автомат – управляющее устройство (УУ).

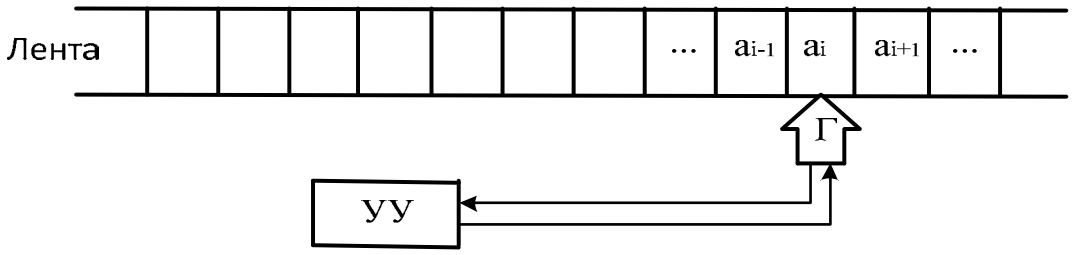


Рисунок 8.1 – Структура машины Тьюринга

Управляющее устройство МТ может перемещаться влево и вправо по ленте, читать и записывать в ячейки ленты символы некоторого конечного алфавита. В процессе своей работы УУ управляется программой, во время каждого шага которой выполняются последовательно следующие действия:

– записывать символ внешнего алфавита в ячейку (в том числе и пустой), заменяя находившийся в ней символ (в том числе и пустой);

– передвигаться на одну ячейку влево или вправо;

– менять свое внутреннее состояние.

Поэтому при составлении программы для каждой пары «символ, состояние» нужно определить три параметра: символ a_i из выбранного алфавита А, направление перемещения каретки («←» – влево, «→» – вправо, «точка» – нет перемещения) и новое состояние автомата q_k.

Например, команда $1 \leftarrow q_2$ обозначает «заменить символ на 1, переместить каретку влево на одну ячейку и перейти в состояние q_2».

Способы представления машины Тьюринга. Существует три способа представления машины Тьюринга: совокупностью команд, в виде графа, в виде таблицы соответствия.

Порядок выполнения работы

Выполнить задание, приведенное в таблице 8.1, согласно варианту.

Таблица 8.1 – Варианты индивидуальных заданий

Вариант	Представить МТ таблицей соответствия для выполнения операции
1	На ленте МТ содержится последовательность символов «+». Напишите программу, которая каждый второй символ «+» заменит на «–». Замена начинать с правого конца последовательности. В состоянии q_1 обозревается один из символов заданной последовательности
2	Дано число n в восьмеричной системе счисления. Разработать МТ, которая увеличивала бы заданное число n на 1. В состоянии q_1 обозревается некая цифра входного слова
3	Дано десятичное натуральное число n. Разработать МТ, которая уменьшала бы это число на 1. В состоянии q_1 обозревается правая цифра числа
4	Дано натуральное число n. Разработать МТ, которая уменьшала бы заданное число на 1, при этом в выходном слове старшая цифра не должна быть 0. Например, если входное слово 100, то выходным должно быть 99, но не 099. В состоянии q_1 обозревается правая цифра числа
5	Дана последовательность открывающих и закрывающих скобок. Построить МТ, которая удаляла бы пары взаимных скобок, т. е. расположенных подряд ( ). Например, дано) (( ) (( ), надо получить)) . . . (. В состоянии q_1 обозревается крайний левый символ строки
6	Дана строка из букв a и b. Разработать МТ, которая переместит все буквы a в левую, а буквы b – в правую часть строки. В состоянии q1 обозревается крайний левый символ строки
7	На ленте МТ записано десятичное число. Умножить это число на 2. В состоянии q_1 обозревается крайняя левая цифра числа
8	Даны два натуральных числа m и n в унарной системе счисления, разделенные пустой ячейкой. В состоянии q_1 обозревается самый правый символ входной последовательности. Разработать МТ, которая на ленте оставит сумму чисел m и n
9	Даны два натуральных числа m и n в унарной системе счисления, разделенные пустой ячейкой. В состоянии q_1 обозревает самый правый символ входной последовательности. Разработать МТ, которая на ленте оставит разность чисел m и n. Известно, что m > n
10	На ленте МТ находится десятичное число. Определить, делится ли это число на 5 без остатка. Если делится, то записать справа от числа слово «да», иначе – «нет». В начальном состоянии обозревается некоторая цифра входного числа
11	Дано число n в четверичной системе счисления. Разработать МТ, которая увеличивала бы заданное число n на 1. В состоянии q_1 обозревается некая цифра входного слова
12	Дано десятичное натуральное число n. Разработать МТ, которая уменьшала бы это число на 1. В состоянии q_1 обозревается некая цифра числа

Контрольные вопросы

1 Что представляет собой машина Тьюринга?
2 Какие основные элементы входят в состав МТ?

3 Прокомментируйте структурную схему МТ.
4 Сформулируйте формальное определение МТ.
5 Что такое входной алфавит ленты МТ?
6 Что такое функция переходов (программа) МТ?
7 Прокомментируйте структуру команды МТ.
8 В каких состояниях может находиться МТ?
9 Прокомментируйте команду МТ.
10 Чем управляется программируемый автомат МТ в процессе работы?

9 Лабораторная работа № 9. Программирование машины Тьюринга

Машина Тьюринга представляет собой простейшую вычислительную машину с линейной памятью, которая согласно формальным правилам преобразует входные данные с помощью последовательности элементарных действий. Элементарность действия заключается в том, что оно меняет лишь небольшой кусочек данных – лишь одну ячейку, а число возможных действий конечно. Несмотря на простоту МТ, на ней можно вычислить все, что можно вычислить на любой другой машине, осуществляющей вычисления с помощью последовательности элементарных действий. Это свойство МТ называется полнотой.

На машине Тьюринга с помощью задания правил перехода можно имитировать всех других исполнителей, реализующих процесс пошагового вычисления, в котором каждый шаг вычисления достаточно элементарен.

Существуют программы для обычных компьютеров, имитирующие работу машины Тьюринга. Но следует отметить, что данная имитация неполная, т. к. в машине Тьюринга присутствует абстрактная бесконечная лента. Бесконечную ленту с данными невозможно в полной мере имитировать на компьютере с конечной оперативной памятью: памятью жёстких дисков и различных внешних носителей, регистров и кэш-процессора и др., которая может быть очень большой, но, тем не менее, всегда конечной.

Одним из вариантов программы, имитирующей работу МТ, является детерминированная машина Тьюринга, интерфейс которой представлен на рисунке 9.1.

Интерфейс программы содержит следующие элементы: «Лента»; текстовый блок «Состояние» с текстовым полем и кнопкой «Установить»; блок «Множество состояний»; текстовое поле «Конфигурация» с кнопкой «Установить»; блок «Алфавит»; блок «Команды» и блок «Краткое руководство».

Элемент «Лента» содержит движок, позволяющий перемещать содержимое ленты вправо-влево.

Блок «Множество состояний» содержит 10 флажков для выбора состояний МТ и текстовые поля для ввода значений состояний, а также кнопку «Еще» для добавления текстовых полей состояний.

Блок «Алфавит» содержит флажки «Цифры», «Буквы», «Символы»;

10 флажков для выбора цифр; 26 флажков для выбора латинских букв; кнопки «B», «<», «>», «=», «+», «-», «*», «/», «^», «%», а также 14 текстовых полей для ввода других символов.

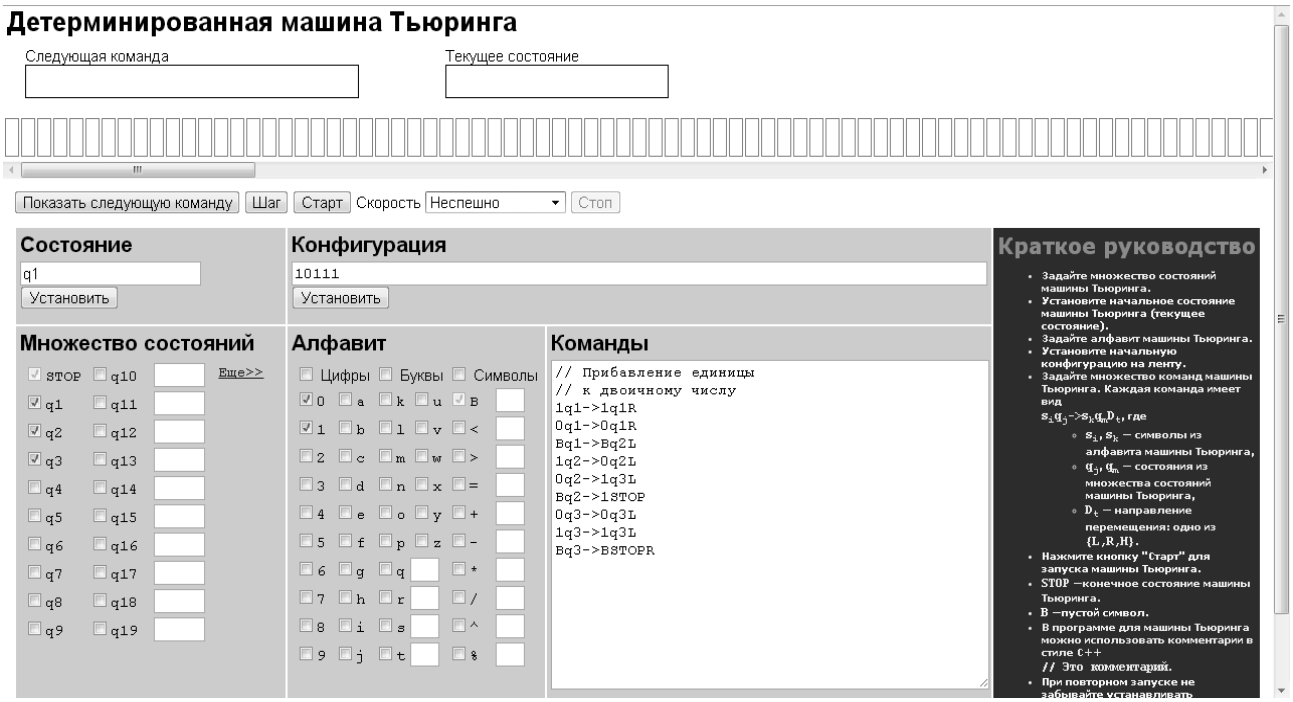


Рисунок 9.1 – Детерминированная машина Тьюринга

Блок «Команды» содержит окно кода для ввода команд программы МТ.

Блок «Краткое руководство» содержит краткую инструкцию по работе с детерминированной машиной Тьюринга.

Кнопка «Показать следующую команду» используется для просмотра следующей после только что исполненной команды.

Кнопка «Шаг» предназначена для покомандного выполнения программы, а «Старт» – для запуска программы на выполнение в автоматическом режиме.

Поле со списком «скорость» позволяет выбрать скорость выполнения программы: мгновенно, очень быстро, быстро, неспешно, медленно или очень медленно.

Кнопка «Стоп» служит для останова программы.

Перед запуском программы необходимо отметить необходимо для работы программы начальное состояние, выбрать символы алфавита, используемые в программе и ввести в окно кода заранее составленную программу.

После этого в блоке «Состояние» интерфейса МТ нажать кнопку «Установить», чтобы установить ее начальное состояние, а в блоке «Конфигурация» нажать кнопку «Установить», чтобы установить начальную конфигурацию (начальное слово) на ленту.

После успешного выполнения перечисленных действий, используя кнопку «Шаг», можно проанализировать исполнение команд программы и выполнить ее отладку.

Порядок выполнения работы

Разработать программу для МТ, реализующую задание (таблица 9.1).

Таблица 9.1 – Варианты индивидуального задания

Вариант	Представить МТ таблицей соответствия для выполнения арифметической
1	111111+1 в унарной системе счисления
2	101110+1 в бинарной системе счисления
3	11111+11 в унарной системе счисления
4	10111011+11 в бинарной системе счисления
5	1001011+10 в бинарной системе счисления
6	10111011+11 в бинарной системе счисления
7	1111011+11 в унарной системе счисления
8	10100110+1 в бинарной системе счисления
9	1111+11 в унарной системе счисления
10	101011011+1 в бинарной системе счисления
11	1111011+11 в бинарной системе счисления
12	111001011+11 в бинарной системе счисления

Контрольные вопросы

1 Назовите основные элементы МТ?
2 Сколько состояний имеет управляющее устройство МТ?
3 Какое состояние МТ означают останов работы ее алгоритма?
4 Какая МТ называется детерминированной?
5 Перечислением каких множеств задается МТ?
6 Прокомментируйте структуру правила (команды) МТ.
7 Как задается направление движения головки МТ после выполнения команды?
8 Прокомментируйте МТ, выполняющей умножение чисел.
9 Что такое протокол работы МТ?
10 Прокомментируйте МТ, выполняющую прибавление единицы к последней цифре на ленте.

10 Лабораторная работа № 10. Изучение формальных грамматик и их свойств

Грамматика – это математическая система, определяющая язык или способ построения языка, т. е. описания правил вывода его цепочек.

Одним из способов задания грамматики является ее формальное описание, построенное на основе системы правил, называемых также продукциями, в следующем виде: G{V_T, V_N, P, S}, где V_T – множество или алфавит терми-

нальных символов; V_N – множество или алфавит нетерминальных символов; P – множество правил вывода (продукций) грамматики вида $\alpha \rightarrow \beta$ (читается: из α выводится β), где $\alpha \in (V_N\ V_T)^+$, $\beta \in (V_N\ V_T)^*$; S – начальный (целевой) символ грамматики, называемый также аксиомой, $S \in V_N$.

Алфавиты терминальных и нетерминальных символов грамматики не пересекаются: $V_N \cap V_T = \varnothing$. Это значит, что каждый символ в грамматике может быть либо терминальным, либо нетерминальным, но не может быть терминальным и нетерминальным одновременно. Начальный символ грамматики – это всегда нетерминальный символ.

Множество терминальных символов V_T содержит символы, которые входят в алфавит языка, порождаемого грамматикой. Как правило, символы из множества V_T встречаются только в цепочках правых частей правил вывода.

Для РБНФ терминальные символы – это либо предопределенные идентификаторы, либо последовательности символов в кавычках и апострофах.

Нетерминальные символы V_N – это элементы грамматики, имеющие собственные имена и структуру. Каждый нетерминальный символ состоит из одного и более терминальных и/или нетерминальных символов, сочетание которых определяется правилами грамматики. Нетерминальные символы определяют слова, понятия и конструкции языка. Каждый символ этого множества может встречаться как в левых, так и в правых частях правил вывода, но должен хотя бы один раз быть в левой части каждого правила.

Правила грамматики – это упорядоченные пары цепочек символов (α, β), соединенных символом вывода «→». В правилах важен порядок цепочек, поэтому их чаще записывают в виде $\alpha \rightarrow \beta$ (или в форме БНФ α ::= β). Такая запись читается так: «из α выводится β» или «α порождает β», или «α по определению есть β».

Во множестве правил грамматики могут быть правила, имеющие одинаковые левые части, т. е. $\alpha \rightarrow \beta_1$, $\alpha \rightarrow \beta_2$, …, $\alpha \rightarrow \beta_n$. Тогда эти правила объединяют вместе и записывают в виде: $\alpha \rightarrow \beta_1 \mid \beta_2 \mid \ldots \mid \beta_n$, т. е. одной строке в такой записи соответствует сразу n правил.

Выводом цепочки называется процесс порождения предложения языка на основе правил, определяющих язык грамматики.

Цепочка $\beta = \delta_1\gamma\delta_2$ называется непосредственно выводимой из цепочки $\alpha = \delta_1\omega\delta_2$ в грамматике $G(V_T, V_N, P, S)$, $V = V_N \cup V_T$, $\delta_1, \gamma, \delta_2 \in V^*$, $\omega \in V^+$, если в грамматике существует правило $\omega \rightarrow \gamma$ P. Иными словами, цепочка β выводима из цепочки α в том случае, если можно взять несколько символов в цепочке α, поменять их на другие символы, согласно некоторому правилу грамматики, и получить цепочку β. Непосредственная выводимость цепочки β из цепочки α обозначается как $\alpha \Rightarrow \beta$.

Цепочка β называется выводимой из цепочки α ($\alpha \Rightarrow^* \beta$) в случае, если выполняется одно из двух условий:

1) β непосредственно выводима из α ($\alpha \Rightarrow \beta$);

2) существует γ такая, что γ выводима из α, а β непосредственно выводима из γ ($\alpha \Rightarrow^* \gamma$, $\gamma \Rightarrow \beta$).

Например, грамматика для языка целых десятичных чисел со знаком:

G ({ 0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , – , +}, {S, T, F}, P, S)
P:
S → T | +T | –T
T → F | TF
F → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Построим несколько цепочек вывода в этой грамматике:
1) S ⇒ –T ⇒ –TF ⇒ –TFF ⇒ –FFF ⇒ –4FF ⇒ –47F ⇒ –479;
2) S ⇒ T ⇒ TF ⇒ T8 ⇒ F8 ⇒ 18;
3) T ⇒ TF ⇒ T0 ⇒ TF0 ⇒ T50 ⇒ F50 ⇒ 350;
4) F ⇒ 5.

Получаем, что S ⇒* –479, S ⇒* 18, T ⇒* 350, F ⇒* 5.

Порядок выполнения работы

Используя заданные в таблице 10.1 правила вывода, построить вывод заданной цепочки.

Таблица 10.1 – Варианты заданий

Вариант	Задание
1	Построить правила грамматики для вывода цепочки x – 5.7*x – 2*a
2	Дана грамматика с правилами вывода: S→T | T+S | T–S; T→F | F*T; F→a | b *b Построить вывод цепочки a – b*a + b*b и дерево вывода
3	Построить правила грамматики для вывода цепочки 79547*d
4	Дана грамматика с правилами вывода: S → aSBC | abC; CB → BC; B → bb C→bc; cC → cc. Построить вывод цепочки aaabbbccc и дерево вывода
5	Построить правила грамматики для вывода цепочки – 47082*c
6	Дана грамматика с правилами вывода: S→T | T/S | T–S; T→F | F/T; F→a | b*b Построить вывод цепочки a – b – a/b*b и дерево вывода
7	Построить правила грамматики для вывода цепочки a*x + 4*x – b
8	Дана грамматика с правилами вывода: S → aSBC | abC; CB → BC; B → bb C→bc; cC → cc. Построить вывод цепочки aaabbbcccccc и дерево вывода
9	Построить правила грамматики для вывода цепочки a + 2*x – 4*b
10	Дана грамматика с правилами вывода: S → BD B → aBbC | ab Cb → bC CD → Dc bDc → bcc abD → abc. Построить вывод цепочки aaaabbCbCbCD и дерево вывода
11	Построить правила грамматики для вывода цепочки a + 5/b + 2*b – x
12	Дана грамматика с правилами вывода: S→T | T+S | T–S; T→F | F*T; F→a | b *b Построить вывод цепочки a – a*b + b*b*a и дерево вывода

Контрольные вопросы

1 Что понимается под цепочкой вывода?
2 Как построить правила грамматики для вывода заданной цепочки?

3 Прокомментируйте структуру выражения, определяющего грамматику.
4 Как классифицируются грамматики по способу задания языка?
5 Как обозначается выводимость цепочки?
6 Какие алфавиты используются в формальных грамматиках?
7 Что такое начальный символ грамматики?
8 Прокомментируйте первое правило вывода грамматики.
9 Что понимается под непосредственной выводимостью цепочки?

11 Лабораторная работа № 11. Изучение классификации формальных грамматик, языков и их свойств

Классификация формальных грамматик, языков и распознавателей. Согласно классификации, предложенной американским лингвистом Н. Хомским, формальные грамматики классифицируются по структуре их правил. Если все без исключения правила грамматики удовлетворяют некоторой заданной структуре, то такую грамматику относят к определенному типу. Достаточно иметь в грамматике одно правило, не удовлетворяющее требованиям ее структуры, и она уже не попадает в данный тип. Хомский выделил четыре следующих типа формальных грамматик [2].

Тип 0. Грамматики с фразовой структурой. На структуру правил данной грамматики не накладывается никаких ограничений: для грамматики вида $G(V_T, V_N, P, S)$, $V = V_N \cup V_T$, правила имеют вид $\alpha \rightarrow \beta$, где $\alpha \in V^+$, $\beta \in V^*$. Это самый общий тип грамматик.

Тип 1. Контекстно-зависимые (КЗ) грамматики. В этот тип входят два основных класса грамматик:

1) контекстно-зависимые грамматики $G(V_T, V_N, P, S)$, $V = V_N \cup V_T$ имеют правила вывода вида $\alpha A \beta \rightarrow \alpha \gamma \beta$, где $\alpha, \beta \in V^*$, $\gamma \in V^+$, $A \in V_N$,

2) неукорачивающие грамматики $G(V_T, V_N, P, S)$, $V = V_N \cup V_T$ имеют правила вывода вида $\alpha A \beta \rightarrow \alpha \gamma \beta$, где $\alpha, \beta \in V^*$, $\gamma \in V^+$, $A \in V_N$ и $|A| <= |\gamma|$.

Структура правил КЗ грамматик такова, что при построении предложений заданного ими языка один и тот же нетерминальный символ может быть заменен в ту или иную цепочку символов в зависимости от того контекста, в котором он встречается.

Тип 2. Контекстно-свободные (КС) грамматики. Неукорачивающие контекстно-свободные (НКС) грамматики $G(V_T, V_N, P, S)$, $V = V_N \cup V_T$ имеют правила вида $A \rightarrow \beta$, где $A \in V_N$, $\beta \in V^+$. Такие грамматики называют НКС-грамматиками, поскольку видно, что в правой части правил у них должен всегда стоять как минимум один символ.

Левая часть правила КС грамматики состоит из одного нетерминала.

Существует также почти эквивалентный им класс грамматик – укорачивающие контекстно-свободные (УКС) грамматики, правила которых могут иметь вид $A \rightarrow \beta$, где $A \in V_N$, $\beta \in V^*$.

Тип 3. Регулярные грамматики. К типу регулярных относятся два эквива-

лентных класса грамматик: леволинейные и праволинейные.

Леволинейные грамматики $G(V_T, V_N, P, S)$, $V = V_N \cup V_T$ имеют правила двух видов: $A \to B\gamma$ или $A \to \gamma$, где $A, B \in V_N$, $\gamma \in V_T^*$, т. е. при выводе нетерминальный символ если и остается, то слева. Праволинейные грамматики $G(V_T, V_N, P, S)$ тоже имеют правила двух видов: $A \to \gamma B$ или $A \to \gamma$, где $A, B \in V_N$, $\gamma \in V_T^*$ – при выводе нетерминальный символ остается справа.

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

С формальными грамматиками тесно связаны такие понятия, как языки и распознаватели, классификация которых представлена на рисунке 11.1.

Классификация языков. Языки классифицируются в соответствии с типами грамматик, с помощью которых они задаются.

Дерево вывода. Деревом вывода грамматики называется дерево или граф, которое соответствует некоторой цепочке вывода.

Из определения видно, что по структуре правил дерево вывода в указанном виде всегда можно построить только для контекстно-свободных и регулярных грамматик. Для грамматик других типов дерево вывода в таком виде можно построить не всегда либо оно будет иметь несколько иной вид. Построить дерево вывода можно, имея только цепочку вывода.

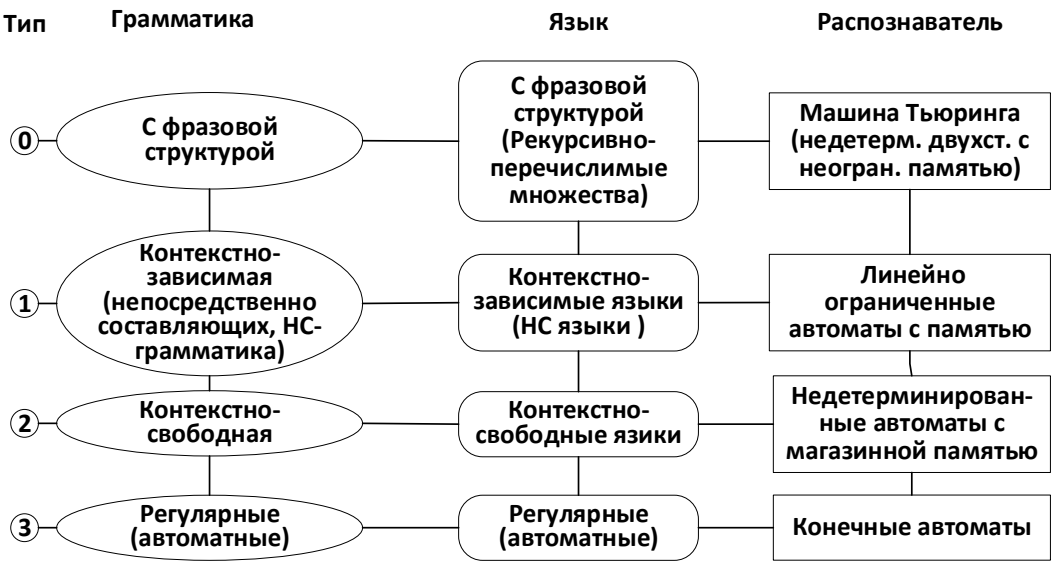


Рисунок 11.1 – Классификация грамматик, языков и распознавателей

На рисунке 11.2 представлены деревья вывода для цепочек –479 и 18.

Порядок выполнения работы

Разработать программу, реализующую задание согласно вариантц (таблица 11.1).

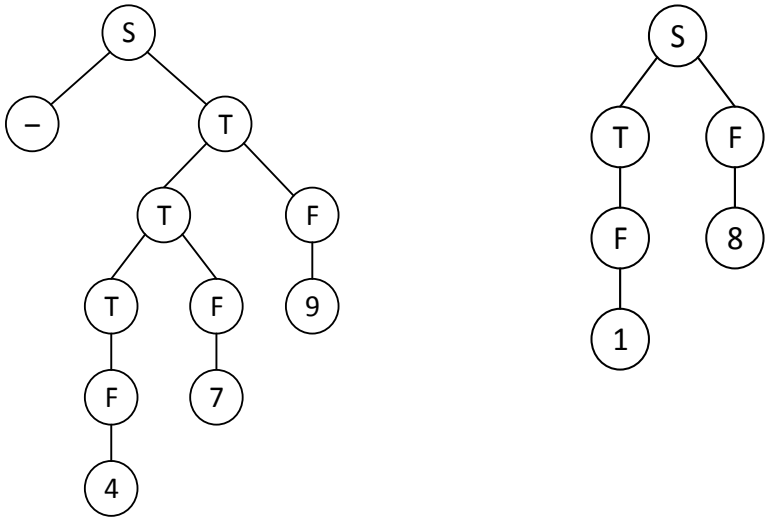


Рисунок 11.2 – Примеры деревьев вывода для целых десятичных чисел

Таблица 11.1 – Варианты заданий

Вариант	Задание	
	1	2
1	Дана грамматика с правилами вывода: S → T | T+S | T–S; T → F | F*T; F → a | b *b Построить вывод цепочки a – b*a + b*b и дерево вывода	Дана грамматика с правилами вывода: S → T | T/S | T–S; T→F | F/T; F → a | b*b Построить вывод цепочки a – b – a/b*b и дерево вывода
2	Дана грамматика с правилами вывода: S → aSBC | abC; CB → BC; B → bb C → bc; cC → cc Построить вывод цепочки aaabbbccc и дерево вывода	Дана грамматика с правилами вывода: S → aSBC | abC; CB → BC; B → bb C → bc; cC → cc Построить вывод цепочки aaabbbcccccc и дерево вывода
3	Построить все сентенциальные формы для грамматики с правилами S → A+B | B+A; A → a; B → b	Построить все сентенциальные формы для грамматики с правилами S → A–B | B–A; A → a; B → b
4	К какому типу по Хомскому относится грамматика с правилами S → a | Ba; B → Bb | b	К какому типу по классификации Хомского относится грамматика с правилами S → Ab; A → Aa | ba
5	К какому типу по Хомскому относится грамматика с правилами: S → 0A1 | 01; 0A → 00A1; A → 01	К какому типу по Хомскому относится грамматика с правилами S → AB; AB → BA; A → a; B → b
6	Построить все сентенциальные формы для грамматики с правилами S → A+B | B+A; A → a; B → b	К какому типу по Хомскому относится грамматика с правилами S → a | Ba; B → Bb | b

Контрольные вопросы

1 Что понимается под цепочкой символов?
2 Как обозначается длина цепочки символов?
3 Какие операции можно производить с цепочками символов?

4 Что такое конкатенация двух цепочек?
5 Что такое замена (подстановка) подцепочек?
6 Что такое обращение цепочек?
7 Что такое итерация цепочек?
8 Что такое пустая цепочка?
9 Какие алфавиты используются в формальных грамматиках?

12 Лабораторная работа № 12. Изучение контекстно-свободных грамматик

Контекстно-свободной грамматикой называется грамматика, у которой в левых частях всех правил стоят только одиночные нетерминальные символы. Контекстно-свободный язык – это язык, задаваемый контекстно-свободной грамматикой.

Грамматический разбор текста контекстно-свободной грамматики начинается с его синтаксического анализа (парсинга) и выполняется совместно с лексическим анализом, а его результатом является дерево разбора, которое, например, для выражения ″1 + 2*3″ имеет вид, представленный на рисунке 12.1.

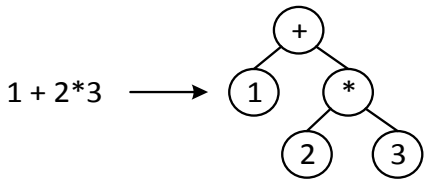


Рисунок 12.1 – Пример дерева разбора выражения

При обработке парсером исходного текста он преобразуется в структуру данных, обычно в дерево, которое отражает синтаксическую структуру входной последовательности и хорошо подходит для дальнейшей обработки.

Синтаксис языка удобно представлять в виде синтаксического графа или графа распознавания, который отражает управление ходом грамматического анализа предложений.

Синтаксический граф является эквивалентным представлением грамматики языка, а также удобной формой грамматического анализа предложений и во многих случаях оказывается предпочтительнее БНФ. Кроме того, граф дает более ясное и точное представление о структуре языка и позволяет лучше отобразить процесс грамматического разбора.

На рисунке 12.2 представлен синтаксический граф языка предложений вида x, (x), (x+x), ((x)) и так далее.

Одним из назначений грамматики языка является построение вывода слов (лексем, токенов).

Вывод может быть левосторонним и правосторонним. Левосторонним выводом слова α называется такой его вывод, в котором каждая последующая строка получена из предыдущей путем замены по одному из правил

самого левого встречающегося в строке нетерминала, а правосторонним – самого правого.

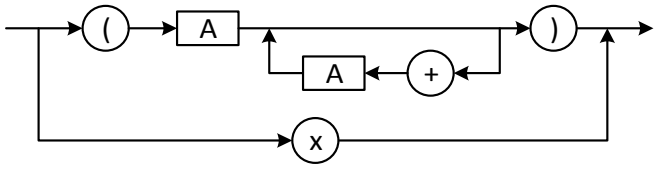


Рисунок 12.2 – Синтаксический граф предложения

Пример – Рассмотрим грамматику G_0, выводящую все правильные скобочные последовательности:

$$G_0 = \{V_T, V_N, P, S\},$$

где V_T = "("и")" – терминальные символы;

S – начальный нетерминальный символ;

P – правила грамматики: $S \to (S)S$; $S \to S(S)$; $S \to \varepsilon$.

Здесь ε – пустой символ.

Выведем цепочку «(()(()))()»:

$S \Rightarrow (S)S \Rightarrow (S)(S)S \Rightarrow (S)()S \Rightarrow (S)() \Rightarrow (S(S))() \Rightarrow (S(S)(S))() \Rightarrow (S(S)(S(S)))() \Rightarrow (S(S)((S)))() \Rightarrow (S()((S)))() \Rightarrow (()((S)))() \Rightarrow (()(()))()$

Деревом разбора грамматики называется графическая структура, представленная в виде дерева, в вершинах которого содержатся терминалы или нетерминалы. Все вершины, помеченные терминалами, называются листьями, а вершины, помеченные нетерминалами, содержат детей. Дети вершин, в которых записаны нетерминалы, могут быть раскрыты по одному из правил, в левой части которых содержится этот нетерминал, и упорядочены так же, как в правой части этого правила.

Крона дерева разбора – это множество терминальных символов, упорядоченное в соответствии с номерами их достижения при обходе дерева в глубину из корня. Крона дерева разбора представляет собой слово языка, которое выводит это дерева.

На рисунке 12.3 приведено дерево разбора скобочной последовательности, выводимой упомянутой выше грамматикой G_0.

Рассмотрим пример контекстно-свободной грамматики арифметического выражения (a + b) * c. Для этого запишем для него правила грамматики:

1) $Z \to +E \mid -E \mid E$;
2) $E \to T + E \mid T - E \mid T$;
3) $T \to F * T \mid F / T \mid F$;
4) $F \to I \mid (Z)$;
5) $I \to a \mid b \mid c \mid d$.

В приведенных правилах использованы следующие обозначения:

Z – выражение, E – выражение без знака, T – терм, F – фактор, I – идентификатор.

Кроме того, в формальной грамматике используются понятия левосторонней и правосторонней прямой рекурсии. В первом случае символ из левой части правила находится в начале его правой части, а во втором – в конце.

В предыдущем примере присутствует правосторонняя рекурсия. Доказано, что грамматику всегда можно преобразовать таким образом, что левосторонняя рекурсия будет заменена правосторонней и наоборот. Преобразуем грамматику предыдущего примера так, чтобы избавиться от правосторонней рекурсии. Для этого достаточно переписать второе и третье правила:

$E \rightarrow E+ T \mid E - T \mid T;$ $T \rightarrow T * F \mid T / F \mid F.$

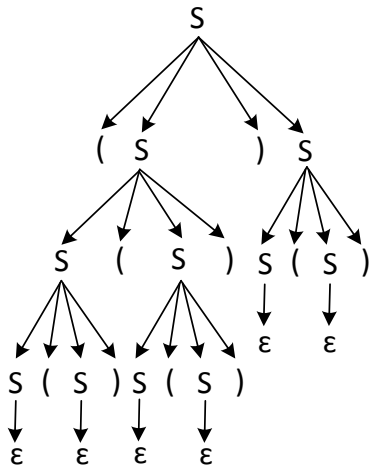


Рисунок 12.3 – Дерево разбора скобочной последовательности

Набор правил синтаксиса языка образует его грамматику и может описывать процедуры либо получения его правильных предложений, либо распознавания принадлежности этих предложений к данному языку. В первом случае грамматику называют порождающей, во втором – распознающей.

Порядок выполнения работы

1 В соответствии с полученным заданием (таблица 12.1) построить алгоритм грамматического разбора языка в виде синтаксического графа.

Таблица 12.1 – Варианты заданий

Вариант	Элемент синтаксиса	Вариант	Элемент синтаксиса
1	a + b / c; оператор if	7	a * c – b * c; оператор цикла for … next
2	a - b * c; оператор выбора	8	a / c – b / c; оператор цикла while … wend
3	a + b + c; оператор if … then	9	a / c – b / a; оператор цикла do while …
4	a * c + c * a; оператор if … then … else	10	a * b * c - a; оператор цикла for … each
5	a / c – b * c; оператор InputBox	11	a / c / c +a; оператор do … while
6	a * b – b / c; оператор MsgBox	12	a + b / c +b; оператор for … next [step]

2 По заданному синтаксическому графу построить несколько предложений языка, в том числе и неправильных, и проанализировать работу алгоритма.

3 Разработать приложение, реализующее построенный алгоритм.

Контрольные вопросы

1 Что понимается под грамматическим разбором текста?
2 Что является результатом грамматического разбора текста?
3 Что такое дерево грамматического разбора текста?
4 Для чего используется парсер – синтаксический анализатор?
5 Управление чем отражает синтаксический граф?
6 Как построить синтаксический граф арифметического выражения?
7 Как преобразовть синтаксический граф в блок-схему алгоритма?
8 Что понимается под выводом грамматики?
9 Что такое дерево разбора грамматики?

13 Лабораторная работа № 13. Изучение метасимволов и модификаторов регулярных выражений

Основой синтаксиса регулярных выражений является то, что некоторые символы, встречающиеся в тексте, рассматриваются не как обычные литералы, а как метасимволы**,** имеющие специальные назначения.

Метасимволы – это специальные символы, из которых формируют шаблоны поиска и выполнения различных манипуляций с текстом: замена, вставка, удаление различных слов или словосочетаний.

Простейшими примерами метасимволов могут быть символы "*" и "?", используемые при поиске файлов в Windows. Например, поиск файла по шаблону "text?.*" найдет файлы «text1.doc», «textf.txt», «texts.html» и другие аналогичные, но не найдет «text.txt» или «text.xls». В языках программирования регулярные выражения обладают более широкими возможностями.

Шаблон (pattern) регулярного выражения начинается с символа «"», за которым следует группа метасимволов, заканчивающаяся таким же символом «"» и может содержать необязательный список модификаторов, позволяющих задать дополнительные условия.

В C# допустимо использование модификатора как через шаблон регулярного выражения, так и через его опции.

Если в регулярном выражении литерал метасимвола должен восприниматься как обычный символ, то перед ним ставится символ ″\″ –обратный слеш. Этот приём называется экранированием. Например, чтобы представить в регулярном выражении символ «точка», следует написать "\.". Метасимвол «\» также может быть экранирован, т. е. представлен как ″\\″.

Обычный символ представляет в выражении сам себя, а метасимвол – некоторый класс символов. Значения основных метасимволов при-

ведены в таблице 13.1.

Модификаторы в регулярных выражениях – это последовательности метасимволов, которые задают режим поиска совпадений и изменяют режим работы модуля (парсера) RegExpr. В C# форма записи регулярного выражения с модификатором следующая: @"(?модификатор) регулярное выражение";.

Таблица 13.1 – Значения метасимволов

Мета-символ	Интерпретация
\	Экранирующий символ
.	Любой символ, кроме \n
^	Привязывает совпадения остальной части регулярного выражения к началу строки. Например,^cat совпадет со строками «cat» и «caterpillar». При использовании ^ внутри квадратных скобок после него указываются символы, которые не должны встречаться
$	Привязывает совпадения остальной части регулярного выражения к концу строки. Например, dog$ совпадет со строками ″bulldog″ и ″hotdog″
( )	Скобки для задания группировки
|	Разделитель альтернатив
{ }	Скобки для задания квантификаторов
[…]	Любой из класса символов, указанных в скобках
[^…]	Любой из класса символов, не указанных в скобках
*	Повторение предыдущего символа 0 и более раз
+	Повторение предыдущего символа 1 и более раз
?	Повторение предыдущего символа 0 или 1 раз

Модификаторы регулярных выражений приведены в таблице 13.2.

Таблица 13.2 – Модификаторы, используемые в C#

Моди-фикатор	Назначение
i	Регистронезависимый поиск
g	Глобальный поиск – обрабатываются все совпадения с шаблоном поиска
u	Поиск минимального по длине соответствия. По умолчанию ищется максимальное по длине соответствие
m	Поиск соответствия только в одной строке. По умолчанию – по всему тексту.
s	Противоположность модификатору m, т. е., указав данный модификатор, соответствие будут искать по всему тексту
x	При использовании данного модификатора пробельные символы будут игнорироватьс. т. е. можно написать хоть 100 пробелов и они будут опущены, если их не экранировать с помощью "\"

Область действия модификатора – от текущей позиции до конца регулярного выражения, т. е. если модификатор описан внутри группы, то он действует до конца этой группы. Примеры записи модификаторов:

– шаблон "a(?i)bc" найдет строки, начинающиеся со строчной буквы "a", за которой следует фрагмент "bc", т. е. будет найдено "abc", "aBC", "aBc", "abC";

– шаблон "a((?i)bc)d": действие модификатора(?i) ограничено пределами группы, поэтому буквы "a" и "d" должны быть строчные, т. к. они не входят в область действия модификатора, а строка "bc" не чувствительна к регистру, т. е. будет найдено "abcd", "aBcd", "abCd";

– шаблон "(?i) float" отыщет "float", "Float", "FLOAT" и другие, поэтому модификатор (?i) удобен для анализа текстов некоторых программ.

Порядок выполнения работы

1 Для заданного варианта (таблица 13.3) построить регулярное выражение.

2 Разработать и отладить программу на C# для работы с построенным регулярным выражением.

Таблица 13.3 – Варианты заданий

Вари-ант	Построить регулярное выражение для поиска
1	Номера мобильного телефона следующего формата: +375 (XXX)NNN-NN-NN, где: XXX может быть 291…299; 336
2	Номера автомобиля в формате: BXX-YYMM, где B – буква A…R; XX, YY – любые цифры; MM – код обл.
3	Даты в формате ДД.ММ.ГГ или ДД/ММ/ГГГГ, где ДД – день; ММ – месяц; ГГ – год
4	Арифметического выражения ИМЯ=a*b±x^t, где ИМЯ, a, b, t – любые идентификаторы (латинские буквы от 1 до 5 латинских букв и цифр, первая буква)
5	Времени суток в формате TT.NN.CC или ММ.ГГ-NN.CC, где TT – часы, NN – минуты, CC – секунды: ММ – номер месяца; ГГ – год
6	Вещественного числа с/без знака ±XX..X,УУУ, где XX..X – целая часть, УУУ – дробная часть (3 цифры)
7	Вещественного числа в формате ±X.XX..XE±УУ
8	Арифметического выражения ИМЯ=a±b*x/t, где ИМЯ, a, b, t – любые идентификаторы (латинские буквы от 1 до 5 латинских букв и цифр, первая буква)
9	Арифметического выражения ИМЯ[k]=x+p[i]/y[i] где ИМЯ, k, p, y – любые идентификаторы (латинские буквы от 1 до 4 латинских букв и цифр, первая буква)
10	Адреса URL в формате www.http//xx..x.mmm. ... .sss, где домены xx..x.mmm,...,sss – латинские буквы в группе до 7 латинских букв; до 4 доменов
11	Адреса E-mail в формате nm@ee.ss, где mm – имя до 7 букв и цифр, ee – (tut | mail | gmail); ss – (by | ru | com)
12	IP – адреса в формате XXX.YYY.ZZZ.TTT, где каждая группа цифр в диапазоне 0…255

Контрольные вопросы

1 Приведите примеры метасимволов, используемых при поиске файлов.
2 Дайте определения регулярного выражения.
3 Для чего используются регулярные выражения?
4 Какие основные понятия используются в регулярных выражениях?
5 Что означает термин метасимвол в регулярных выражениях?
6 Чем метасимволы отличаются от обычных символов?
7 Перечислите основные метасимволы регулярных выражений.
8 Как выполнить регистронезависимый поиск текста?
9 Для чего используются модификаторы в регулярных выражениях?
10 Что означают метасимволы ″^″ и ″$″ в регулярном выражении?

14 Лабораторная работа № 14. Изучение синтаксиса регулярных выражений

Синтаксис регулярных выражений предусматривает использование значительного количества элементов: метасимволы, модификаторы, квантификаторы и другие. Рассмотрим пример использования регулярного выражения для нахождения телефонного номера в формате 111-111-1111:

```
strings = "456-435-2318";
Regexregex = newRegex(@"\d{3}-\d{3}-\d{4}");
```

В данном примере предполагается, что мы точно знаем, сколько определенных символов должно быть в каждой группе цифр, поэтому мы явным образом указали их количество в фигурных скобках, например, \d{3} – три цифры.

После создания шаблона регулярного выражения с ним можно осуществить различные действия. Например, метод IsMatch() позволяет проверить, существует ли в исходной строке текст, соответствующий шаблону, а метод Replace() – извлечь совпадения в исходной строке и заменить их новыми значениями. В таблице 14.1 показано, как формируются метасимволы замены.

Таблица 14.1 – Метасимволы замены в регулярных выражениях C#

Символ	Описание	Пример шаблона	Пример шаблона замены	Результат (входная → результирующая строки)
1	2	3	4	5
$ number	Замещает часть строки, соответствующую группе number	\b(\w+)(\s)(\w+)\b	$3$2$1	"один два" → "два один"
$$	Подставляет литерал "$"	\b(\d+)\s?USD	$$1	"103 USD" → "$103"

Окончание таблицы 14.1

1	2	3	4	5
$&	Замещает копией полного соответствия	(\$*(\d*(\.+\d+)?){1})	**$&	"$1.30" → "**$1.30**"
$`	Замещает весь текст входной строки до соответствия	B+	$`	"AABBCC" → "AAAACC"
$'	Замещает весь текст входной строки после соответствия	B+	$'	"AABBCC" →"AACCCC"
$+	Замещает последнюю захваченную группу	B+(C+)	$+	"AABBCCDD" → "AACCDD"
$_	Замещает всю входную строку	B+	$_	"AABBCC" →"AAAABBCCCC"

Нередко возникает задача проверить корректность данных, введенных пользователем. Это может быть проверка, например, электронного адреса или номера телефона. Класс Regex предоставляет статический метод IsMatch(), который позволяет проверить входную строку на соответствие шаблону:

```
string pattern = @"^(?("")(""[^""]+?)|(([0-9a-z]((\.(?!\.))|"+
@"[-!#\$%&'\*\+/=\?\^`\{\}\| ~\w])*)(?<=[0-9a-z])@))(?(\[)"+
@"(\[(\d{1,3}\.){3}\d{1,3}\])|(([0-9a-z][-\w]*[0-9a-z]*\.)+[a-z0-9]{2,17}))$";
while (true){
Console.WriteLine("Введитеадресэлектроннойпочты");
string email = Console.ReadLine();
if (Regex.IsMatch(email, pattern, RegexOptions.IgnoreCase)){
Console.WriteLine("Email подтвержден");
break;
            }
else
Console.WriteLine("Некорректный email");
}
Console.ReadLine();
```

Результат:

```
Введите адрес электронной почты
bru@by
Некорректный email
Введите адрес электронной почты
asu@bru.by
Email подтвержден
```

В данном примере переменная pattern задает регулярное выражение, предложенное Microsoft на страницах msdn, для проверки адреса электронной почты.

Порядок выполнения работы

Составить программное приложение, выполняющего действие, которое указано в регулярном выражении (таблица 14.2).

Таблица 14.2 – Варианты заданий

Вариант	Действие, выполняемое регулярным выражением
1	Проверка на корректность формата введенного пользователем десятичного числа со знаком или без знака в формате с фиксированной точкой: целая и дробная части разделяются точкой или запятой
2	Проверка на корректность формата введенного пользователем вещественного десятичного числа со знаком или без знака, записанного в экспоненциальной форме
3	Проверка на корректность формата введенного адреса e-mail
4	Проверка на корректность введенного IP-адреса (от 0.0.0.0 до 255.255.255.255)
5	Проверка на корректность формат введенного пользователем шестнадцатеричного числа. Символы должны быть нечувствительны к регистру. Если число начинается с буквы, то перед ней обязательно должен стоять ноль, например, вместо FFзаписано 0FF
6	Проверка корректности введенного времени в формате час.мин.сек (часы от 0 до 23, минуты и секунды от 0 до 59)
7	Разделение URL на составные части: протокол (например, HTTP), имя сайта, путь к каталогу на сайте, имя файла. Учесть, что некоторые части могут отсутствовать.
8	Выделение 10-значного номера сотового телефона, который начинается с трех введенных пользователем цифр, из заданной последовательности, например 8097653728936276637382…
9	Разделение полного пути к файлу на диске на составляющие (имя диска, путь к каталогу, имя, расширение файла). Учесть, что некоторые части могут отсутствовать
10	Копирование и вывода на экран всех тегов <ahref= …>и <img …> из произвольного HTML-файла
11	Выделение 10-значного номера сотового телефона, который начинается с четырех введенных пользователем цифр, из заданной последовательности, например 57428097653728936276637…
12	Вывод на экран номеров строк, в которых встречаются в заданном файле теги, вводимые пользователем

Контрольные вопросы

1 Что означает термин «метасимвол» в регулярном выражении?
2 Какие метасимволы используются в регулярных выражениях?
3 Какие модификаторы регулярных выражений Вы знаете?
4 Какие метасимволы замены в регулярных выражениях Вы знаете?
5 Какие методы класса Regex Вы знаете?
6 Для чего используется метод IsMatch?
7 Чем отличаются метасимволы от обычных символов?
8 Какие метасимволы регулярных выражений Вы знаете?
9 Какие методы класса Regex Вы знаете?
10 Для чего используется метод IsMatch() класса Regex?

15 Лабораторная работа № 15. Изучение технологии построения регулярных выражений

Рассмотрим технологию построения регулярных выражений на примере квантификаторов.

Квантификаторы. Если в шаблоне регулярных выражений встречаются символы *, +, ?, { и }, то обработчик регулярных выражений интерпретирует их как кванторы (квантификаторы) или как часть их конструкций, если они не включены в класс символов. Кванторы (повторители, или умножители) – это метасимволы, позволяющие задать количество повторений символа или группы символов в строке. Они ограничиваются парой фигурных скобок { }, располагаются непосредственно после обычного символа, символьного класса или группы и указывают, сколько раз он повторяется.

Виды квантификаторов перечислены в таблице 15.1, в которой n и m – целые числа.

Таблица 15.1 – Квантификаторы

Квантификатор	Описание	Пример
*	Ноль или более повторений предыдущего элемента	"ca*t" → ct, cat, caat, …
+	Один или более повторений предыдущего элемента	"ca+t" → cat, caat, caaat, …
?	Ноль или одно повторение предыдущего элемента	"ca?t"→ ct, cat
{n}	Точный – выражение повторяется ровно n раз	"\d{3}" трехзначное число
{n,m}	Конечный – выражение может повторяться от n до m раз. Пробелы в скобках не допускаются	"\d{2,3}" → двух- или трехзначное число
{n,}	Бесконечный – выражение может повторяться от n раз до бесконечности	"ab{2,}" → находит «abb», «abbb», «abbbb» и так далее

Квантификаторы имеют два режима работы: жадный (greedy) – по умолчанию и ленивый (lazy) – не жадный. Квантификатор называется жадным, если из всех возможных подстрок, соответствующих шаблону поиска, он выбирает максимально длинную. Или жадная квантификация – это стремление захватить максимально длинную строку, которая соответствует шаблону.

Когда повторять уже нельзя, например, нет больше цифр для \d+, парсер продолжает поиск в оставшейся части текста. Если совпадение найти не удалось – отступает обратно, уменьшая количество повторений.

Ленивым, или не жадным, называют квантификатор, который прекращает поиски, как только найдет первую подходящую подстроку минимальной длины, т. е. ленивая квантификация – это стремление захватить максимально короткую строку, которая соответствует шаблону. Ленивость распространяется только на тот квантификатор, после которого следует "?".

Варианты жадных и ленивых квантификаторов приведены в таблице 15.2.

Пример – Найти в строке "hello! how are you?" по шаблону "h.*o" строку, начинающуюся с буквы "h", за которой следует несколько произвольных символов, а за ними буква "o".

Таблица 15.2 – Жадные и ленивые квантификаторы

Квантификатор		Выполняет
жадный	ленивый	
*	*?	Повторение ноль и более раз. То же, что и {0,}
+	+?	Повторение один и более раз. То же, что и {1,}
?	??	Повторение ноль или один раз. То же, что и {0,1}
{n}	{n}?	Повторение точно n раз
{n,m}	{n,m}?	Повторение не менее n и не более m раз
{n,}	{n,}?	Повторение не менее n раз

Порядок выполнения работы

Разработать и отладить консольное приложение на C# для указанного в таблице 15.3 варианта.

Таблица 15.3 – Варианты заданий

Вари-ант	Выбрать метод класса Regex для
1	Проверки содержатся ли в текстовом файле var1.txt имена Андрей и Наталья
2	Проверки содержатся ли в текстовом файле var2.txtназвания городов Минск и Якутск
3	Проверки содержатся ли в текстовом файле var3.txt названия государств Китай и Франция
4	Замены слова «зима», содержащегося в текстовом файле var4.txt, на «лето»
5	Замены слова «часы», содержащегося в текстовом файле var5.txt, на «таймер»
6	Замены слова «февраль», содержащегося в текстовом файле var6.txt, на «март»
7	Поиска слова «терминал» и его однокоренных слов в текстовом файле var7.txt. Результаты вывести на консоль красным цветом
8	Поиска слова «значение» и его однокоренных слов в текстовом файле var8.txt. Результаты вывести на консоль зеленым цветом
9	Поиска слова «тема» и его однокоренных слов в текстовом файле var9.txt. Результаты вывести на консоль синим цветом
10	Формирования массива строк, полученных в результате разделения текста, содержащегося в файле var10.txt, на фрагменты, разделенные символом «;»
11	Проверки, содержатся ли в текстовом файле var11.txt названия рек Днепр и Западная двина
12	Проверки, содержатся ли в текстовом файле var12.txt названия озер Боровое и Нарочь

Контрольные вопросы

1 Что такое квантификаторы в регулярных выражениях?
2 Назовите виды квантификаторов и сформулируйте правила их записи.
3 Для чего используются квантификаторы "*" и "+" ?
4 Что найдет регулярное выражение \(.*\) в строке «(12345)124(32521)»?
5 Прокомментируйте назначение квантификаторов {*n*,*m*} и {*n*,}.
6 Какие квантификаторы называют ленивыми?
7 Для чего используются квантификаторы "?" и "{n}" ?
8 Какие квантификаторы называют жадными?
9 Что означают квантификаторы a{2,5}, a{,5}, a{2,} ?

16 Лабораторная работа № 16. Изучение технологии использования регулярных выражений

В регулярных выражениях активно используются такие элементы, как группировки и обратные ссылки.

Группировки позволяют выделять отдельные части из строк уже найденных регулярным выражением. Например, метасимволы от \1 до \9 используются для обозначения найденных частей строк, а именно: \1 означает строку, найденную в первой группировке, \2 – во второй и так далее. Метасимволы \1 – \9 называются обратными ссылками, поскольку позволяют ссылаться в регулярном выражении на подстроки, найденные этим же регулярным выражением ранее.

Группировки используются также для задания альтернатив и выделения подстрок, которые нужно повторить заданное число раз совместно с квантификаторами. Например, шаблон "(123)+" найдет строки "123", "123123", "123123123" и так далее, а шаблон "(Кара){1,2}кум" найдет "Каракум", "Кара Каракум".

Более сложный пример – выделение доменного имени интернет-сайта. Доменное имя состоит из имен доменов первого, второго и так далее уровней, разделенных точками. Каждое имя может содержать латинские буквы, цифры, тире и знак подчеркивания. Шаблон "[_A-Za-z\d\–]+(\.[_A-Za-z\d\–]+)+" состоит из двух частей: первая часть описывает имя первого из доменов, а вторая – остальные имена через точку (группировка с повторителем "+").

Группировки часто используются для разделения найденной строки, соответствующей шаблону, на составные части.

Например, если из текста нужно выделить даты, записанные в формате dd/mm/yyyy. Шаблон для поиска "\d{2}/\d{2}/\d{4}" найдет в тексте "Срок действия вашего договора с 15/02/2011 по 27/06/2011" – две подчеркнутые подстроки. Если для последующей обработки из найденных подстрок необходимо выделить день, месяц или год по отдельности, то удобно использовать группировки.

Перепишем шаблон поиска: "(\d{2})/(\d{2})/(\d{4})", заключив день, месяц, год в скобки – получилось три группы. Функции языков программирования, выполняющие поиск по регулярным выражениям, как правило, возвращают найденные строки в массиве. Назовем такой массив, как Match[]. Тогда элемент Match[0] будет содержать всю найденную подстроку, элемент Match[1] – значение первой группы, Match[2] – значение второй группы и т. д. После первого поиска Match[0] = "15/02/2018", Match[1] = "15", Match[2] = "02", Match[3] = "2018", после второго поиска Match[0] = "27/06/2018", Match[1] = "27", Match[2] = "06", Match[3] = "2018". Так, с помощью групп можно выделить из найденной строки нужные части.

Если в строке записано вещественное десятичное число со знаком и нужно выделить знак, целую и дробную части, то для такой операции можно использовать следующий шаблон: "(\+|–)(\d+)\.(\d+)", в котором использованы три группы: знак, целая часть, дробная часть.

Рассмотрим распознавание магических последовательностей цифр: когда их последовательность повторяется друг за другом несколько раз, например, 123123, 454545 или 77777. Шаблон будет выглядеть так: "(\d+)\1+"". Здесь группировка ‘(\d+)’ ищет последовательность из одной или более цифр, а обратная ссылка ‘\1’ с квантификатором ‘+’ заставляет повторять эту последовательность ещё один или несколько раз.

Группировки можно использовать также для распознавания магических дат, например: 03/03/03 (3 марта 2003 г). Шаблон строится аналогично: ‘(\d{2})/\1/\1’. Группировка ″(\d{2})″ ищет двузначное число, а обратная ссылка \1 заставляет его подставлять на место месяца и года.

Порядок выполнения работы

Создать текстовый файл var.txt с исходным текстом, содержащим необходимые для выполнения варианта задания слова.

Разработать приложение в среде C# для указанного в таблице 16.1 варианта. Интерфейс приложения должен содержать элемент ComboBox для выбора вариантов заранее построенных регулярных выражений, кнопку и текстовое поле для его проверки, а также кнопку Стоп.

Таблица 16.1 – Варианты заданий

Вари-ант	Выбрать метод класса Regex для
1	Поиска слова «терминал» и его однокоренных слов в текстовом файле
2	Проверки, содержатся ли в текстовом файле названия планет Марс и Юпитер
3	Проверки, содержатся ли в текстовом файле названия озер Боровое и Нарочь
4	Проверки, содержатся ли в текстовом файле названия рек Днепр и Лена
5	Формирования массива строк, полученных в результате разделения текста в файле на фрагменты, разделенные символом «;»
6	Поиска слова «тема» и его однокоренных слов в текстовом файле

Окончание таблицы 16.1

7	Поиска слова «значение» и его однокоренных слов в текстовом файле и выделить их в консоли красным цветом
8	Замены слова «февраль», содержащегося в текстовом файле, на «март»
9	Замены слова «часы», содержащегося ли в текстовом файле, на «таймер»
10	Замены слова «зима», содержащегося ли в текстовом файле, на «весна»
11	Проверки, содержатся ли в текстовом файле названия государств Китай и Франция
12	Проверки, содержатся ли в текстовом файле названия городов Минск и Якутск

Контрольные вопросы

1 Что понимается под группировками регулярных выражений?
2 Какие модификаторы регулярных выражений Вы знаете?
3 Как построить шаблон для поиска вещественного числа со знаком?
4 Какой модификатор позволяет отменяет регистрозависимый поиск?
5 Для чего используются модификаторы U и m в регулярных выражениях?
6 Что такое обратные ссылки в регулярных выражениях?
7 Для чего используется модификатор x в регулярных выражениях?
8 Назовите операции, выполняемые с использованием группировок.
9 Какие методы класса Regex Вы знаете?
10 Какие операции можно выполнять с помощью метода IsMatch?

17 Лабораторная работа № 17. Программирование регулярных выражений

Регулярные выражения наиболее эффективны в программах, написанных на интерпретируемых (скриптовых) языках программирования Perl, JavaScript, PHP, VBScript и др.

В языке C# также имеются средства для работы со строками текста, например, в пространстве имен System.Text имеется класс String, который предоставляют достаточную функциональность. Однако в этом же языке содержится еще один мощный инструмент для работы с текстами – регулярные выражения (Regular Expression), представляющие собой основанный на использовании метасимволов формальный язык по обработке больших текстов, т. е. по выполнению манипуляций с его подстроками. Основная функциональность регулярных выражений в этом языке сосредоточена в пространстве имен System.Text.RegularExpressions, а центральным классом для работы с ними является класс Regex, в котором содержатся следующие методы:

IsMatch() – проверяет, содержит ли строка хотя бы одну подстроку, соответствующую шаблону регулярного выражения;

Match() – возвращает первую подстроку, соответствующую шаблону, в виде объекта типа Match, который содержит различную информацию о подстроке текста – длину, индекс, само значение и так далее;

Matches() – возвращает все подстроки, соответствующие шаблону в виде коллекции типа MatchCollection. Каждый элемент этой коллекции имеет тип Match;

Replace() – возвращает строку, в которой все подстроки, соответствующие шаблону, заменены новой строкой;

Split() – возвращает массив строк, полученный в результате разделения входящей строки в местах соответствия шаблону регулярного выражения.

Для поиска подстрок текста используется строка-образец (pattern – шаблон, маска), состоящая из символов и метасимволов и задающая правило поиска.

Для выполнения манипуляций со строками текста используются движки регулярных выражений, представляющие собой методы, выполняющие операции с текстом в соответствии с задаваемыми шаблонами. В языке C# эти шаблоны заключаются в двойные кавычки. При записи многострочных шаблонов и текста в двойных кавычках их следует предварять литералом @.

Рассмотрим пример использования регулярного выражения для нахождения телефонного номера в формате 111-111-1111:

```
String s = "456-435-2318";
Regex regex = new Regex (@"\d{3}-\d{3}-\d{4}");
```

В данном примере предполагается, что известно, сколько определенных символов должно быть в каждой группе цифр, поэтому явным образом в фигурных скобках указано их количество, например,\d{3} – три цифры.

Порядок выполнения работы

Разработать приложение в среде C# для указанного в таблице 17.1 варианта.

Таблица 17.1 – Варианты заданий

Вариант	Создать текстовый файл var.txt и	
	определить	выполнить
1	2	3
1	Содержится ли в нем заданное слово и сколько раз	Замену цифр словами: "0" – на "ноль", "1" – на "один" и так далее
2	Содержатся ли в нем теги <html>, <form> и <h1>	Вывод на экран всех операторов языка C#
3	Является ли текст английским	Определение номера телефона с самым продолжительным разговором
4	Количество строк комментариев в C# программе	Поиск цитат, т. е. предложений, заключенных в кавычки
5	Количество предложений в тексте, в которых интервал между словами больше одного	Поиск предложения, содержащего максимальное количество повторяющихся слов

Окончание таблицы 17.1

1	2	3
6	Сколько предложений начинается со слова Интернет	Поиск всех операторов языка C# ввода информации с клавиатуры
7	Сумму положительных чисел	Поиск предложения, содержащего максимальное число в файле
8	Строку, содержащую максимальное количество знаков пунктуации	Определение всех телефонных номеров, время связи с которыми было менее 1 мин
9	Сколько операторов for содержится в файле программы	Поиск предложений, содержащих только слова
10	Количество согласных букв	Выборку всех семизначных номеров телефона
11	Является ли текст русским	Замену цифр "5" – на "пять" и т. д.
12	Сумму положительных чисел	Определения номера телефона с наименьшей суммой его цифр

Контрольные вопросы

1 Что понимается под регулярным выражением?
2 Для решения каких задач предназначены регулярные выражения?
3 В каком пространстве имен сосредоточена основная функциональность регулярных выражений в языке C#?
4 Какие методы класса Regex Вы знаете?
5 Прокомментируйте назначение метода Match() класса Regex.
6 Что такое RegexOptions?
7 Какие значения может принимать перечисление RegexOptions?
8 Прокомментируйте символику языка регулярных выражений.
9 Что такое повторители в регулярных выражениях?

18 Лабораторная работа № 18. Обработка текстов с использованием регулярных выражений

Для обработки текстов с использованием регулярных выражений можно использовать методы Match(), Matches(), Replace(), Split() класса Regex.

Класс Regex имеет ряд конструкторов, позволяющих выполнить начальную инициализацию объекта. Эти конструкторы в качестве одного из параметров принимают перечисление RegexOptions, которое может принимать значения:

CultureInvariant – предписывает игнорирование региональных установок строки;

ExplicitCapture – модифицирует способ поиска, обеспечивая только буквенное соответствие;

IgnoreCase – игнорирует регистр символов во входной строке;

IgnorePatternWhitespace – производит удаление из строки пробелов и разрешает комментарии, начинающиеся со знака #;

Multiline – изменяет значения символов "^" и "$" так, что они применяются к началу и концу каждой строки, а не всего входного текста;

RightToLeft – предписывает читать строку справа налево (по умолчанию слева направо);

Singleline – специфицирует однострочный режим, в котором весь текст рассматривается как одна строка. Точка "." символизирует соответствие любому символу.

Например, в строке кода

```
Regexregex = newRegex(@"тур(\w*)"), RegexOptions.IgnoreCase);
```

создается регулярное выражение "тур(\w*)", которое может использоваться для выполнения регистронезависимого поиска подстроки "тур" в тексте. При необходимости можно установить несколько параметров, например,

```
Regex regex = newRegex(@"тур(\w*)", RegexOptions.Compiled | RegexOptions.IgnoreCase);
```

Порядок выполнения работы

Составить приложение на языке C# для работы со строками по одному из вариантов (таблица 18.1).

Таблица 18.1 – Варианты заданий

Вариант	Действие программы
1	2
1	Вывести на экран две исходные строки по три слова, создать на их основе новую строку из четырех слов и расположить их в алфавитном порядке
2	Извлечь из исходной строки три подстроки по три слова, сравнить их по количеству слов в каждой и расположить в порядке возрастания
3	Сформировать массив из шести слов исходной строки, затем объединить в новые строки каждую пару четных и нечетных слов и сравнить их на наличие одинаковых элементов
4	Создать исходную строку из массива, содержащуго восемь слов, объединить в новые строки каждую пару слов, расположив их в алфавитном порядке, и вывести на экран
5	С помощью регулярных выражений выполнить проверку на наличие в исходной строке из семи слов заданного слова, вывести на экран количество найденных совпадений слов и заменить найденные слова на произвольные другие
6	Создать символьную строку путем объединения двух исходных строк, выполнить поиск в ней первого и последнего слов, объединить найденные слова в новую строку и вывести ее на экран
7	В двух заданных исходных строках выполнить проверку первых и последних слов на совпадение. Из совпавших слов построить новую строку так, чтобы совпавшие слова чередовались два раза
8	Сформировать исходную строку из шести символов. Создать вторую строку из элементов символьного массива и поместить ее посередине исходной строки

Окончание таблицы 18.1

1	2
9	Вывести на экран две исходные строки по пять слов, создать на их основе новую строку из трех слов и расположить их в алфавитном порядке
10	Разобрать исходную строку на слова и записать их в массив строк, затем объединить в новые строки каждую пару нечётных слов
11	Создать строку слов путём объединения трех исходных строк, выполнить поиск первого и второго слов созданной строки, объединить найденные слова в новую строку и вывести ее на экран
12	Выполнить поиск в двух текстовых строках совпадающих слов. Из совпавших слов построить новую строку так, чтобы в ней каждое слово повторялось 2 раза

Контрольные вопросы

1 Для чего используются метасимволы «[ ]»,«( )», «{ }»?
2 Что означают метасимволы "^" и "$" в регулярном выражении?
3 Прокомментируйте назначение метасимволов «|»,«*»,«+».
4 Что означают шаблоны "a.p", "rs*", "a(cd)*k" ?
5 Что означают шаблоны "a(bc)+d", "ca(bine)?t", "boo(k|st|t)"?
6 Что означают шаблоны "\d+", "–?\d+\.\d+"?
7 Что такое символьный класс и как он записывается?
8 Что означает класс [abcd]?
9 Что понимается под регистронезависимым поиском?
10 Чему соответствует шаблон "б[аи]ржа"?

Список литературы

1 **Лондо, С. К.** Введение в дискретную математику / С. К. Лондо. – Москва : МНЦМО, 2014. – 264 с. : ил.

2 Теория формальных языков, грамматик и автоматов : методические указания к лабораторному практикуму / Сост. Е. Н. Ишакова. – Оренбург : ОГУ, 2005. – 54 с.

3 **Фридл, Дж.** Регулярные выражения : пер. с англ. / Дж. Фридл. – 4-е изд. – Санкт-Петербург : Символ, 2014. – 608 с. : ил.