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 A**

Выполнили студенты гр. 121702 Пунинский И.Д.

Промчук Д.В.

Проверил Ивашенко В. П.

Минск 2023

**Цель:** приобрести навыки алгоритмизации синтаксического разбора формул языка логики высказываний.

**Задача:** найти количество различных подформул формулы сокращённого языка логики высказываний

**Описание лабораторной работы:**

По ходу лабораторной работы необходимо подсчитать количество различных подформул заданной формулы. Для этого необходимо:

1. Проверить, принадлежат ли все символы формулы алфавиту языка логики высказываний.
2. Проверить, соответствует ли данная формула правилам синтаксиса языка логики высказываний.
3. Выделить все различные подформулы данной формулы.

Для этих задач были синтезированы следующие подзадачи

1. Анализ символов, входящих в строку, содержащую формулу
2. Проверка правильности расстановки скобок в формуле
3. Проверка правильности следования символов в строке
4. Выделение различных подформул:
   1. Выделение всех различных атомарных формул
   2. Выделение всех различных унарных сложных формул
   3. Выделение всех различных бинарных сложных формул

**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> ::= 1|0

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

<цифра> ::= 0|<ненулевая цифра>

<символ> ::= 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

<открывающая скобка> ::= (

<отрицание> ::= !

<закрывающая скобка> ::= )

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

<конъюнкция> ::= /\

<дизъюнкция> ::= \/

<импликация> :: = ->

<эквиваленция> ::= ~

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

<формула> ::= <логическая константа> | <атомарная формула> | <унарная сложная формула> | <бинарная сложная формула>

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

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

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

***Примеры разбора формулы на различные подформулы:***

(A\/B) ⊂ A, B, (A\/B)

((!A)\/(B/\A)) ⊂ A, (!A), B, (B/\A), ((!A)\/(B/\A))

((A\/B)/\(A\/B)) ⊂ A, B, (A\/B), ((A\/B)/\(A\/B))

((A\/B)/\(A/\B)) ⊂ A, B, (A\/B), (A/\B), ((A\/B)/\(A/\B))

((C/\B)\/((D/\E)\/G)) ⊂ C, B, D, E, G, (C/\B), (D/\E), ((D/\E)\/G), ((C/\B)\/((D/\E)\/G))

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

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

**Описание программы и алгоритма:**

Программа включает в себя структуру LogicalThing, которая описывает тип элемента формулы, а так же следующие функции:

BraсketCheck(string formula),

AlphabetCheck(string formula),

ParseFormula(string formula, vector <LogicalThing>& ParsedVector),

OneOperatorForOneBracketCheck(vector <LogicalThing> ParsedVector, int& index),

DoubleBracketCheck(vector <LogicalThing> ParsedVector),

LogicalCheck(string formula),

VariablesCheck(string formula, set <string>& variables),

CheckSymbolsAroundOperator(string formula, int i),

CheckOperators(string formula),

isFormula(string formula),

SearchForClosingBracket(string formula, int firstBracket, int& output),

AddSubformula(string formula, int firstBracket, int secondBracket, set <string>& subFormuls), FindSubformuls(string formula).

1. Функция BracketCheck(string formula) проверяет соответствие количества левых и правых скобок в полученной строке. При нахождении левой скобки счётчик увеличивается на 1, при нахождении правой скобки счётчик уменьшается на 1. Если по окончанию обхода строки счётчик равен 0, то количество левых и правых скобок равно.
2. Функция AlphabetCheck(string formula) проводит проверку каждого символа полученной строки на принадлежность алфавиту языка логики высказываний.
3. Функция ParseFormula(string formula, vector <LogicalThing>& ParsedVector) создаёт объекты типа LogicalThing, каждый из которых помечает соответствующим образом, и добавляет их в ParsedVector.
4. Функция OneOperatorForOneBracketCheck(vector <LogicalThing> ParsedVector, int& index) проверяет строку на наличие внутри неё символов логических операций.
5. Функция DoubleBracketCheck(vector <LogicalThing> ParsedVector) проверяет строку на предмет равенства числа пар скобок числу символов логических операций.
6. Функция LogicalCheck(string formula) выполняет проверку соответствия последующих символов синтаксису языка логики высказываний для каждого символа полученной строки.
7. Функция VariablesCheck(string formula, set<string>& variables) проверяет полученную строку на соответствие атомарных формул алфавиту языка логики высказываний.
8. Функция CheckSymbolsAroundOperator(string formula, int i) проверяет символ строки перед полученной позицией, и символ строки после следующей за полученной позицией на соответствие синтаксису языка логики высказываний.
9. Функция CheckOperators(string formula) проверяет правильность написания символов, состоящих из 2 элементов, обозначающих логические операции в полученной строке на соответствие заданному алфавиту.
10. Функция isFormula(string formula) проверяет является ли полученная строка формулой, вызывая все необходимые проверки .
11. Функция SearchForClosingBracket(string formula, int firstBracket, int& output) выполняет поиск позиции, на которой находится правая скобка, закрывающая левую скобку, позицию которой получил метод.
12. Функция AddSubformula(string formula, int firstBracket, int secondBracket, set<string>& subFormuls) считывает строку между полученными позициями скобок, добавляет прочитанную подформулу в набор.
13. Функция FindSubformuls(string formula) производит поиск всех подформул внутри полученной формулы.

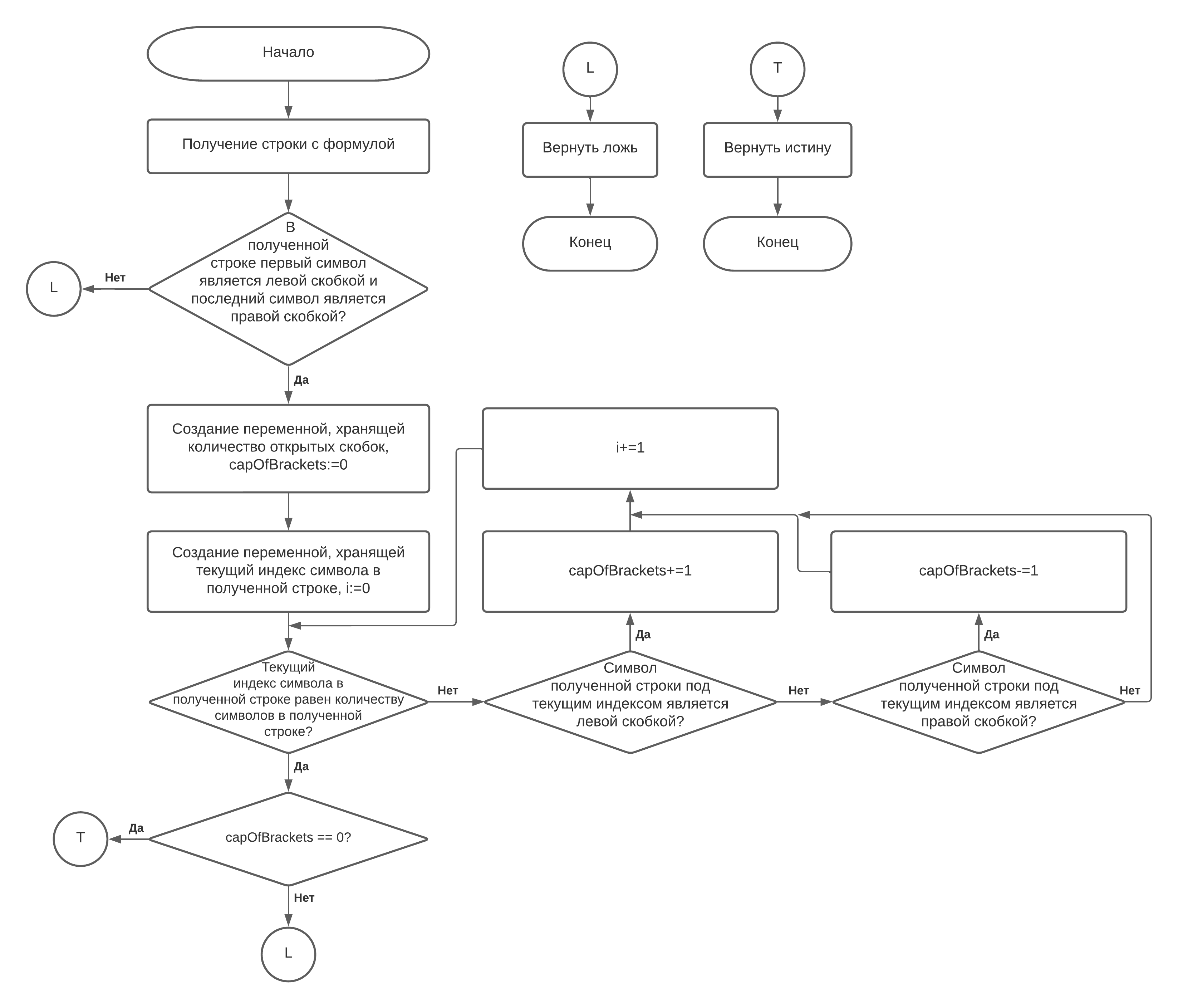


Рис. 1. Алгоритм функции BracketCheck

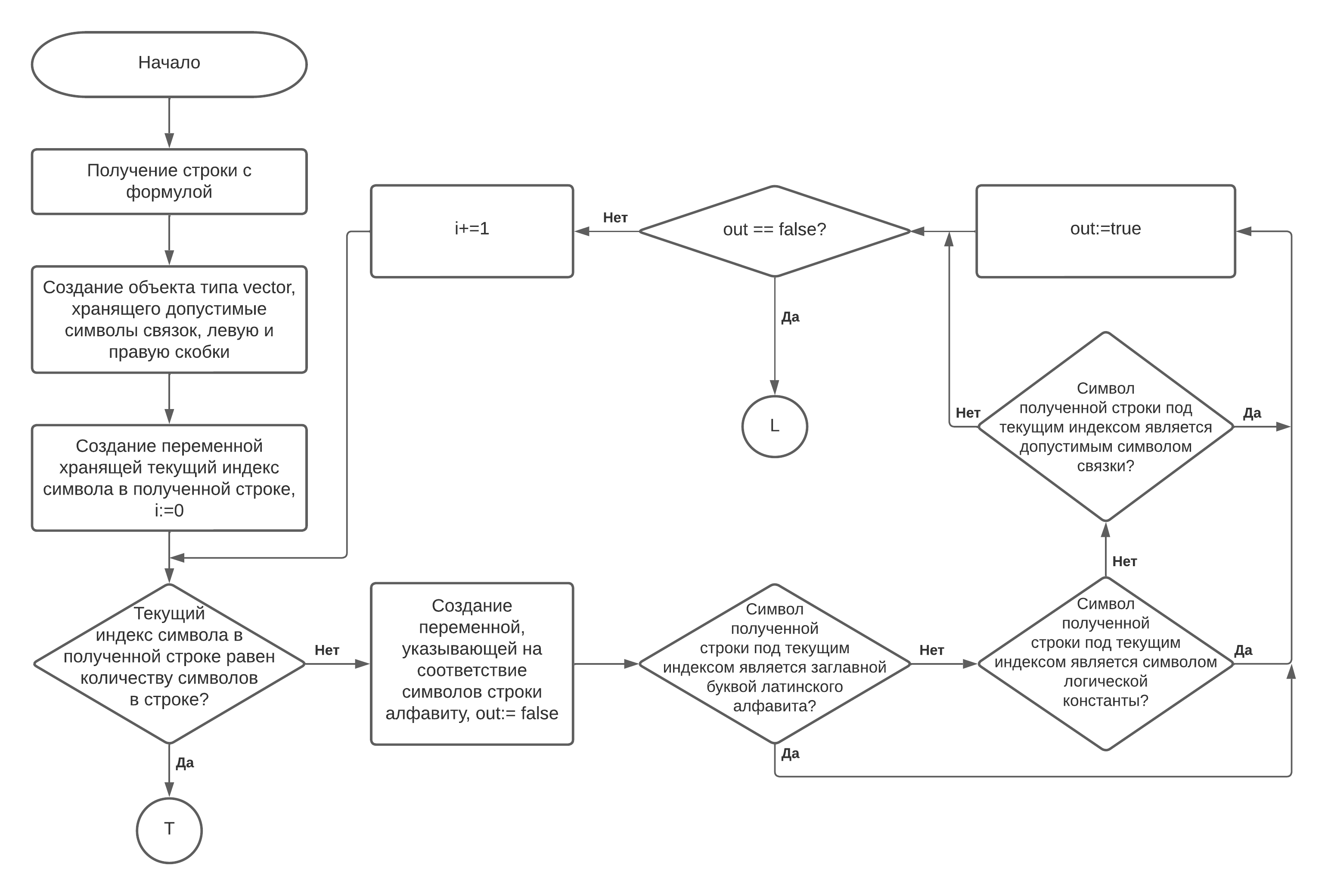


Рис 2. Алгоритм функции AlphabetCheck

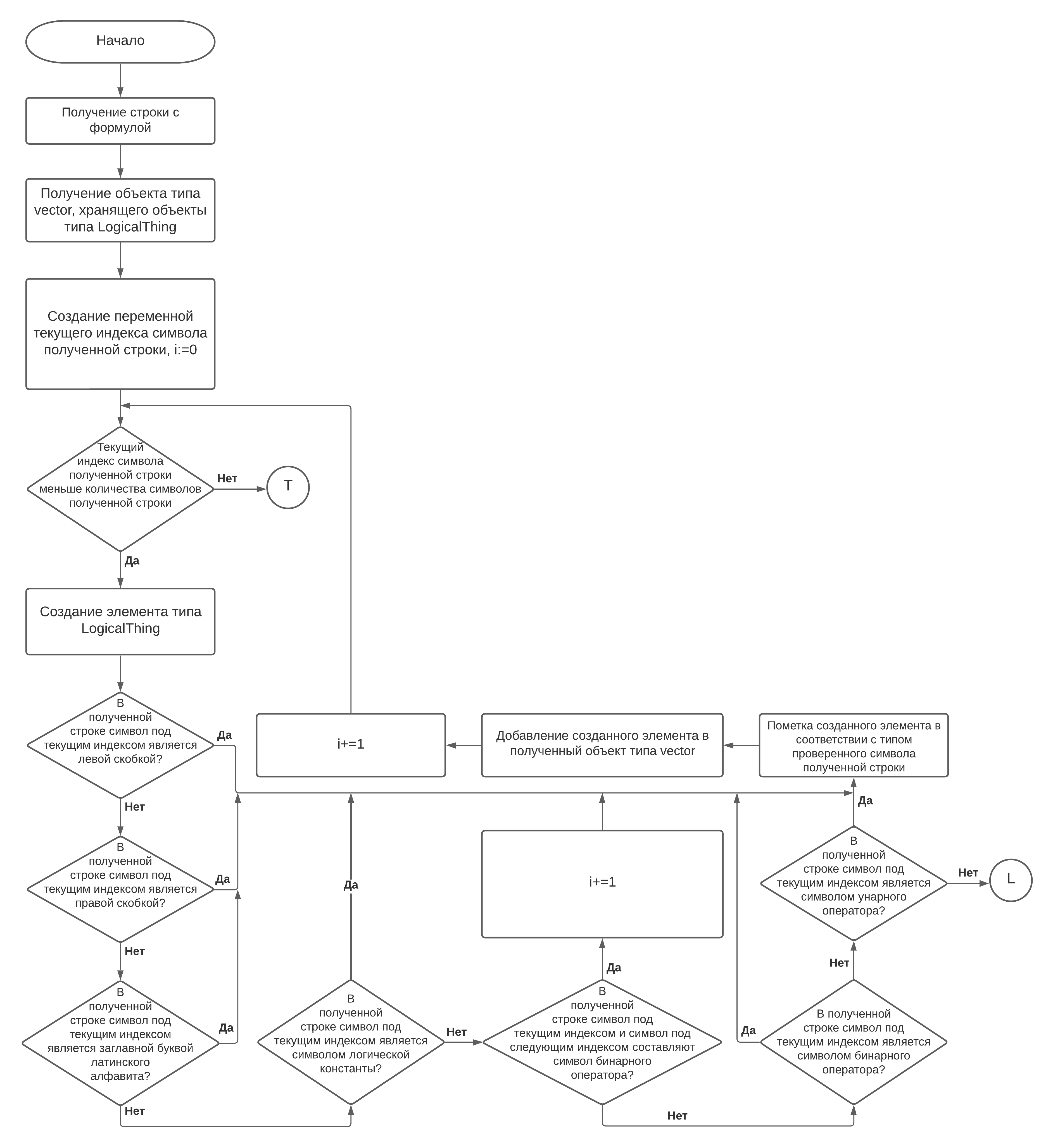


Рис. 3. Алгоритм функции ParseFormula

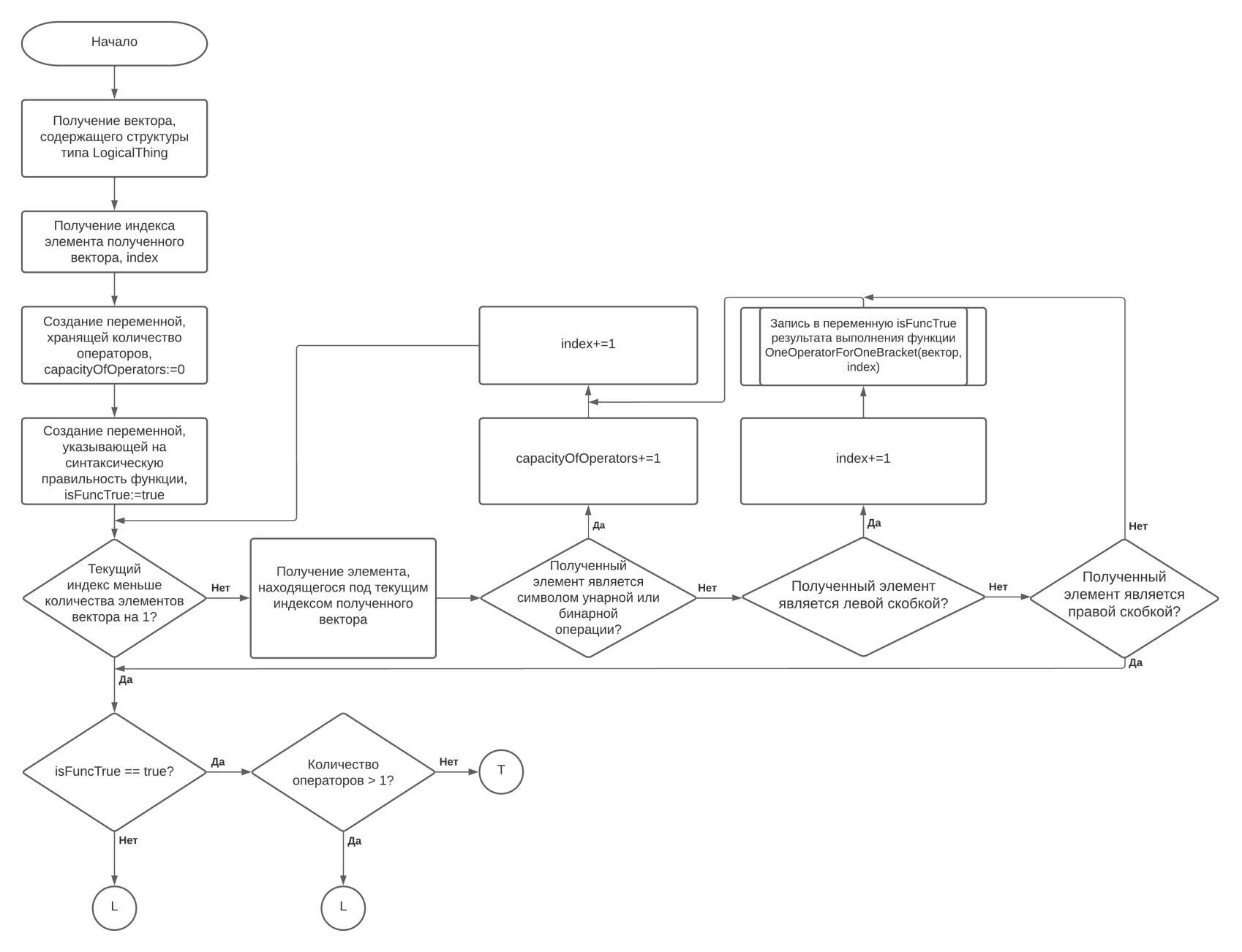


Рис. 4. Алгоритм функции OneOperatorForOneBracketCheck

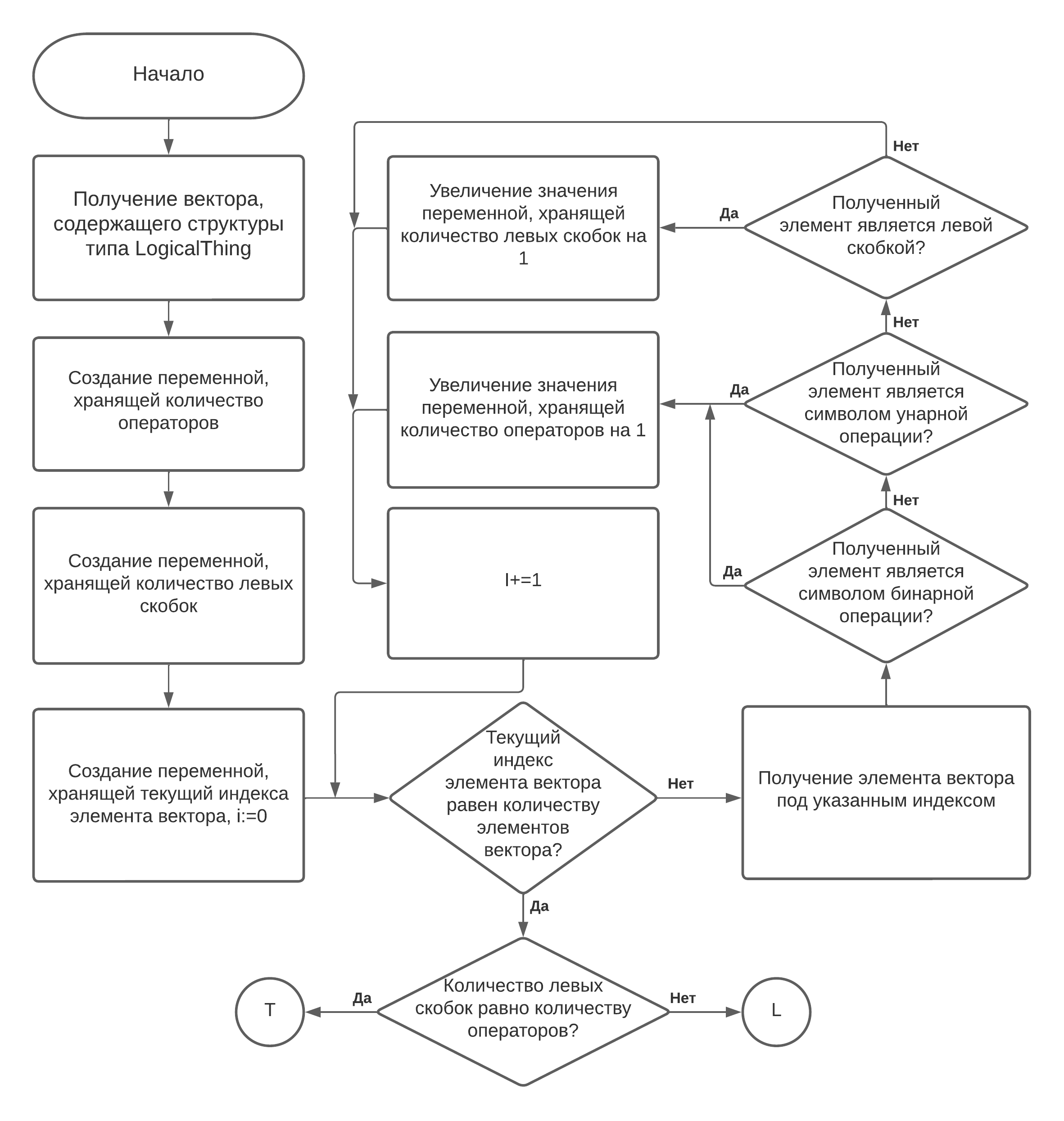


Рис. 5. Алгоритм функции DoubleBracketCheck

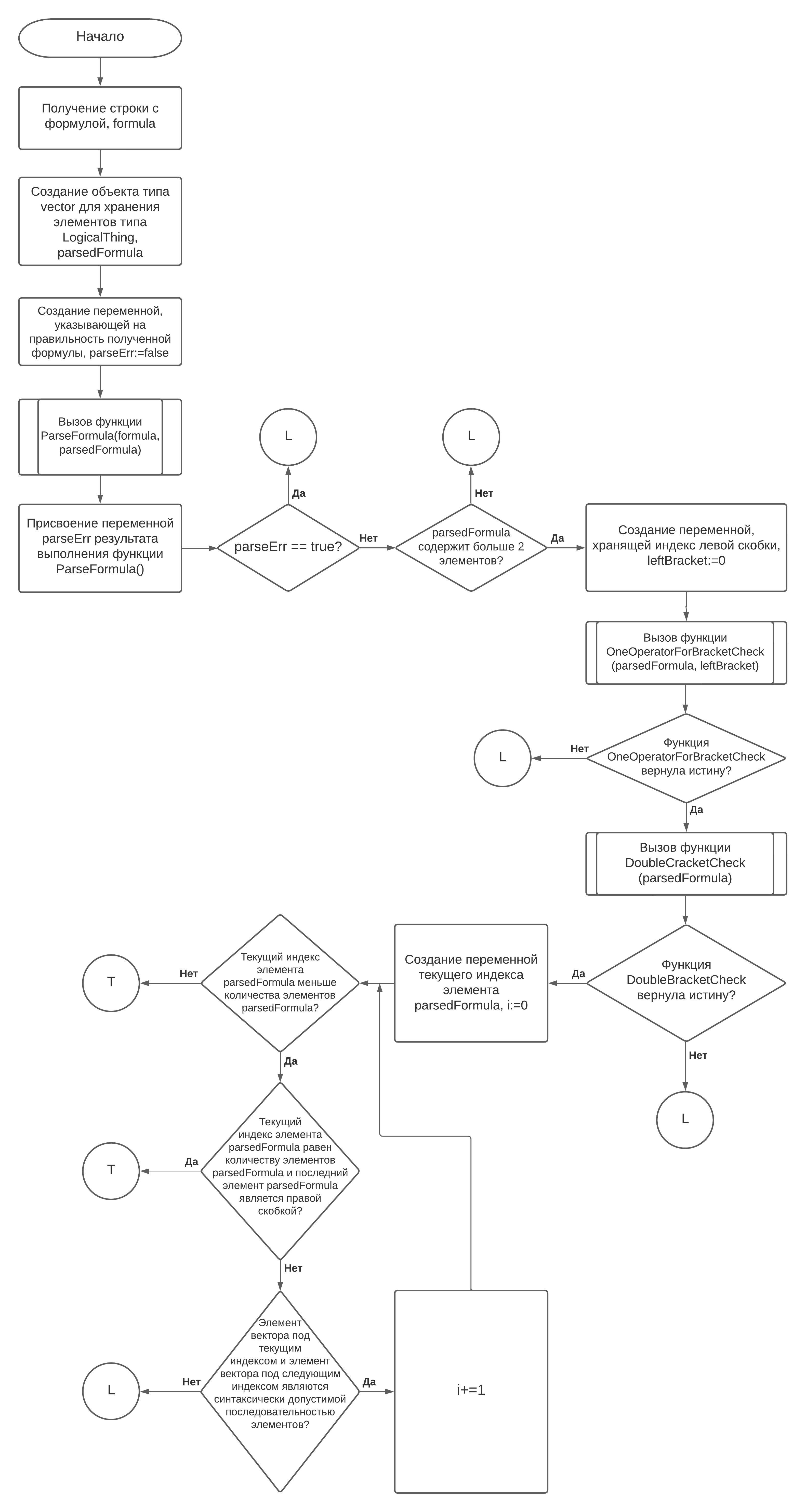


Рис. 6. Алгоритм функции LogicalCheck

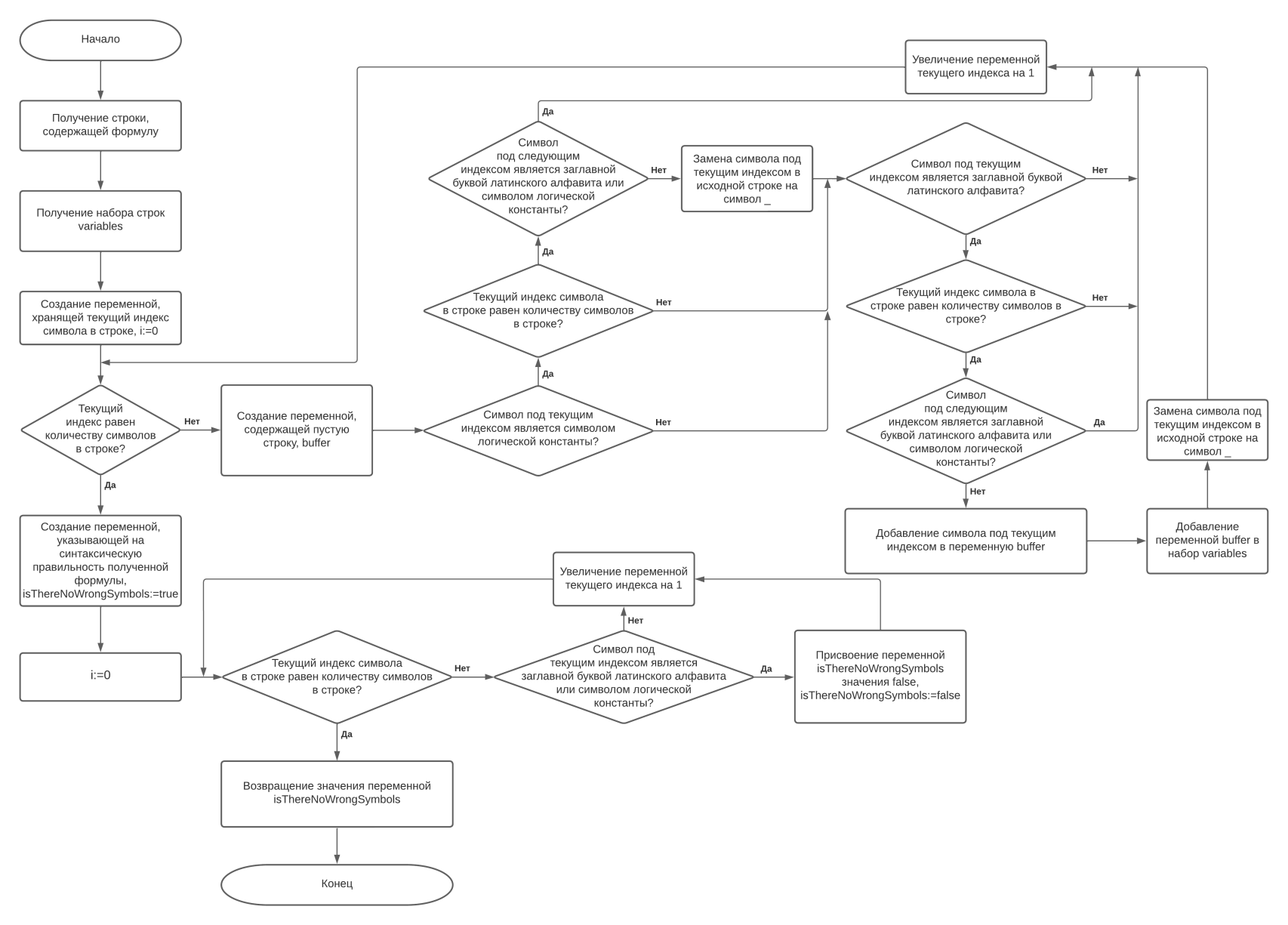


Рис. 7. Алгоритм функции VariablesCheck

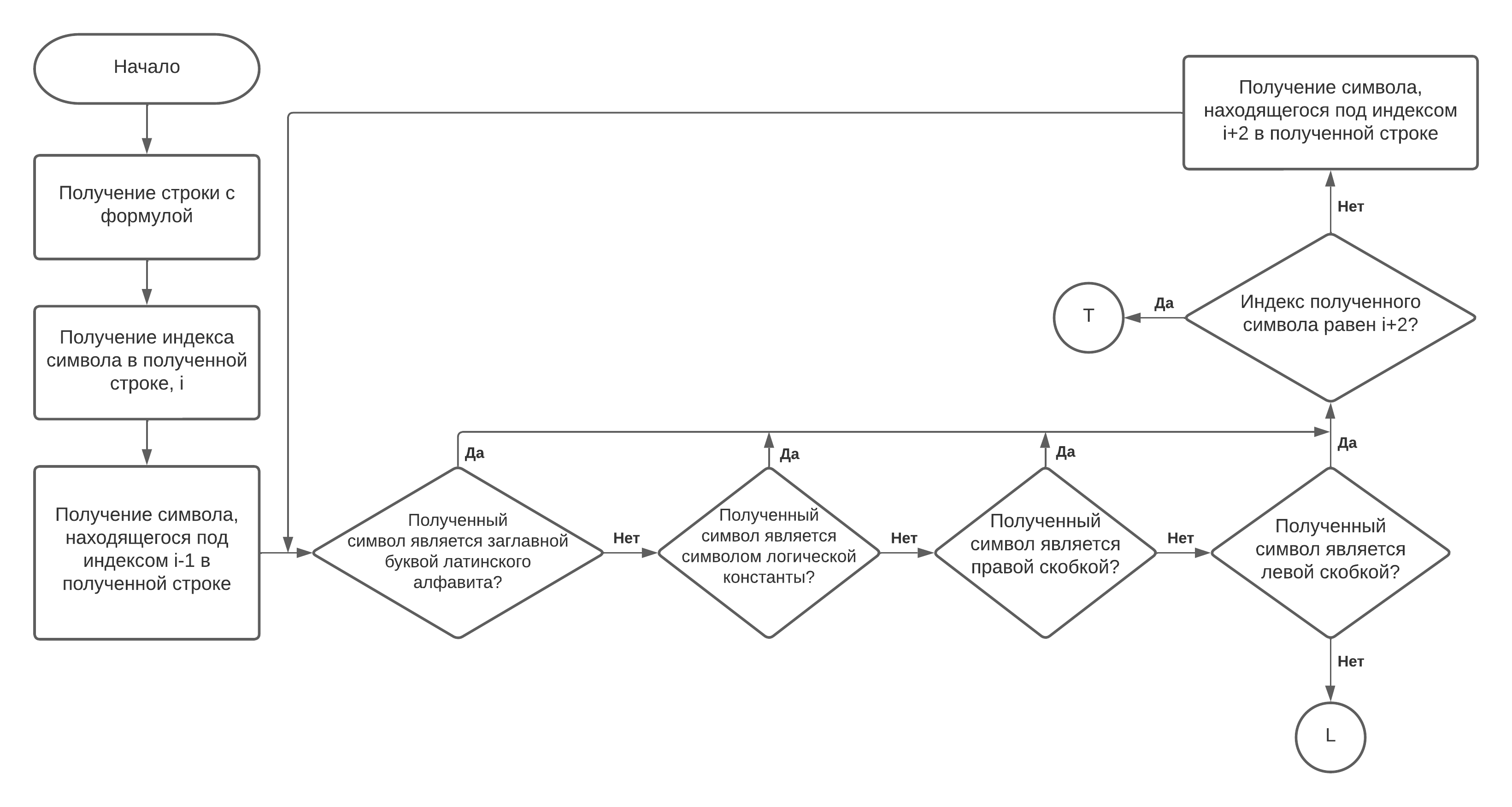


Рис. 8. Алгоритм функции CheckSymbolsAroundOperator

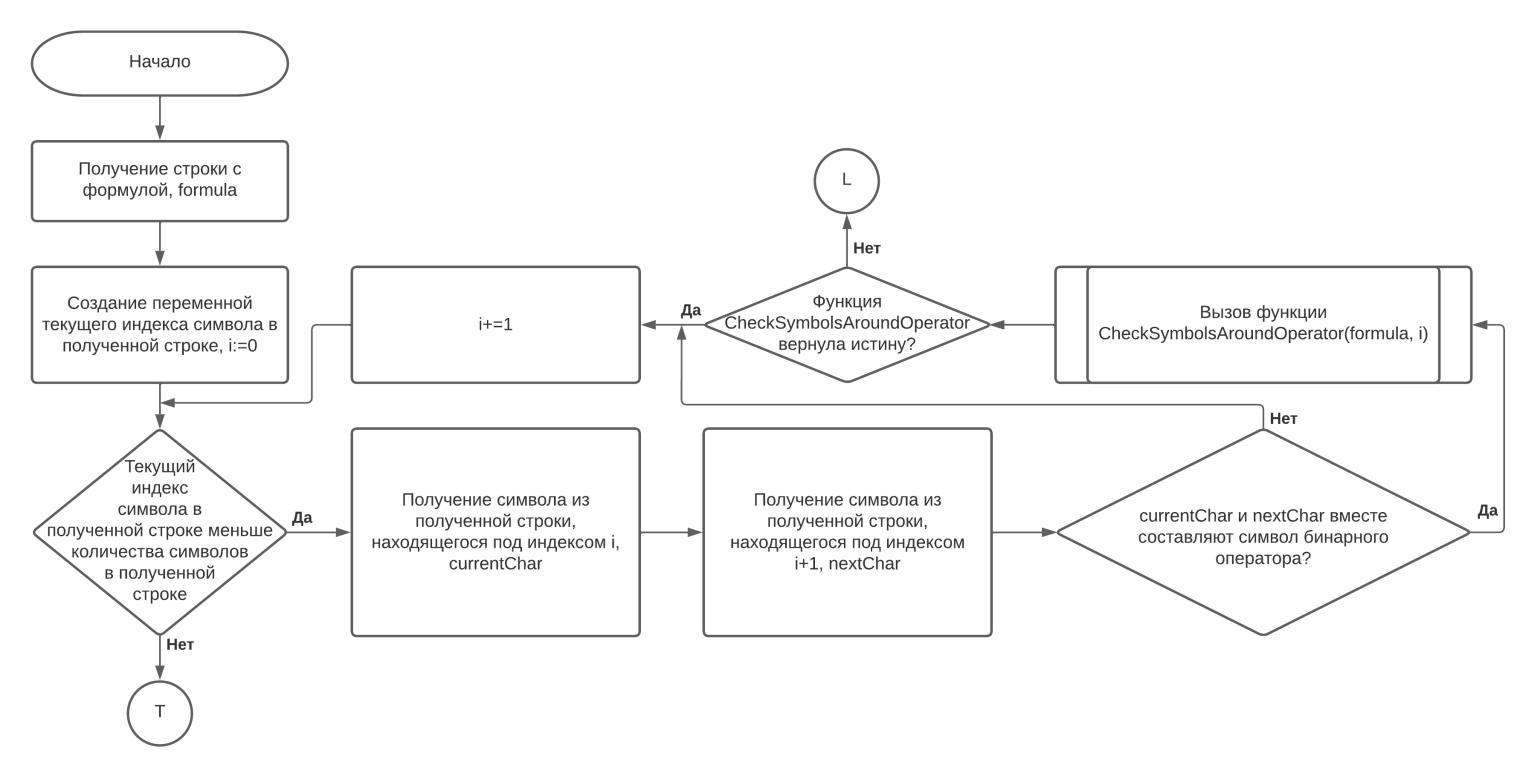


Рис. 9. Алгоритм функции CheckOperators

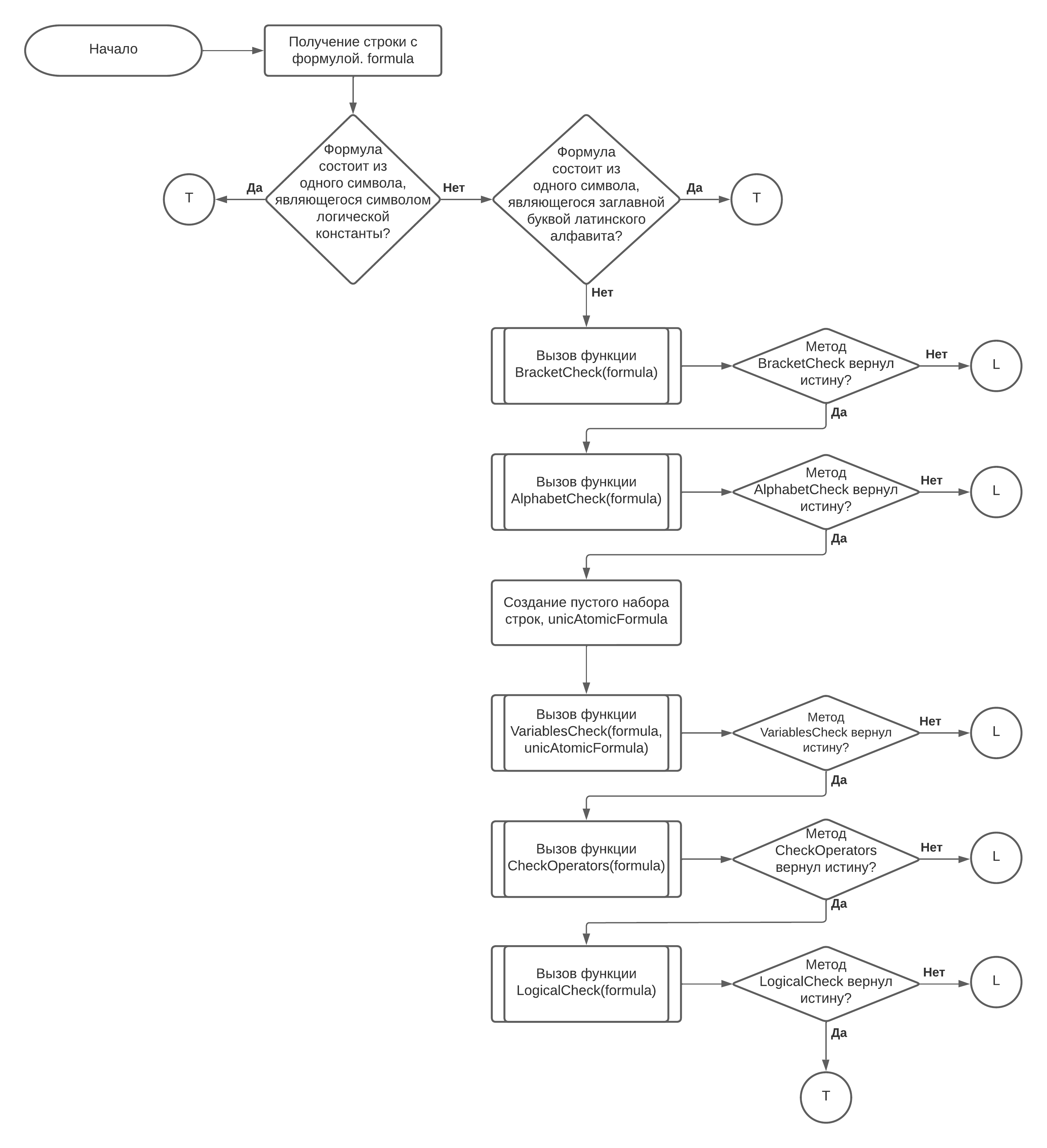


Рис. 10. Алгоритм функции isFormula

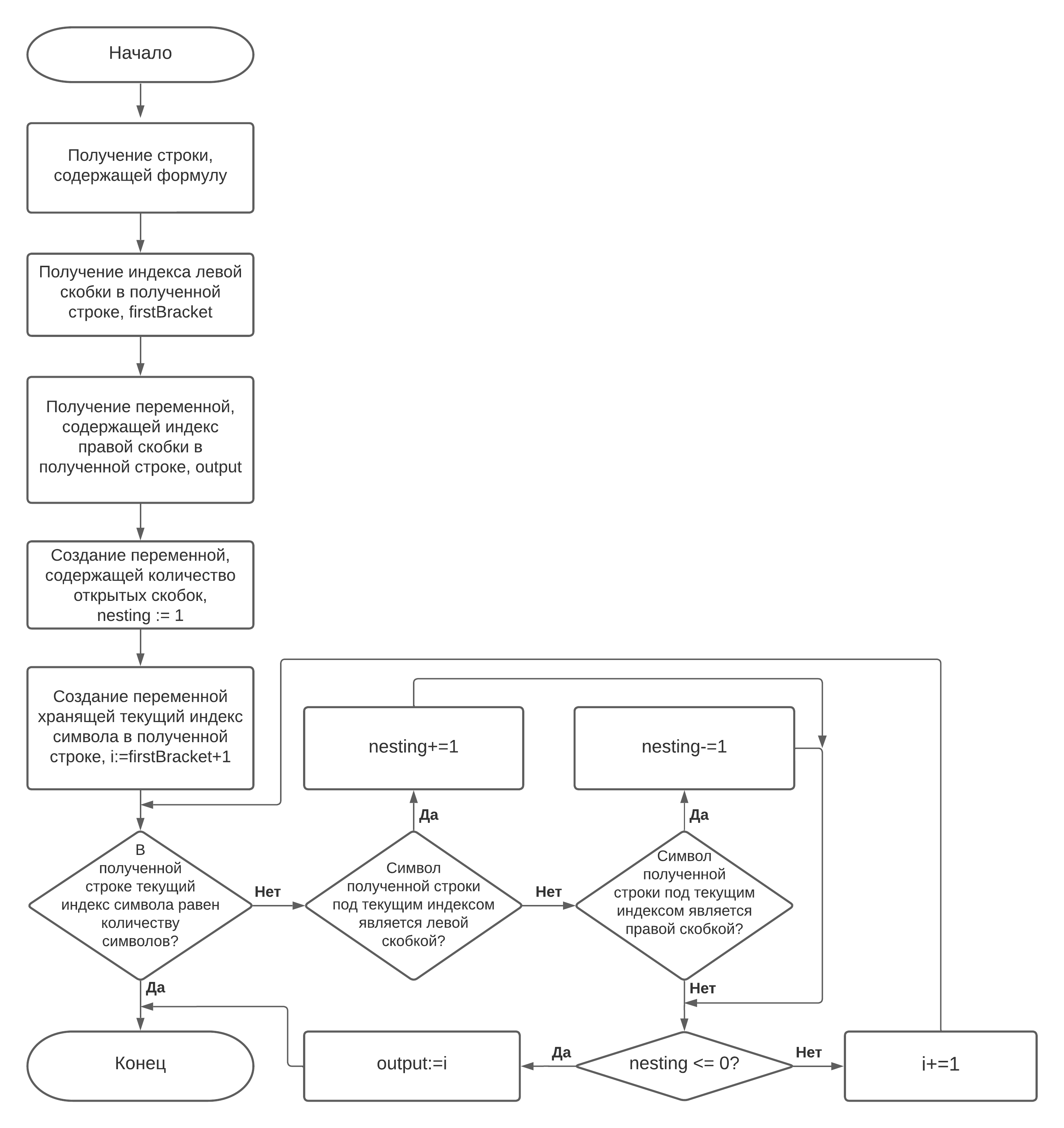


Рис. 11. Алгоритм функции SearchForClosingBracket

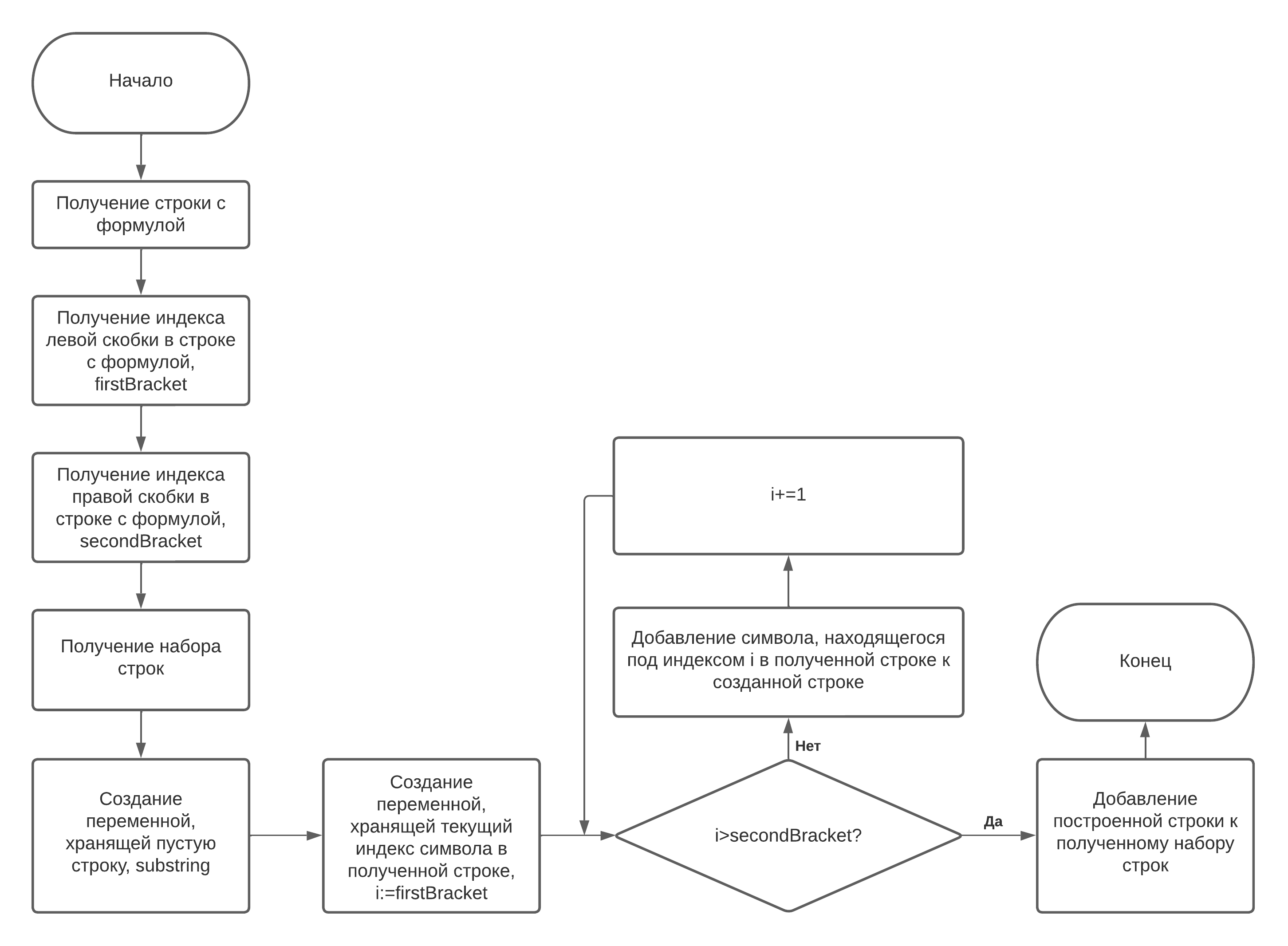


Рис. 12. Алгоритм функции AddSubformula

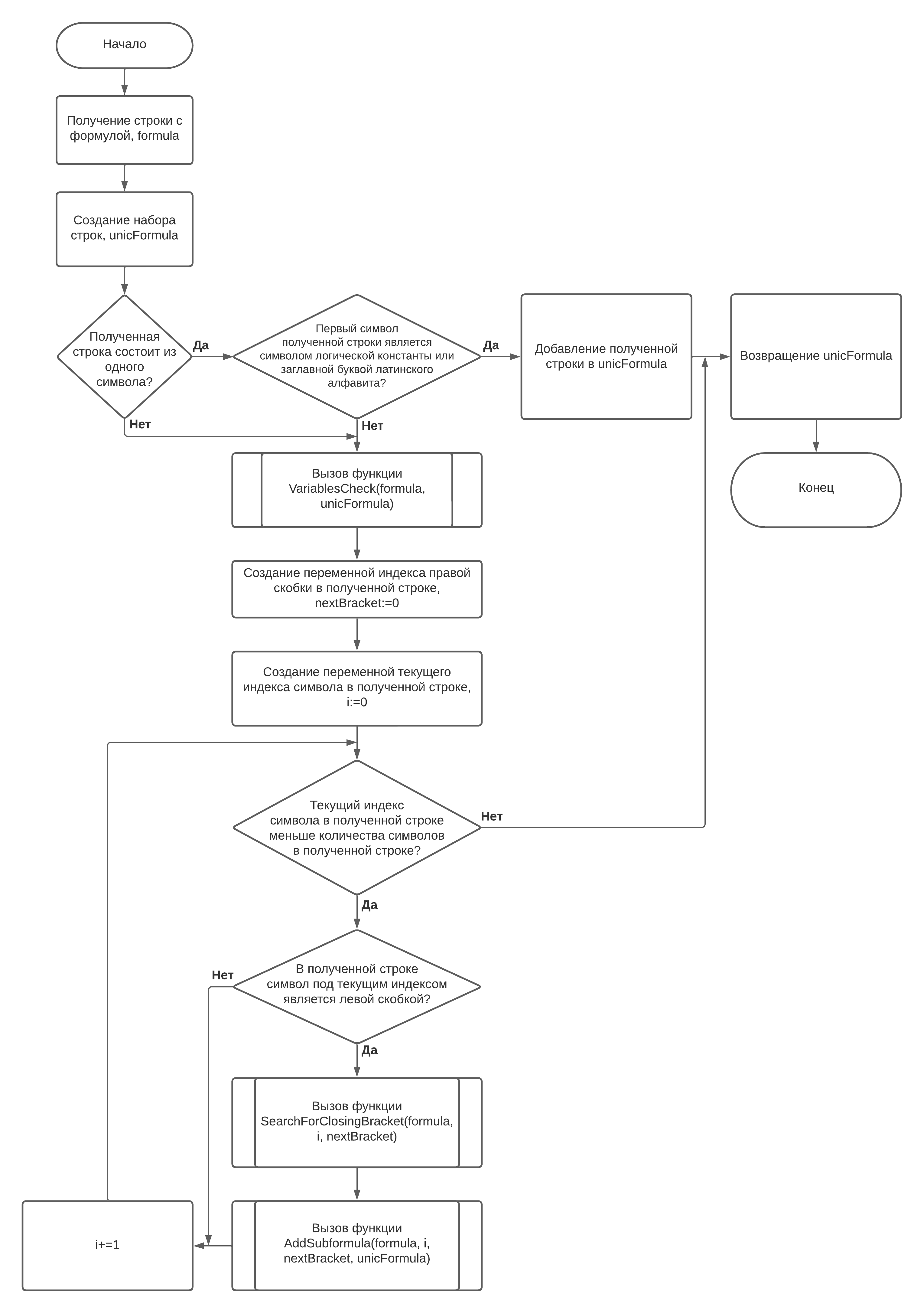


Рис. 13. Алгоритм функции FindSubformuls

**Строки, используемые для теста программы:**

Тест 1 : A

Тест 2 : (A\/B)

Тест 3 : ((C/\B)\/(D/\E))

Тест 4 : ((C/\B)\/((D/\E)\/G))

Тест 5 : (((A/\B)\/(F/\(E/\D)))\/((C/\B)\/(D/\E)))

Тест 6 : ((A/\B)

Тест 7 : A/\B

Тест 8 : (A)

Тест 9 : (!A)

Тест 10 : !(A)

Тест 11 : (!A\/B)

Тест 12 : ((!A)/\B)

Тест 13 : ((!!A)/\B)

Тест 15 : (!!A/\B)

Тест 16 : (A/\B\/C)

Тест 17 : (A9/\B)

Тест 18 : (A0/\B)

Тест 19 : ((A\/B)/\(A\/B))

Тест 20 : (AB\/B)

Тест 21 : (B\\C)

Тест 22 : (A//K)

**Тесты программы:**

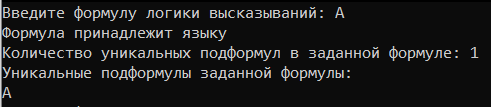


Рис. 14. Тест 1

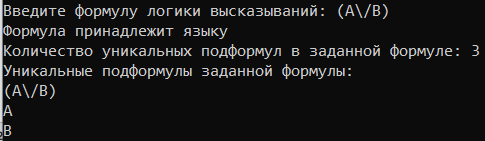


Рис. 15. Тест 2

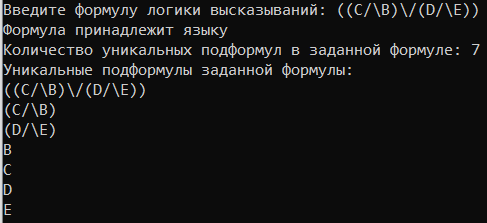


Рис. 16. Тест 3

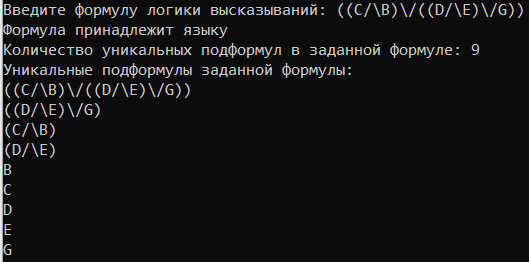


Рис. 17. Тест 4

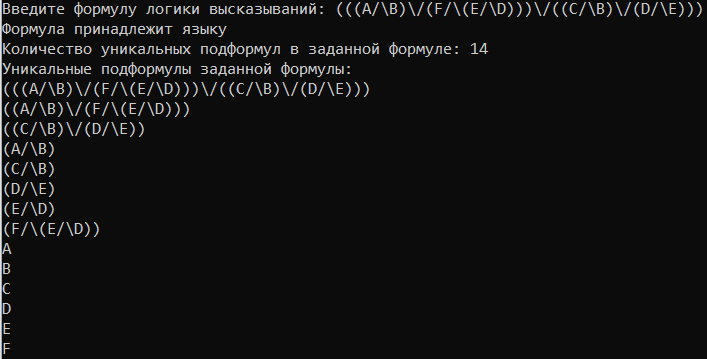


Рис. 18. Тест 5



Рис. 19. Тест 6



Рис. 20. Тест 7



Рис. 21. Тест 8

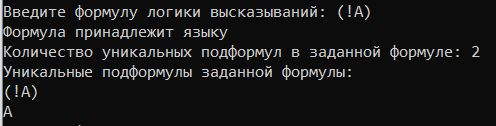


Рис. 22. Тест 9



Рис. 23. Тест 10



Рис. 24. Тест 11

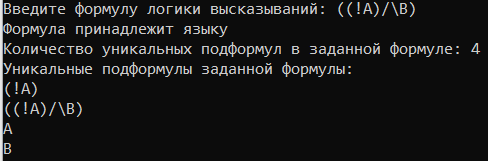


Рис. 25. Тест 12



Рис. 26. Тест 13



Рис. 27. Тест 14



Рис. 28. Тест 15



Рис. 29. Тест 16



Рис. 30. Тест 17

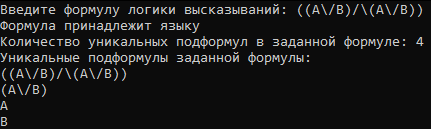


Рис. 31. Тест 18

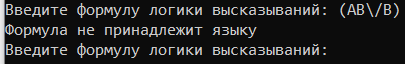


Рис. 32. Тест 19



Рис. 33. Тест 20



Рис. 34. Тест 21

**Вывод:**

В процессе выполнения лабораторной работы были приобретены навыки синтаксического анализа формул языка логики высказываний. Это было достигнуто через создание и реализацию алгоритмов, описанных ранее. Были разработаны блок-схемы для каждого из использованных алгоритмов, проведена отладка программы и осуществлено ручное тестирование окончательного результата.

**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.

1. <https://en.cppreference.com/w/cpp/container/vector>
2. <https://en.cppreference.com/w/cpp/container/set>
3. <https://en.cppreference.com/w/cpp/string>
4. https://github.com/Danshicu/Lois\_labs