Министерство образования Республики Беларусь

Учреждение образования
“Белорусский государственный университет
информатики и радиоэлектроники”

Факультет информационных технологий и управления

Кафедра интеллектуальных информационных технологий

ЛАБОРАТОРНАЯ РАБОТА №1

по дисциплине «Логические основы интеллектуальных систем»

на тему

«Представление и синтаксическая проверка формул языка логики высказываний»

Вариант Е

Выполнили студентки гр. 121702 Мойсевич А. В.

Проверил Ивашенко В. П.

Минск 2023

Цель: приобрести навыки работы с синтаксическим разбором формул языка логики высказываний.

Задача: Проверить, является ли формула КНФ.

Описание лабораторной работы:

В ходе лабораторной работы необходимо проверить, является ли введенная формула КНФ. Для этого требуется:

1. Проверить все символы формулы на соответствие алфавиту языка логики высказываний.
2. Проверить формулу на соответствие правилам грамматики языка логики высказываний.
3. Проверить формулу на соответствие определению КНФ.

Для выполнения данной проверки были выделены следующие подзадачи:

1. Упрощение формулы до вида, когда она состоит лишь из бинарных связок и атомарных формул.
   1.1. Поиск в формуле переменных, названия которых удовлетворяют грамматике атомарной формулы в варианте, где есть не только символ, но и натуральное, с последующей заменой в исходной формуле <символ>[{<натуральное>}] на <символ>.
   1.2. Поиск подформул в формуле, являющихся дизъюнкцией двух атомарных формул или их отрицаний.
   1.3. Анализ синтаксической и грамматической структуры подформулы.
   1.4. Замена подформулы атомарной формулой.
2. Проверка упрощенной формулы на соответствие КНФ:
   2.1. Анализ синтаксической и грамматической структуры формулы
   2.2. Проверка соответствия грамматике КНФ

Дополнительные теоретические сведения:

Алфавит языка логики высказываний — алфавит, включающий символы логических констант и логических связок, символы для обозначения высказываний, скобки для указания приоритета операций (45 символов: 2 логических константы, десятичные цифры, заглавные буквы латинского алфавита для обозначения высказываний, 5 логических связок).

Алфавит – конечное или счетное множество символов.

Множество — абстрактная сущность, непосредственно связывающая одну или несколько сущностей в целое.

Абстрактный — существующий во внутренней памяти субъекта.

Субъект — носитель действия.

Действие — явление, которое имеет событие, предшествующее всем остальным событиям.

Целое — отнесенное к себе или к своим частям.

Отношение — множество связок.

Связка — абстрактная связь, множество не менее чем из одного элемента.

Формальный язык — множество текстов формального языка над некоторым алфавитом.

Грамматика формального языка состоит из правил вида п::=ф.

Грамматика языка логики высказываний:

<логическая константа> ::= T | .L

<ненулевая цифра> ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10

<цифра> ::= 0 | <ненулевая цифра>

<натуральное> ::= <ненулевая цифра>[{<цифра>}]

<латинская заглавная буква> ::= A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

<формула> ::= <логическая константа> | < латинская заглавная буква > | <унарная сложная формула> | <бинарная сложная формула>

<унарная сложная формула> ::= <открывающая скобка><отрицание><формула><закрывающая скобка>

<открывающая скобка> ::= (

<отрицание> ::= !

<закрывающая скобка> ::=)

<атомарная формула> ::= <символ>[{<натуральное>}]

<бинарная сложная формула> ::= <открывающая скобка><формула> <бинарная связка><формула><закрывающая скобка>

<бинарная связка> ::= <конъюнкция> | <дизъюнкция> | <импликация> | <эквиваленция>

<конъюнкция> ::= /\

<дизъюнкция> ::= \/

<импликация> :: = ->

<эквиваленция> ::= ~

Подформула языка логики высказываний — формула языка логики высказываний, которая является подстрокой формулы языка логики высказываний.

Конъюнктивная нормальная форма(КНФ)— нормальная форма, в которой логическая формула имеет вид конъюнкции нескольких простых дизъюнктов.

Простой дизъюнкцией или *дизъюнктом* называется дизъюнкция одной или нескольких переменных или их отрицаний.

Примеры формул в КНФ:

(A13 /\ B)

((A \/ B) /\ (!(A \/ C)))

((A \/ B) /\ (B \/ C) /\ ((A \/ C12) \/ B))

(((A4 \/ C) \/ D1) /\ (((A4 \/ B) \/ (B \/ C)) \/ D))

Примеры формул не в КНФ:

(A \/ B1)

(!((A12 /\ B) \/ C1))

((A -> B) /\ (C \/ D))

(((!C) \/ (A \/ B)) /\ ((((!B) \/ (!C)) \/ (!D)) \/ A) \/ (A \/ ((!C) \/ (!D))))

Описание программы и алгоритма:

Программа включает в себя метод поиска и замены атомарных формул, метод проверки отрицаний, метод поиска подформул, являющихся дизъюнкцией атомарных формул или их отрицаниями, метод проверки подформулы на соответствие алфавиту языка и грамматике дизъюнкции, метод проверки подформулы на соответствие алфавиту языка и грамматике отрицания, метод проверки формулы на соответствие грамматике КНФ.

1. Метод поиска и замены атомарных формул проверяет каждую атомарную формулу на соответствие грамматике атомарных подформул: <символ>[{<натуральное>}] –, если оно есть, то заменяет её на <символ>, тем самым упрощая запись формулы для последующего анализа.
2. Метод проверки отрицаний проверяет наличие отрицания неатомарной формулы. Если таковое найдено, метод возвращает False, что трактуется как то, что анализируемая формула не является КНФ.
3. Метод поиска подформул, являющихся дизъюнкцией атомарных формул или их отрицаниями, проверяет с помощью методов проверки найденные подформулы на соответствие грамматике бинарной или унарной сложной формулы: <открывающая скобка> <формула> <дизъюнкция> <формула> <закрывающая скобка> | <открывающая скобка><отрицание> <формула><закрывающая скобка> –, если оно есть, то подформула заменяется на <латинскую заглавную букву>, тем самым упрощая запись формулы для последующего анализа. При отсутствии соответствия грамматике или алфавиту, данный метод возвращает значение False, что трактуется как то, что анализируемая формула не является КНФ.
4. Метод проверки подформулы на соответствие алфавиту языка и грамматике дизъюнкции проверяет формулу на соответствие <формула><дизъюнкция><формула>, где <формула> – это <латинская заглавная буква>. Так как поиск подформул осуществляется от <открывающаяся скобка> до <закрывающаяся скобка>, то дополнительно сверять их при данном методе избыточно.
5. Метод проверки подформулы на соответствие алфавиту языка и грамматике отрицания проверяет формулу на соответствие <отрицание><формула>, где <формула> – это <латинская заглавная буква>. Так как поиск подформул осуществляется от <открывающаяся скобка> до <закрывающаяся скобка>, то дополнительно сверять их при данном методе избыточно.

6. Метод проверки формулы на соответствие грамматике КНФ проверяет, выполняется ли грамматика конъюнкции и соответствуют ли переменные алфавиту. Если всё соблюдено, метод возвращает True, что означает соответствие конъюнктивной нормальной

7. форме, в противном случае – несоответствие КНФ.

Блок-схемы разработанных алгоритмов:

start()

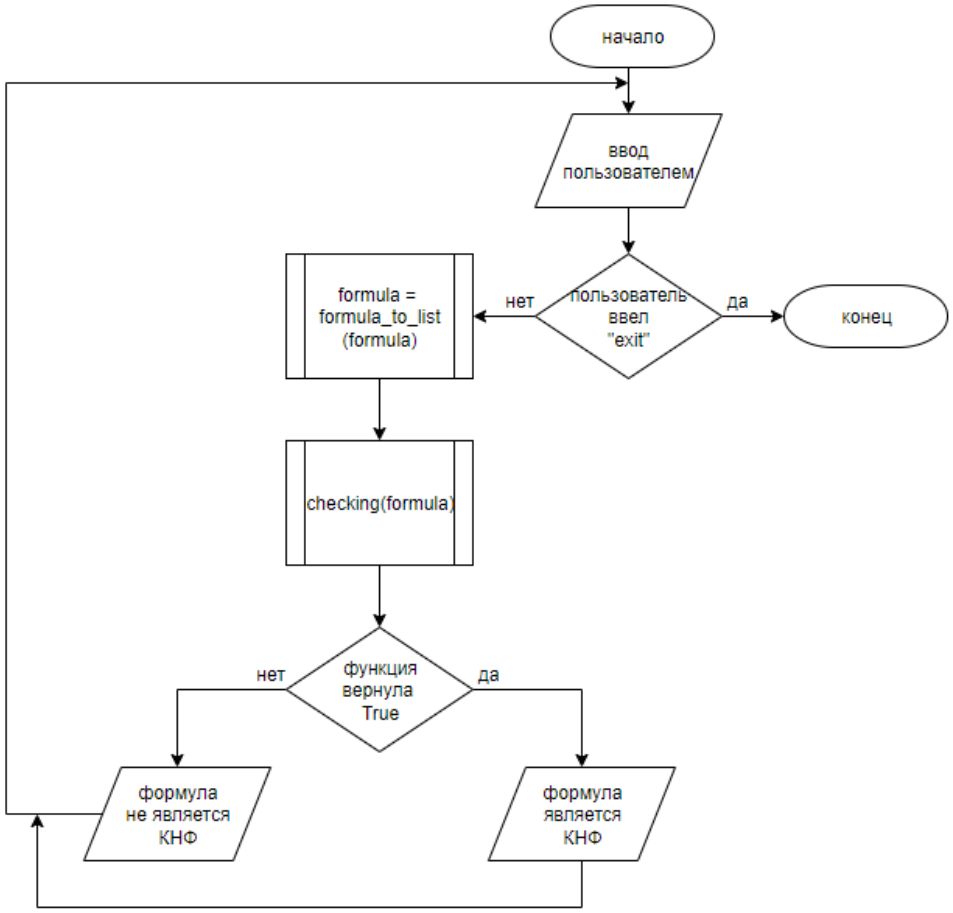

Рис. 1 – Блок-схема функции start()

formula_to_list(formula)

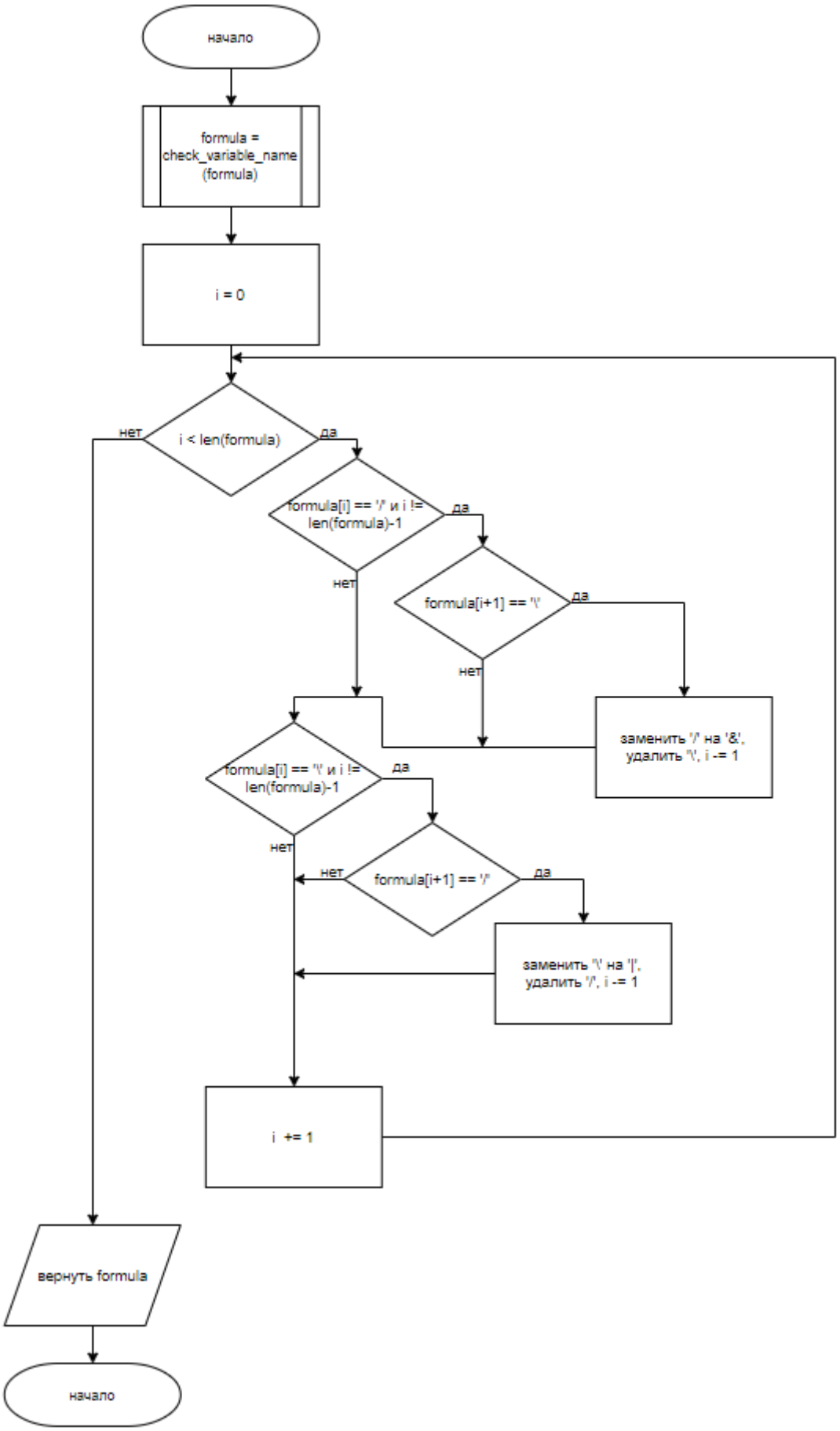

Рис. 2 – Блок-схема функции *formula to_list(formula)*

negation_check(expression)

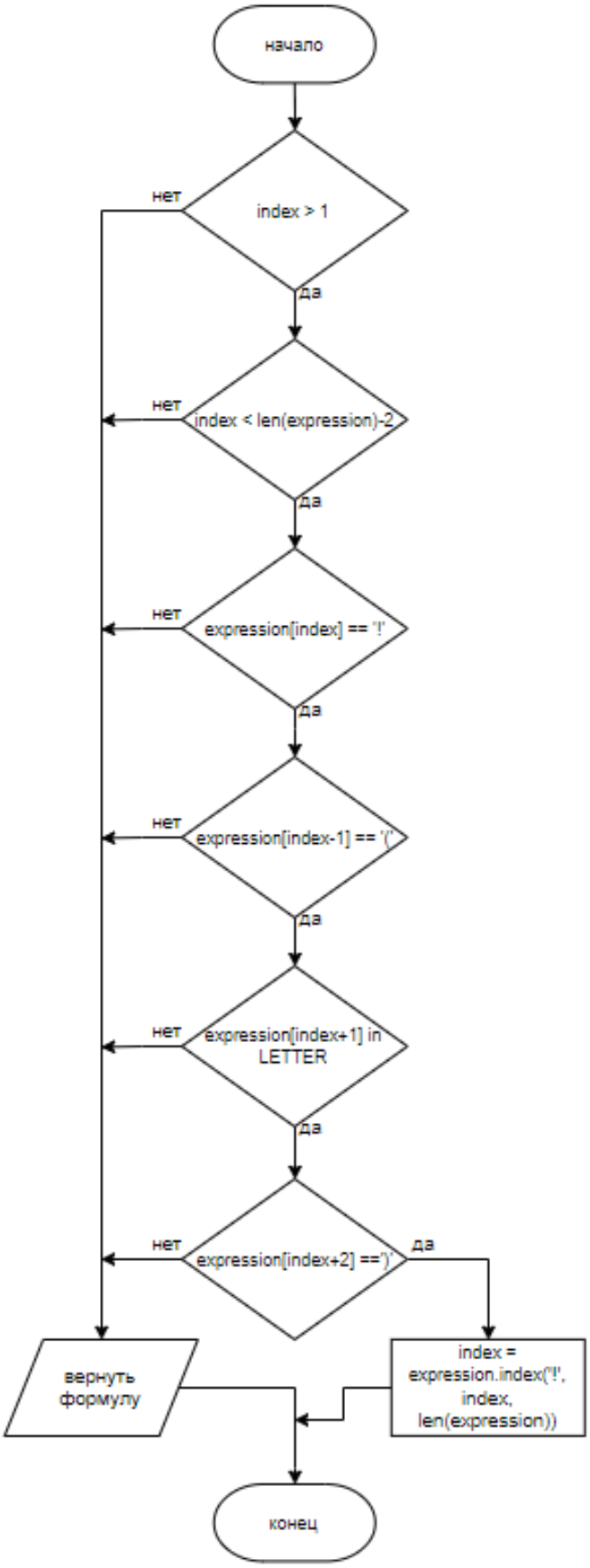

Рис. 3 – Блок-схема функции *negation check(expression)*

check_variable_name(formula)

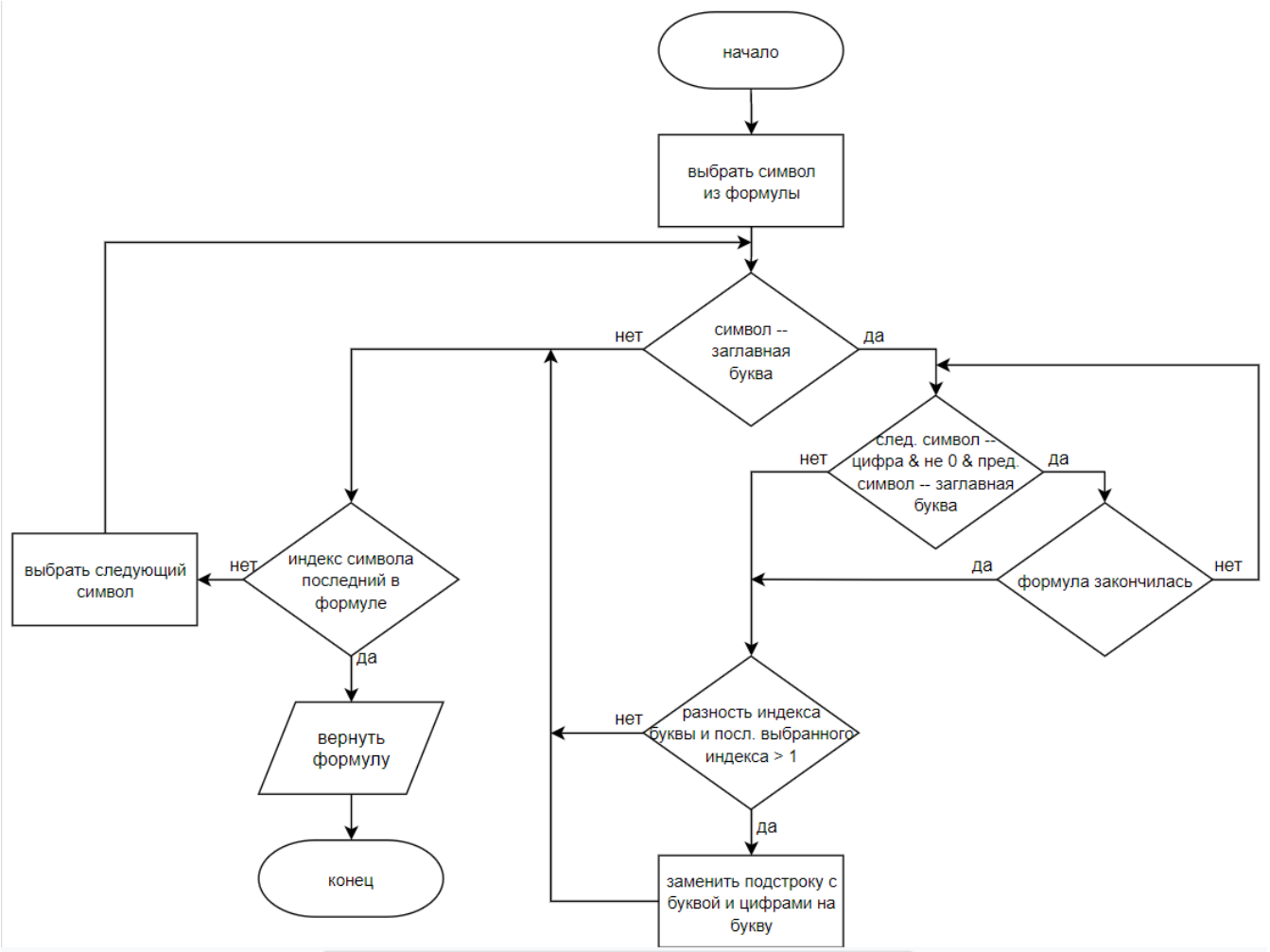

Рис. 4 – Блок-схема функции *check variable_name(formula)*

find_index(expression, start)

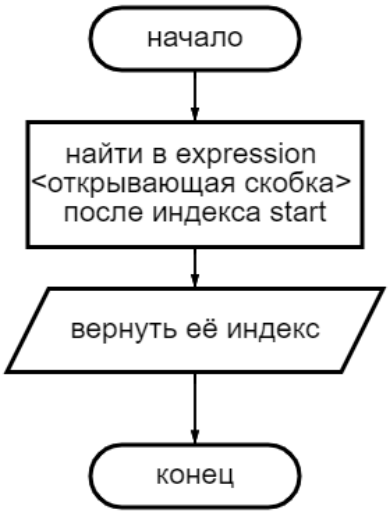

Рис. 5 – Блок-схема функции *find index(expression, start)*

checking_disjunction(expression)

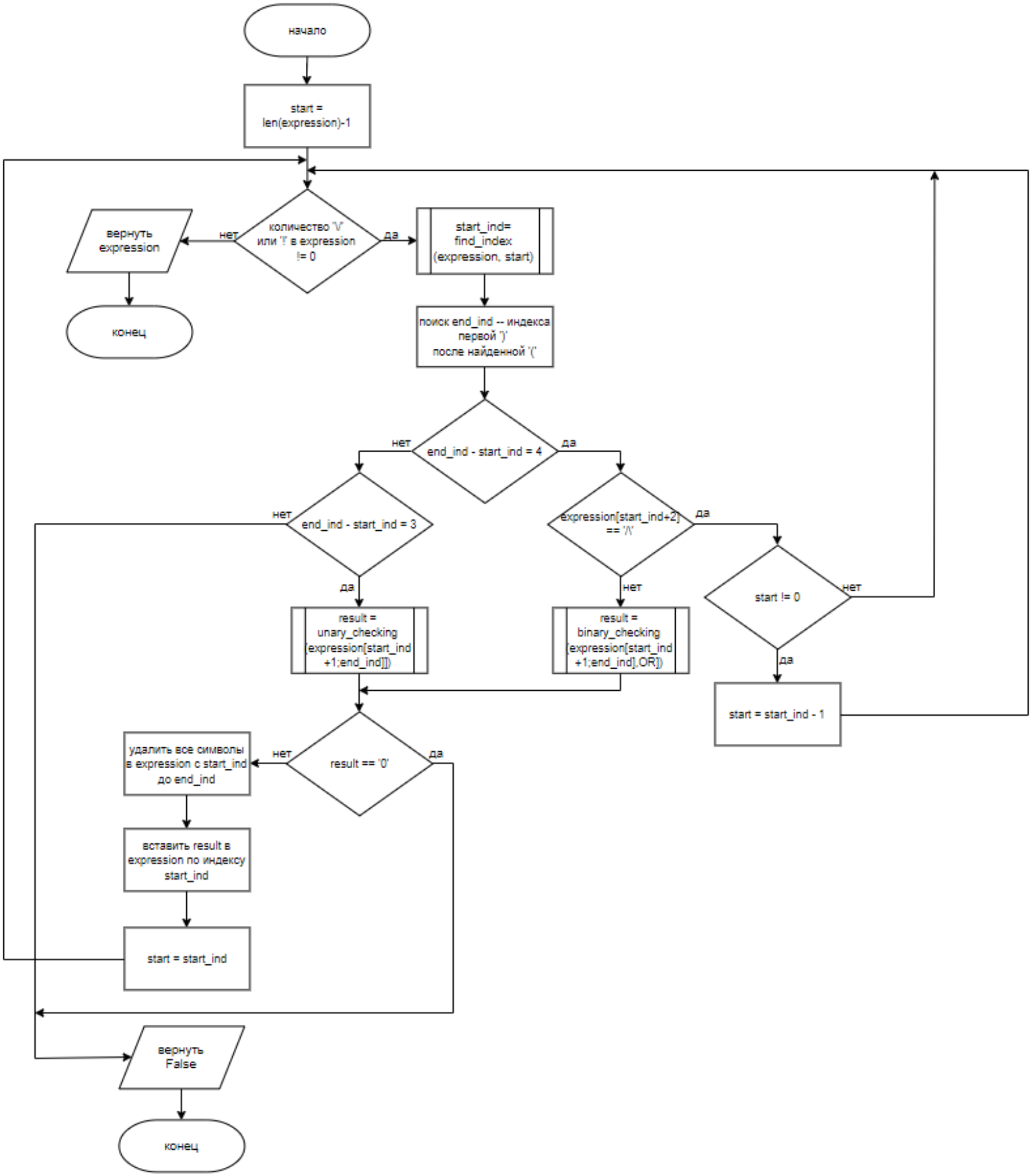

Рис. 6 – Блок-схема функции *checking disjunction(expression)*

checking(expression)

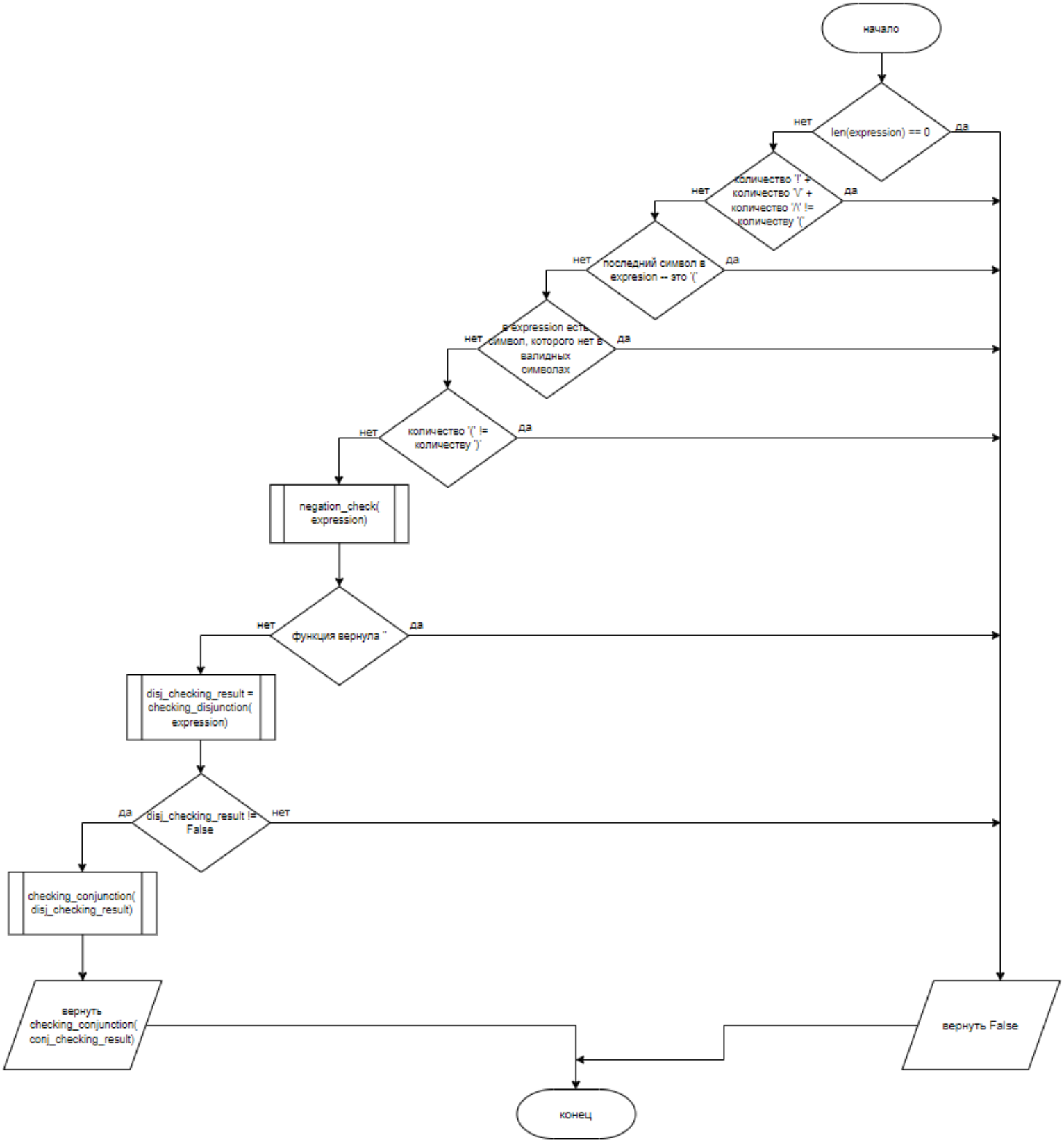

Рис. 7 – Блок-схема функции *checking(expression)*

binary_checking(expression, operation)

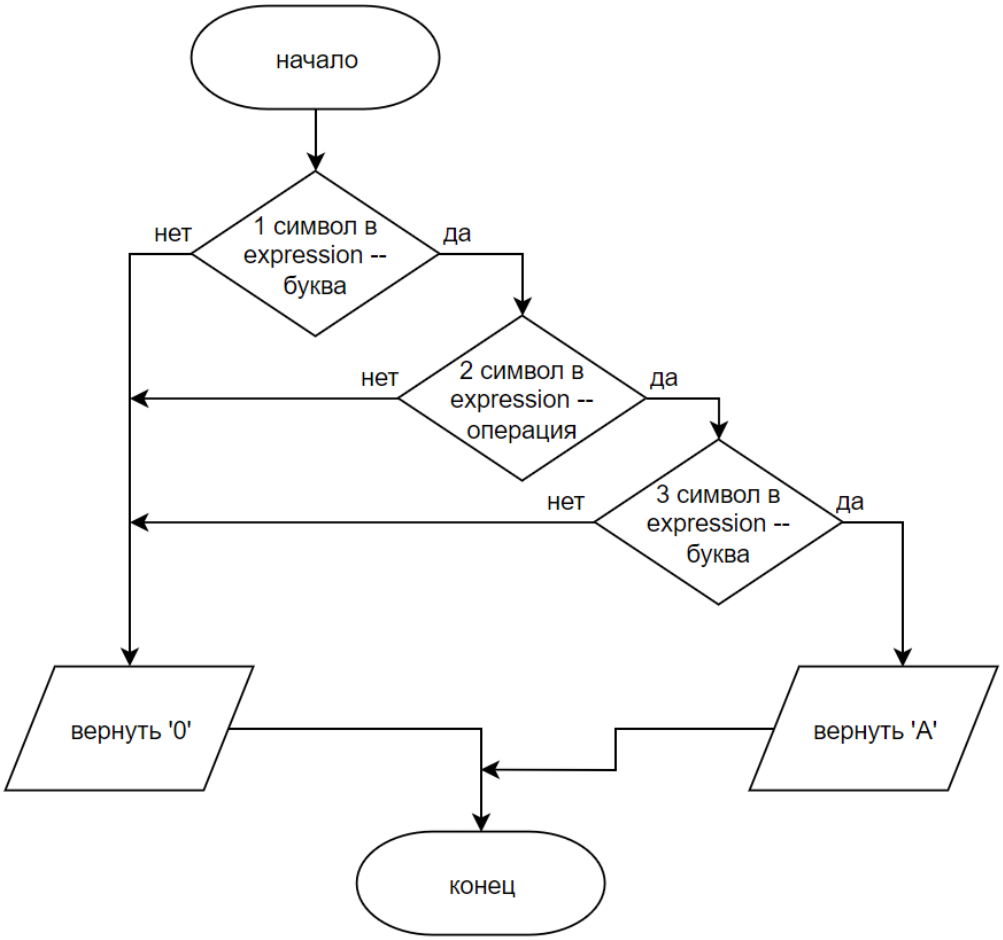

Рис. 8 – Блок-схема функции binary checking(expression, operation)

unary_checking(expression)

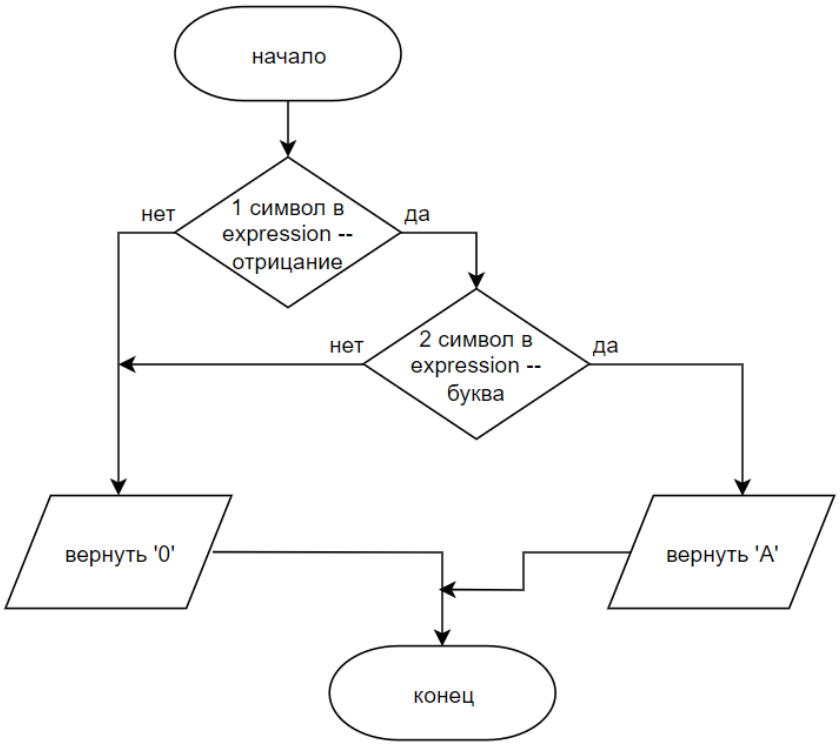

Рис. 9 – Блок-схема функции *unary checking(expression)*

checking_conjunction(expression)

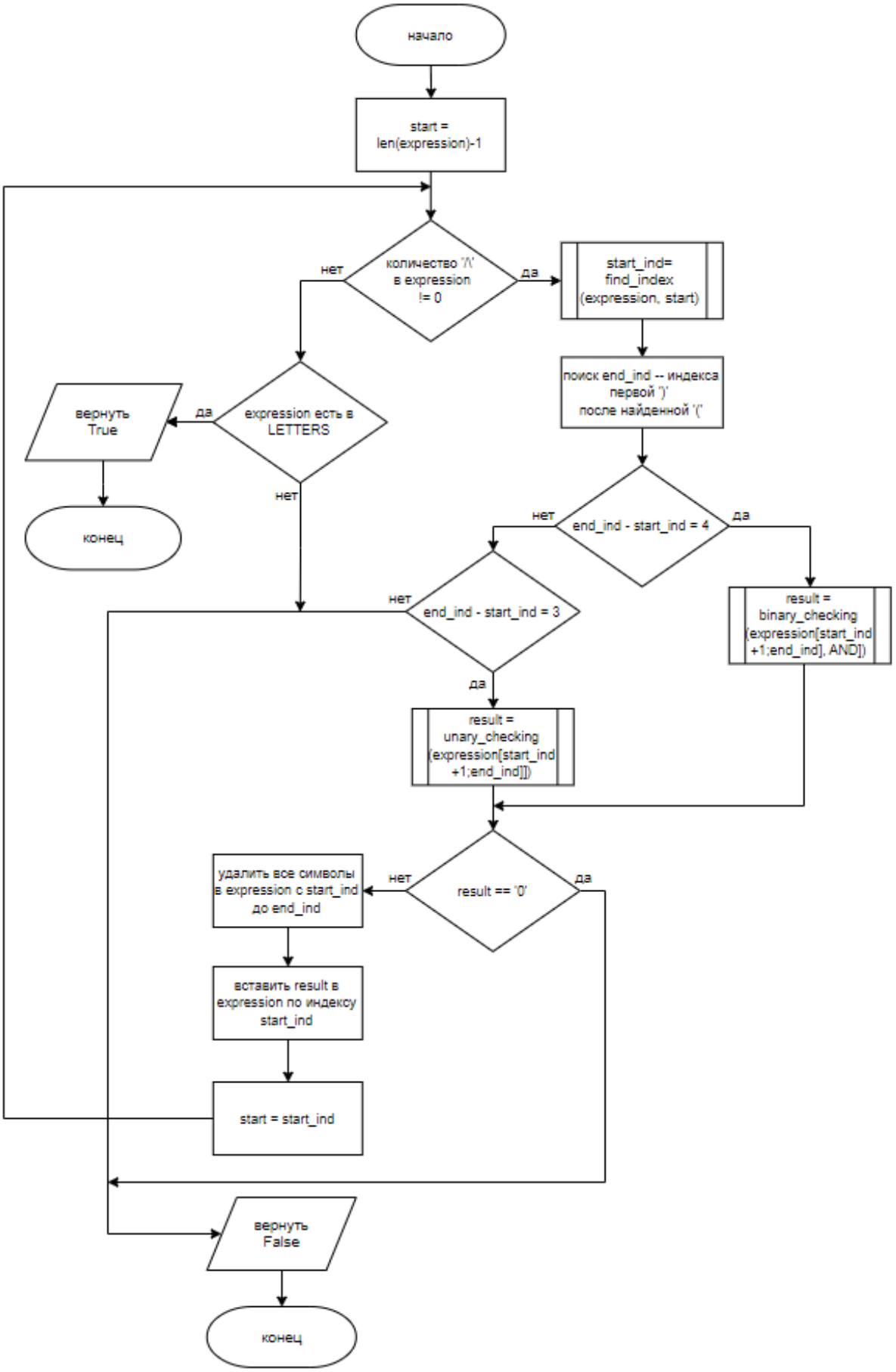

Рис. 10 – Блок-схема функции *checking conjunction(expression)*

Описание результатов тестирования:

Условия тестов:

```
def start_tests():
    expect_true('A')
    expect_true('(!A)')
    expect_false('(A)')
    expect_false('!A')
    expect_false('1')
    expect_false('0')
    expect_false('.L')
    expect_true('(A/\\((B\\/C)\\/D))')
    expect_false('A/\\B')
    expect_true('(A/\\B)')
    expect_false('()()')
    expect_false('((A->B)/\\(C\\/D))')
    expect_false('((A\\/B)/\\(C~D))')
    expect_false('(1/\\2)')
    expect_true('(A12/\\D)')
    expect_true('(A345/\\((((B34\\/C123)\\/D43)/\\(B34\\/((C123\\/M3)\\/N4))))')
    expect_false('(A345/\\((B0\\/C)\\/D))')
    expect_false('(!1/\\B)')
    expect_false(' ')
    expect_false('')
    expect_false('((\\/)/\\(\\/))')
    expect_false('(A/\\(B\\/C1)')
    expect_false('((!A)\\/(B/\\(!C))')
    expect_false('(((!C)\\/(A\\/ B))/\\((((!B)\\/(!C))\\/(!D))\\/A)\\/(A\\/((!C)\\/(!D))))')
    expect_true('((((!C)\\/(A\\/B))/\\((((!B)\\/(!C))\\/(!D))\\/A))/\\(A\\/((!C)\\/(!D))))')
    expect_true('(((!C)\\/(A\\/B))/\\(((!B)\\/(A\\/((!C)\\/(!D)))))')
    expect_false('((!A)/\\(B\\/(!C))/\\)')
    expect_false('((!A)\\/(!B))/\\((B\\/(!C))\\/))')
    expect_false('((A\\/B)(C\\/D))')
```

Результат тестов:

```
formula: A, the program returned: True, test passed
formula: (!A), the program returned: True, test passed
formula: (A), the program returned: False, test passed
formula: !A, the program returned: False, test passed
formula: 1, the program returned: False, test passed
formula: 0, the program returned: False, test passed
formula: .L, the program returned: False, test passed
formula: (A/\((B\/C)\/D)), the program returned: True, test passed
formula: A/\B, the program returned: False, test passed
formula: (A/\B), the program returned: True, test passed
formula: ()(), the program returned: False, test passed
formula: ((A->B)/\(C\/D)), the program returned: False, test passed
formula: ((A\/B)/\(C~D)), the program returned: False, test passed
formula: (1/\2), the program returned: False, test passed
formula: (A12/\D), the program returned: True, test passed
formula: (A345/\(((B34\/C123)\/D43)/\(B34\/((C123\/M3)\/N4)))), the program returned: True, test passed
formula: (A345/\((B0\/C)\/D)), the program returned: False, test passed
formula: (!1/\B), the program returned: False, test passed
formula:  , the program returned: False, test passed
formula: , the program returned: False, test passed
formula: ((\/)/\(\/)), the program returned: False, test passed
formula: (A/\(B\/C1), the program returned: False, test passed
formula: ((!A)\/(B/\(!C)), the program returned: False, test passed
formula: (((!C)\/(A\/ B))/\((((!B)\/(!C))\/(!D))\/A)\/(A\/((!C)\/(!D)))), the program returned: False, test passed
formula: ((((!C)\/(A\/B))/\((((!B)\/(!C))\/(!D))\/A))/\(A\/((!C)\/(!D)))), the program returned: True, test passed
formula: (((!C)\/(A\/B))/\(((!B)\/(A\/((!C)\/(!D)))))), the program returned: True, test passed
formula: ((!A)/\(B\/(!C))/\), the program returned: False, test passed
formula: ((!A)\/(!B))/\((B\/(!C))\/)), the program returned: False, test passed
formula: ((A\/B)(C\/D)), the program returned: False, test passed
```

Вывод:

В ходе выполнения лабораторной работы был получен навык синтаксического разбора формул языка логики высказываний. Были разработаны алгоритмы упрощения формулы и определения принадлежности формулы к классу КНФ. Были составлены схемы алгоритмов, произведена отладка программы, написаны тесты и предусмотрен случай ручного тестирования работоспособности программы.

Список использованных источников:

1. Логические основы интеллектуальных систем. Практикум : учеб.-метод. пособие / В. В. Голенков [и др.]. – Минск : БГУИР, 2011. – 70 с. : ил. ISBN 978-985-488-487-5.