МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное   
учреждение высшего образования

**"Южно-Уральский государственный университет**

**(национальный исследовательский университет)"**

**Высшая школа электроники и компьютерных наук**

**Кафедра системного программирования**

**ОТЧЕТ   
по учебной практике**

бакалавра направления 09.03.04 "Программная инженерия"

|  |  |
| --- | --- |
|  | Выполнил: студент группы КЭ-104  А.А Минниханов  Проверил:  Лаборант кафедры СП,  Н.С. Быков  Дата: \_\_\_\_\_\_\_\_, Оценка: \_\_\_\_\_\_\_\_\_\_ |

Челябинск-2022

Министерство науки и высшего образования Российской Федерации

Южно-Уральский государственный университет

Кафедра системного программирования

УТВЕРЖДАЮ

Зав. кафедрой

системного программирования

\_\_\_\_\_\_\_\_\_\_\_\_\_ Л.Б. Соколинский

**ЗАДАНИЕ**

**по** **учебной практике**

1. **Цель работы**

Необходимо разработать распознаватель заданной символьной цепочки. Символьная цепочка задается с помощью формул Бэкуса-Наура.

1. **Исходные данные к работе**
2. *Йенсен К., Вирт Н.* Паскаль. Руководство пользователя и описание языка. М.: Компьютер, 1995.
3. *Льюис Ф., Розенкранц Д., Стирнз Р.* Теоретические основы проектирования компиляторов. М.: Мир, 1979.
4. **Перечень подлежащих разработке вопросов**
5. Выполнить анализ требований и разработать спецификации программы.
6. Провести проектирование программы.
7. Реализовать спроектированные модули.
8. Провести тестирование и отладку реализованных модулей.
9. **Сроки**

Дата выдачи задания: "27" июня 2022 г.

Срок сдачи законченной работы: "24" июля 2022 г.

**Руководитель:**

Лаборант кафедры СП,

\_\_\_\_\_\_\_\_\_\_\_\_\_ Н.С. Быков

подпись

**Задание принял к исполнению:**

\_\_\_\_\_\_\_\_\_\_\_\_\_ А.А. Минниханов

подпись

**ОГЛАВЛЕНИЕ**

[1. Спецификация 4](#_Toc108651477)

[2. Проектирование 6](#_Toc108651478)

[2.1. Модульная структура 6](#_Toc108651479)

[2.2. Интерфейсы модулей 7](#_Toc108651480)

[3. Кодирование 11](#_Toc108651481)

[3.1. Структура текста программы 11](#_Toc108651482)

[3.2. Алгоритмы реализации модулей 12](#_Toc108651483)

[3.2.1. Блок транслитерации 12](#_Toc108651484)

[3.2.2. Лексический блок 13](#_Toc108651485)

[3.2.3. Синтаксический блок 15](#_Toc108651486)

[3.2.4. Блок идентификации ключевых слов 16](#_Toc108651487)

[3.3. Размер текста программы (в строках) 16](#_Toc108651488)

[4. Тестирование 17](#_Toc108651489)

[4.1. Автономное тестирование 17](#_Toc108651490)

[4.2. Комплексное тестирование 18](#_Toc108651491)

[Заключение 20](#_Toc108651492)

[Литература 21](#_Toc108651493)

# Спецификация

Необходимо разработать распознаватель заданной символьной цепочки. Символьная цепочка задается с помощью формул Бэкуса-Наура. (**Вариант S**)

<цепочка>::=<оператор while-do>

<оператор while-do>::=WHILE <условие> DO <оператор>;

<условие>::=<логическая константа>

<логическая константа>::=TRUE | FALSE

<оператор>::=<оператор if-then>

<оператор if-then>::=IF <условие> THEN <оператор1>;

<условие>::=<идентификатор><операция сравнения><16-ричная константа>

<идентификатор>::= <буква> | <идентификатор><буква> | <идентификатор><цифра>

<операция сравнения>::= > | < | >= | <= | <>

<цифра>::=0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<буква>::=A | B | C | D | E | F | ... | Z

<16-ричная константа>::= $<список 16-ричных букв и цифр>

<список 16-ричных букв и цифр>::=<цифра> | <16-ричная буква><список 16-ричных букв и цифр>

<16-ричная буква>::=A | B | C | D | E | F

<оператор1>::=<вызов подпрогрммы>

<список параметров>::=<список идентификаторов>

<список идентификаторов>::=<идентификатор> | <идентификатор>,<список идентификаторов>

<вызов подпрограммы>::=<идентификатор> (<список параметров>)

Помимо этого, на цепочку накладывается следующее семантическое ограничение: идентификатор, входящий в цепочку, не должен совпадать с ключевыми словами языка Pascal [1].

**Описание входных данных**

Цепочка записана в текстовом файле INPUT.TXT, который состоит из одной строки. Длина цепочки не превышает 80 символов.

**Описание выходных данных**

Результат распознавания необходимо записать в текстовый файл OUTPUT.TXT в одно из следующих сообщений: ACCEPT, если цепочка допустима, и REJECT, если цепочка недопустима.

**Таблица 1.** Примеры входных и выходных данных

|  |  |
| --- | --- |
| **INPUT.TXT** | **OUTPUT.TXT** |
| while false do if A>$a then p(i); | ACCEPT |
| while true do if MyVar<>$FFFF then Snooze(param); | ACCEPT |
| while True do if Check<>$1F then Kill(p1z, p2, p3, p4); | ACCEPT |
| while True do if MyWorld <>$FF then Space(another); | ACCEPT |
| while false do if car > $a1 then petrol(m); | REJECT |
| while do if B > $ then l; | REJECT |
| while true if baby < $BB then f; | REJECT |
| while true do if n>$AAA then; p(a4) | REJECT |
| while if do if n>$AAA then p(a4; | REJECT |
| while false do: if B>$b then hh2(2i); | REJECT |

# Проектирование

В данном параграфе описана модульная структура программы и интерфейсы модулей. Также описана семантика модулей программы.

## Модульная структура

**Рис 1.** Модульная структура

Модульная структура представляет собой иерархию процедур и функций (называемых модулями), с помощью которых программа решает поставленную задачу. При этом программа является головным модулем в данной иерархии.

**Ввод данных** – считывает данные из файла INPUT.txt для дальнейшей работы с ними.

**Вывод данных** – записывает итоговый результат в файл OUTPUT.txt.

**Программа** – главный модуль, из которого происходит запуск программы и обработка входной цепочки.

**Блок транслитерации** – подпрограмма, преобразующая исходную символьную цепочку в цепочку лексем вида ("символ цепочки", "класс символа цепочки ").

**Лексический блок** – подпрограмма, преобразующая цепочку лексем, полученную от транслитератора, в цепочку лексем вида ("символ входного языка", "класс символа входного языка").

**Блок идентификации** – подпрограмма, которая устанавливает, какое из ключевых слов языка Pascal соответствует заданному идентификатору, либо сообщает, что заданный идентификатор не является ключевым словом языка Pascal.

**Синтаксический блок** – подпрограмма, которая получает цепочку лексем вида ("символ входного языка", "класс символа входного языка") и устанавливает, соответствует ли она заданным формулам Бэкуса-Наура.

## Интерфейсы модулей

Программа:

Подпрограмма main

Модуль вызывает остальные подпрограммы.

Ввод данных:

Входные данные: - исходная символьная цепочка

Выходные данные: string – строка, считанная из файла input.txt.

Вывод данных:

Входные данные: result – переменная, в зависимости от которой в файл output.txt выводится ACCEPT или REJECT.

Выходные данные: –

В зависимости от значения переменной result в файл output.txt

записывается результат.

Блок транслитерации:

Семантика: def block1(string):

Входные данные: string – строка, считанная из input.txt. Если будет встречен неописанный в блоке символ, будет выведено сообщение: "ошибка, недопустимый символ".

Выходные данные: возвращает двумерный массив размера n \* 2 вида ["символ цепочки"] ["класс символа цепочки"], где n – длина исходной строки.

Например, символьную цепочку “while false do if A>$a then p(i);” блок транслитерации должен преобразовать в двумерный массив лексем, пример представлен в таблице 2.

**Таблица 2**. Пример массива блока транслитерации

|  |  |
| --- | --- |
| w | буква |
| h | буква |
| i | буква |
| l | буква |
| e | буква16 |
|  | пробел |
| f | буква |
| a | буква16 |
| l | буква |
| s | буква |
| e | буква16 |
|  | пробел |
| d | буква16 |
| o | буква |
|  | пробел |
| i | буква |
| f | буква16 |
|  | пробел |
| A | буква16 |
| > | сравнение |
| $ | доллар |
| s | буква |
|  | пробел |
| t | буква |
| h | буква |
| e | буква16 |
| n | буква |
|  | пробел |
| p | буква |
| ( | скобка1 |
| i | буква |
| ) | скобка2 |
| ; | тчкзпт |

Лексический блок:

Семантика: def block2(string):

Входные данные: string –массив кортежй, в котором находится строка, разбитая на символы с присвоенными им классами.

Блок отвечает за преобразование двумерного массива лексем, полученного от блока транслитерации, в двумерный массив лексем вида ["слово входного языка"] ["класс слова входного языка"]. В рассматриваемом примере лексический блок должен выдать цепочку лексем, описанную в таблице 3.

**Таблица 3.** Пример массива лексического блока

|  |  |
| --- | --- |
| while | идентиф |
| false | идентиф |
| do | идентиф |
| if | идентиф |
| A | идентиф |
| > | сравнение |
| $abc | конст |
| then | идентиф |
| p | идентитф |
| ( | скобка1 |
| i | идентиф |
| ) | скобка2 |
| ; | тчкзпт |

Блок идентификации ключевых слов:

Семантика: def block3(string):

Входные данные: string – двумерный массив, идентификаторы которого просматриваются на совпадения с ключевыми словами языка Паскаль.

Выходные данные: arr – двумерный массив, идентификаторы, классы которых были изменены в случае совпадения с ключевыми словами языка Pascal. Блок устанавливает, присутствуют ли в двумерном массиве лексем ключевые слова языка Паскаль.

Синтаксический блок:

Семантика: def block4(string):

Входные данные: arr – двумерный массив, содержащий лексемы и их классы.

Выходные данные: если данная цепочка не совпадает с приведенными в подпрограмме примерами будет возвращено REJECT, иначе – ACCEPT.

Блок устанавливает, соответствует ли двумерный массив заданным формулам Бэкуса-Наура. Данный блок использует для работы только классы символов входного языка.

# Кодирование

На данном этапе описан процесс кодирования: структура текста программы, алгоритмы реализации модулей, размер текста программы. Алгоритмы реализации модулей поделены на три блока: блок транслитерации, лексический блок и синтаксический блок. Программа написана на языке Python в программе PyCharm.

## Структура текста программы

**Рис 2.** Структура программы

main.py – головной модуль, содержащий функцию, по очереди вызывающую функции из каждого модуля.

transliteration.py – модуль транслитерации, содержащий функцию, получающую на вход символьную цепочку data, считанную из файла input.txt.

lexical.py – лексический модуль, содержащий функцию, получающую на вход список классов «символ/тип символа», состоящий из исходной строки, разбитой на символы и возвращающий список классов «символ/тип символа», состоящий из слов входного языка и их типов.

identifical.py – модуль идентификации ключевых слов. Функция получает на вход список лексем, состоящих из слов входного языка и их типов, проверяет, есть ли среди слов входного языка ключевые слова языка Pascal и, в случае их наличия, меняет класс этих слов на само слово.

syntax.py – синтаксический модуль, содержащий функцию, получающую на вход список лексем, состоящих из слов входного языка и их типов, проверенный на наличие ключевых слов, и возвращающий значение 0 или 1 в зависимости от корректности цепочки.

## Алгоритмы реализации модулей

В данном подразделе описаны детали реализации главных модулей программы.

Программа представляет собой автоматы с явным выделением состояний, ее компоненты подразделяются на события и входные переменные.

Совокупность конечного множества состояний и конечного множества входных воздействий образует автомат. Такой автомат реагирует на входные воздействия, определенным образом изменяя текущее состояние. Смена состояний, происходит с помощью переходов автомата. Состояния хранятся в переменной state.

### Блок транслитерации

Все символы входной строки обрабатываются и разбиваются по классам, представленным в таблице 4.

**Таблица 4**. Таблица транслитерации

|  |  |
| --- | --- |
| A…F | буква16 |
| G…Z | буква |
| $ | доллар |
| 0..9 | цифра |
|  | пробел |
| , | запятая |
| >,<,= | сравнение |
| ; | тчкзпт |
| ( | скобка1 |
| ) | скобка2 |
| <Другие> | ошибка |

Если хотя бы одному символу класса error при присвоении будет выведена ошибка, то выполнение программы прекратится, т.е. исходная цепочка будет отвергнута.

### Лексический блок

В Таблице 5 определены состояния конечного распознавателя лексического блока символьной цепочка.

**Таблица 5**. Состояния конечного распознавателя лексического блока

| **№** | **Состояние** | **Семантика** |
| --- | --- | --- |
| 1 | пробел | Чтение пробелов между лексемами. Начальное состояние |
| 2 | идентиф | Чтение идентификатора |
| 3 | 16-ричное целое | Чтение 16-ричного числа |
| 4 | сравн | Чтение оператора сравнения |
| 5 | ошибка | Ошибка |

В таблице 6 представлен конечный распознаватель лексического блока.

**Таблица 6**. Конечного распознавателя лексического блока

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | буква | буква16 | цифра | пробел | сравнение | $ | ( | ) | , | ; |
| пробел | идент | идент | Е | пробел | сравн | конст | Е | пробел | пробел | пробел | 1 |
| идент | идент | идент | идент | пробел | сравн | Е | пробел | пробел | пробел | Е | 0 |
| сравн | Е | Е | Е | пробел | сравн | конст | Е | Е | Е | Е | 0 |
| конст | Е | конст | конст | пробел | Е | Е | Е | Е | Е | Е | 0 |
| E | Е | Е | Е | Е | E | Е | Е | Е | Е | Е | 0 |

В полученном конечном распознавателе нет недостижимых состояний. Дальнейший процесс редукции конечного распознавателя лексического блока приведен в Таблице 7.

**Таблица 7**. Редукция конечного распознавателя лексического блока

| **Шаг** | **Результат (блоки состояний)** | **Действия** |
| --- | --- | --- |
| 1 | P0 = {пробел, идент, сравн, конст, E} | Разбиваем Р0 на два блока: допустимые и отвергающие состояния |
| 2 | P11 = {пробел}  P12= {идент, сравн, конст, Е} | Разбиваем P12 по входу *пробел*. |
| 3 | P21= {пробел}  P22= {идент, сравн, конст}  P23 = {E} | Разбиваем P22 по входу *сравнение*. |
| 4 | P31= {пробел}  P32= {идент, сравн}  P33 = {E}  P34 = {конст} | Разбиваем P32 по входу *буква*. |
| 5 | P41= {пробел}  P42= {идент, сравн}  P43 = {сравн}  P44 = {E}  P45 = {конст} |  |

Проведенные преобразования доказывают, что конечный распознаватель, полученный после замены эквивалентных состояний, является минимальным.

**Таблица 8** – Примитивные процедуры обрабатывающего автомата лексического блока

|  |  |  |
| --- | --- | --- |
| **№ п/п** | **Процедура** | **семантика** |
| 1 | Да | Остановить обработку и допустить цепочку |
| 2 | Нет | Остановить обработку и отвергнуть цепочку |
| 3 | Обработать | Добавить входной символ к значению текущей лексемы |
| 4 | Лексема (класс) | Увеличить счетчик лексем на 1, установить заданный класс текущей лексемы. |

**Таблица 9** – Процедуры переходов обрабатывающего автомата лексического блока

| **Действие** | **Семантика** |
| --- | --- |
| ➊ | Обработать; |
| ➋ | Лексема(идентиф);  Обработать; |
| ➌ | Лексема (конст)  Обработать; |
| ➍ | Лексема(сравн);  Обработать; |
| ➎ | Лексема(лев\_скобка);  Обработать; |
| ➏ | Лексема(прав\_скобка);  Обработать; |
| ❼ | Лексема(запятая);  Обработать; |
| ❽ | Лексема(точксзп);  Обработать; |

**Таблица 10**. Обрабатывающий автомат лексического блока

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | буква | буква16 | цифра | пробел | сравнение | $ | ( | ) | , | ; |
| пробел | Идент  ➋ | Идент  ➋ | Е | пробел | Сравн  ➍ | Конст  ➌ | Е | Пробел  ➏ | Пробел  ❼ | Пробел  ❽ | 1 |
| идент | Идент  ➊ | Идент  ➊ | Идент  ➊ | пробел | Сравн  ➍ | Е | Пробел  ➎ | пробел | пробел | Е | 0 |
| сравн | Е | Е | Е | пробел | Сравн  ➊ | Конст  ➌ | Е | Е | Е | Е | 0 |
| конст | Е | Конст  ➊ | Конст  ➊ | пробел | Е | Е | Е | Е | Е | Е | 0 |
| E | Е | Е | Е | Е | E | Е | Е | Е | Е | Е | 0 |

### Синтаксический блок

Составляется конечный распознаватель на основе исходных формул Бэкуса-Наура и лексем, полученных в Лексическом блоке и в Блоке идентификации ключевых слов.

Конечный распознаватель синтаксического блока рассматриваемого нами примера приведен в таблице 11.

**Таблица 11.** Конечный распознаватель синтаксического блока

|  | **KW\_WHILE** | **KW\_TRUE, KW\_FALSE** | **KW\_DO** | **KW\_IF** | **IDENT** | **COMPAR** | **CONST** | **KW\_THEN** | **L\_BR** | **COMMA** | **R\_BR** | **SEMICOL** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| WHILE | BOOL |  |  |  |  |  |  |  |  |  |  |  |
| BOOL |  | DO |  |  |  |  |  |  |  |  |  |  |
| DO |  |  | IF |  |  |  |  |  |  |  |  |  |
| IF |  |  |  | IDENT1 |  |  |  |  |  |  |  |  |
| IDENT1 |  |  |  |  | COMPAR |  |  |  |  |  |  |  |
| COMPAR |  |  |  |  |  | HEX\_CONST |  |  |  |  |  |  |
| HEX\_CONST |  |  |  |  |  |  | THEN |  |  |  |  |  |
| THEN |  |  |  |  |  |  |  | FUNC |  |  |  |  |
| FUNC |  |  |  |  | OPEN |  |  |  |  |  |  |  |
| OPEN |  |  |  |  |  |  |  |  | ARG |  |  |  |
| ARG |  |  |  |  | CLOSE |  |  |  |  |  |  |  |
| CLOSE |  |  |  |  |  |  |  |  |  | ARG | SEMICOL |  |
| SEMICOL |  |  |  |  |  |  |  |  |  |  |  | END |
| END |  |  |  |  |  |  |  |  |  |  |  |  |

Описанный выше автомат является минимальным. Пустым клеткам соответствует вызов примитивной процедуры НЕТ, которая отвергает цепочку.

Все пустые клетки обозначают вызов примитивной процедуры НЕТ.

### Блок идентификации ключевых слов

Ключевые слова языка Pascal хранятся в массиве строк в алфавитном порядке. Для идентификации ключевых слов проводится линейный поиск по массиву. Если идентификатор является ключевым словом, его класс меняется на это ключевое слово. Если в цепочке встречается более одного ключевого слова или любое ключевое слово, не являющееся обязательным для данной цепочки, то такая цепочка отвергается.

## Размер текста программы (в строках)

Размер программы составляет суммарно 306 строк:

В модуле transliteration содержится 31 строка,

В модуле lexical содержится 146 строк,

В модуле identifical содержится 22 строки,

В модуле syntax содержится 99 строк,

В модуле main содержится 8 строк.

# Тестирование

## Автономное тестирование

Далее описано тестирование автономное тестирование созданных модулей. В таблице приведены протоколы тестирования блока транслитерации.

**Таблица 12**. Тестирование блоков

| **Модуль** | **Тесты** | |
| --- | --- | --- |
| **Входные данные** | **Выходные данные** |
| Блок транслитерации | while false do if A>$a then p(i); | ('w', 'буква'), ('h', 'буква'), ('i', 'буква'), ('l', 'буква'), ('e', 'буква'), (' ', 'пробел'), ('f', 'буква'), ('a', 'буква'), ('l', 'буква'), ('s', 'буква'), ('e', 'буква'), (' ', 'пробел'), ('d', 'буква'), ('o', 'буква'), (' ', 'пробел'), ('i', 'буква'), ('f', 'буква'), (' ', 'пробел'), ('A', 'буква'), ('>', 'больше'), ('$', 'доллар'), ('a', 'буква'), (' ', 'пробел'), ('t', 'буква'), ('h', 'буква'), ('e', 'буква'), ('n', 'буква'), (' ', 'пробел'), ('p', 'буква'), ('(', 'скобка1'), ('i', 'буква'), (')', 'скобка2'), (';', 'тчкзпт'), (' ', 'пробел') |
| Лексический блок | 'w', 'буква'), ('h', 'буква'), ('i', 'буква'), ('l', 'буква'), ('e', 'буква'),  (' ', 'пробел'), ('f', 'буква'), ('a', 'буква'), ('l', 'буква'), ('s', 'буква'), ('e', 'буква'),  (' ', 'пробел'), ('d', 'буква'),  ('o', 'буква'),  (' ', 'пробел'), ('i', 'буква'), ('f', 'буква'),  (' ', 'пробел'), ('A', 'буква'), ('>', 'больше'), ('$', 'доллар'), ('a', 'буква'),  (' ', 'пробел'), ('t', 'буква'), ('h', 'буква'), ('e', 'буква'), ('n', 'буква'),  (' ', 'пробел'), ('p', 'буква'), ('(', 'скобка1'), ('i', 'буква'), (')', 'скобка2'), (';', 'тчкзпт'),  (' ', 'пробел') | ('while', 'ИДЕНТИФИКАТОР'),  ('false', 'ИДЕНТИФИКАТОР'), ('do', 'ИДЕНТИФИКАТОР'),  ('if', 'ИДЕНТИФИКАТОР'),  ('A', 'ИДЕНТИФИКАТОР'),  ('>', 'больше'),  ('$', 'доллар'),  ('a', 'ЦЕЛОЕ'),  ('then', 'ИДЕНТИФИКАТОР'),  ('p', 'ИДЕНТИФИКАТОР'),  ('(', 'скобка1'),  ('i', 'ИДЕНТИФИКАТОР'),  (')', 'скобка2'),  (';', 'тчкзпт') |
| Блок идентификации ключевых слов | 'while' | 'КЛСЛОВО\_while' |
| 'false' | 'ЛОГКОНСТ\_false' |
| 'do' | 'КЛСЛОВО\_do' |
| 'if' | 'КЛСЛОВО\_if' |
| 'A' | 'ИДЕНТИФИКАТОР' |
| '>' | 'больше' |
| '$' | 'доллар' |
| 'а' | 'ЦЕЛОЕ' |
| 'then' | 'КЛСЛОВО\_then' |
| 'p' | 'ИДЕНТИФИКАТОР' |
| '(' | 'скобка1' |
| Синтаксический блок | 'КЛСЛОВО\_while'  'ЛОГКОНСТ\_false'  'КЛСЛОВО\_do'  'КЛСЛОВО\_if'  'ИДЕНТИФИКАТОР'  'больше'  'доллар'  'ИДЕНТИФИКАТОР'  'КЛСЛОВО\_then'  'ИДЕНТИФИКАТОР'  'скобка1'  'ИДЕНТИФИКАТОР'  'скобка2'  'тчкзпт' | ACCEPT |
| 'КЛСЛОВО\_while'  'ЛОГКОНСТ\_false'  'КЛСЛОВО\_do'  'КЛСЛОВО\_if'  'ИДЕНТИФИКАТОР'  'больше'  'доллар'  'ИДЕНТИФИКАТОР'  'КЛСЛОВО\_then'  'ИДЕНТИФИКАТОР'  'скобка1'  'тчкзпт'  'ИДЕНТИФИКАТОР'  'скобка2' | REJECT |
| 'КЛСЛОВО\_WHILE', 'ЛОГКОНСТ\_FALSE', 'КЛСЛОВО\_DO', 'КЛСЛОВО\_IF', 'доллар', 'ЦЕЛОЕ', 'больше', 'ИДЕНТИФИКАТОР', 'КЛСЛОВО\_THEN', 'ИДЕНТИФИКАТОР', 'скобка1', 'ИДЕНТИФИКАТОР', 'скобка2', 'тчкзпт' | REJECT |

## Комплексное тестирование

**Таблица 13.** Пример набора тестов для модулей распознавателя цепочки

| **№**  **п/п** | **Входные данные** | **Выходные данные** | **Статус** |
| --- | --- | --- | --- |
| 1 | while false do if A>$a then p(i); | ACCEPT | Пройден |
| 2 | while true do if MyVar<>$FFFF then Snooze(param); | ACCEPT | Пройден |
| 3 | while True do if Check<>$1F then Kill(p1z, p2, p3, p4); | ACCEPT | Пройден |
| 4 | while True do if MyWorld <>$FF then Space(another); | ACCEPT | Пройден |
| 5 | while false do if car > $a1 then petrol(m); | REJECT | Пройден |
| 6 | while do if B > $ then l; | REJECT | Пройден |
| 7 | while true if baby < $BB then f; | REJECT | Пройден |
| 8 | while true do if n>$AAA then; p(a4) | REJECT | Пройден |
| 9 | while if do if n>$AAA then p(a4; | REJECT | Пройден |
| 10 | while false do: if B>$b then hh2(2i); | REJECT | Пройден |

# Заключение

Практика является обязательным разделом и представляет собой вид учебной деятельности, направленной на формирование, закрепление, развитие практических навыков и компетенций в процессе выполнения определенных видов работ, связанных с будущей профессиональной деятельностью.

За время прохождения учебной практики был получен практический опыт работы с распознаванием цепочки, заданной формулами Бэкуса-Наура, были практически закреплены навыки работы с автоматами, разработки алгоритмов и кодирования.

Были пройдены все этапы разработки программы: анализ, спецификация, проектирование, разработка алгоритмов, кодирование, тестирование и сопровождение.

Для каждого этапа, кроме сопровождения, посвящен отдельный пункт отчета, в котором подробно описаны проделанная работа и ее результаты.

Полученные в ходе учебной практики знания можно применить в будущем при создании синтаксических и лексических анализаторов, тестировании программного обеспечения на основе моделей, а также разработке искусственного интеллекта.

# Литература

1. Дональд Кнут . Искусство программирования, том 3. Сортировка и поиск = The Art of Computer Programming, vol.3. Sorting and Searching. — 2-е изд. — М.: «Вильямс», 2007.

2. Ван Тассел Д. Стиль, разработка, эффективность, отладка и испытание программ. - М.: Мир, 1985.

3. Йенсен К., Вирт Н. Паскаль. Руководство пользователя и описание языка. М.: Компьютер, 1995.

4. Льюис Ф., Розенкранц Д., Стирнз Р. Теоретические основы проектирования компиляторов. М.: Мир, 1979.

5. Силкина Н.С. Автоматы: [Электронный ресурс] URL: https://edu.susu.ru/pluginfile.php/8423876/mod\_resource/content/3/2-Automates.pdf (дата обращения: 14.07.2022 г.).