МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»
(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа № 1

по дисциплине: Математическая логика и теория алгоритмов
тема: «Логика высказываний»

Выполнил: ст. группы ПВ-202
Аладиб Язан

Проверил:

Куценко Дмитрий Александрович
Бондаренко Татьяна Владимировна

Белгород 2021 г.

Теоретическая часть :

Задание варианта №2:

Номер варианта		Номера задач											
2		2.5	7.2	8.17	10.4	12.1	14.5	18	23.4	30.3	35.9	38.5	49.1

2. Запишите символически следующие фразы:

5) Четырёхугольник является квадратом тогда и только тогда, когда все его стороны и все углы равны

пусть :
A – стороны четырехугольника равны
B – углы четырехугольника равны
C – четырехугольник квадрат

Тогда : $(A \wedge B) \leftrightarrow C$

7. Постройте таблицы истинности, соответствующие следующим формулам:

2) $X \& \overline{Y}$

X	Y	$\overline{Y}$	$X \& \overline{Y}$
0	0	1	0
0	1	0	0
1	0	1	1
1	1	0	0

8. Используя таблицы истинности, докажите равносильность формул:

17) $((A \vee B)\&(A \vee \overline{B})) \equiv A$

A	B	$\overline{B}$	$A \vee B$	$A \vee \overline{B}$	$((A \vee B)\&(A \vee \overline{B}))$	$((A \vee B)\&(A \vee \overline{B})) \equiv A$
0	0	1	0	1	0	1
0	1	0	1	0	0	1
1	0	1	1	1	1	1
1	1	0	1	1	1	1

10. Используя равносильные преобразования докажите, выполняются ли следующие соотношения:

4) $X \to Y \equiv \overline{X\bar{Y}}$

$X \to Y = \underbrace{\bar{X} \vee Y}_{\text{выражение импликации через дизъюнкцию и отрицание.}}$

$\overline{X\bar{Y}} = \underbrace{\bar{X} \vee \bar{\bar{Y}}}_{\text{Законы де М´органа}} = \underbrace{\bar{X} \vee Y}_{\text{правило снятия двойного отрицания}}$

Тогда : $X \to Y \equiv \overline{X\bar{Y}} \equiv \bar{X} \vee Y$

12. Найдите ДНФ для следующих формул:

1)$X\&(\bar{X} \vee \bar{Y})$

$X\&(\bar{X} \vee \bar{Y}) = \underbrace{(X\&\bar{X}) \vee (X\&\bar{Y})}_{\text{законы дистрибутивности}} = \underbrace{0 \vee (X\&\bar{Y})}_{\text{законы противоречия}} = X\&\bar{Y}$

X	Y	$\bar{X}$	$\bar{Y}$	$\bar{X} \vee \bar{Y}$	$X\&(\bar{X} \vee \bar{Y})$
0	0	1	1	1	0
0	1	1	0	1	0
1	0	0	1	1	1
1	1	0	0	0	0
ДНФ = $X\&\bar{Y}$					

14. Упростите вид следующих формул, используя равносильные преобразования:

5) $\overline{\bar{X}\&\bar{Y}} \vee (X \to Y)\&X$

$\overline{\bar{X}\&\bar{Y}} \vee \underbrace{(X \to Y)}_{\text{выражение импликации}}\&X = \overline{\bar{X}\&\bar{Y}} \vee \underbrace{(\bar{X} \vee Y)\&X}_{\text{Законы дистрибутивности}} = \overline{\bar{X}\&\bar{Y}} \vee \underbrace{(X\&\bar{X})}_{\text{закон противоречия}} \vee (X\&Y) =$

$= \underbrace{\overline{\bar{X}\&\bar{Y}}}_{\text{Законы де Моргана}} \vee 0 \vee (X\&Y) = \bar{\bar{X}} \vee \underbrace{\bar{\bar{Y}} \vee 0}_{\text{Свойства нуля}} \vee (X\&Y) = \underbrace{\bar{\bar{X}}}_{\text{правило снятия двойного отрицания}} \vee \bar{\bar{Y}} \vee (X\&Y) =$

$\underbrace{X \vee \bar{\bar{Y}}}_{\text{правило снятия двойного отрицания}} \vee (X\&Y) = X \vee \underbrace{Y \vee (X\&Y)}_{\text{Законы поглощения}} = X \vee Y$

18. Возможна ли формула, которая находится и в КНФ, и в ДНФ? Если да, то приведите пример.

Да , Только если она является конъюнктом или дизъюнктом.

например :

$(X \vee ((X \wedge Y) \vee (\bar{X} \wedge \bar{Y} \wedge \bar{Z}))) \wedge Z$

$(X \vee ((X \wedge Y) \vee (\bar{X} \wedge \bar{Y} \wedge \bar{Z}))) \wedge Z \equiv (X \vee (\bar{X} \wedge \bar{Y} \wedge \bar{Z})) \wedge Z \equiv (X \vee \bar{X}) \wedge (X \vee \bar{Y}) \wedge (X \vee \bar{Z}) \wedge Z \equiv$

$\equiv (X \vee \bar{Y}) \wedge (X \vee \bar{Z}) \wedge Z \equiv (X \vee (\bar{Y} \wedge \bar{Z})) \wedge Z \equiv (X \wedge Z) \vee (\bar{Y} \wedge \bar{Z} \wedge Z) \equiv X \wedge Z$

$X \wedge Z$ Следовательно, $X \wedge Z$ действительно является КНФ/ДНФ.

23. Приведением к совершенным нормальным формам докажите неравносильность следующих формул:

4) $XY \vee Z \not\equiv X\&(Y \vee Z)$

$$XY \vee Z$$

X	Y	Z	XY	$XY \vee Z$
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	0	1
1	0	0	0	0
1	0	1	0	1
1	1	0	1	1
1	1	1	1	1
$XY \vee Z = \bar{X}\bar{Y}Z \vee \bar{X}YZ \vee X\bar{Y}Z \vee XY\bar{Z} \vee XYZ$				

$$X\&(Y \vee Z)$$

X	Y	Z	$Y \vee Z$	$X\&(Y \vee Z)$
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1
$X\&(Y \vee Z) = X\bar{Y}Z \vee XY\bar{Z} \vee XYZ$				

из заданной ДНФ видно, что $XY \vee Z \not\equiv X\&(Y \vee Z)$

30. Для следующих выражений найдите двойственные:

3) $X \vee 0$

$\mathfrak{A} = X \vee 0$

$\mathfrak{A}^* = X\& 0 \equiv 0$

35. Выяснить, является ли первая формула логическим следствием остальных:

9) $X \to Y; X \to Y, \bar{X}, Z$

$$((X \to Y) \wedge \bar{X} \wedge Z) \to (X \to Y) = \overline{\underbrace{(\bar{X} \vee Y) \wedge \bar{X}}_{\text{закон поглощения}} \wedge Z} \vee (\bar{X} \vee Y) =$$

$$= \overline{\bar{X} \wedge Z} \vee (\bar{X} \vee Y) = X \vee \bar{Z} \vee \bar{X} \vee Y = 1$$

Поскольку формула абсолютно верна, то она является логическим следствием ($X \to Y, \bar{X}, Z$)

38. Докажите правильность умозаключений:

5) $\frac{A \to B, B \to C, \ A}{C}$

$$((A \to B) \wedge (B \to C) \wedge A) \to C$$

A	B	C	$A \to B$	$B \to C$	C
0	0	0	1	1	0
0	0	1	1	1	1
0	1	0	1	0	0
0	1	1	1	1	1
1	0	0	0	1	0
1	0	1	0	1	1
1	1	0	1	0	0
1	1	1	1	1	1

49. Найдите все следствия из посылок:

1) $X \to Y, \bar{Y}$

$(X \to Y) \wedge \bar{Y} \equiv (X \vee \bar{Y}) \wedge (\bar{X} \vee Y) \wedge (\bar{X} \vee \bar{Y})$

Следствия :

1. $X \vee \bar{Y} \equiv Y \to X$

2. $\bar{X} \vee Y \equiv X \to Y$

3. $\bar{X} \vee \bar{Y} \equiv \overline{X \wedge Y}$

4. $(X \vee \bar{Y}) \wedge (\bar{X} \vee Y) \equiv X \leftrightarrow Y$
5. $(X \vee \bar{Y}) \wedge (\bar{X} \vee \bar{Y}) \equiv \bar{Y} \vee (X \wedge \bar{X}) \equiv \bar{Y} \vee 0 \equiv \bar{Y}$
6. $(\bar{X} \vee Y) \wedge (\bar{X} \vee \bar{Y}) \equiv \bar{X} \vee (Y \wedge \bar{Y}) \equiv \bar{X} \vee 0 \equiv \bar{X}$
7. $(X \vee \bar{Y}) \wedge (\bar{X} \vee Y) \wedge (\bar{X} \vee \bar{Y}) \equiv \bar{X} \wedge \bar{Y}$

Практическая часть :

1) Программа должна строить полную таблицу истинности введённой формулы.
3) Программа должна доказывать противоречивость введённой формулы.

Файл mathlogic.h :

```
#ifndef MATHLOGIC
#define MATHLOGIC

#include <iostream>
#include <string>
#include <set>
#include <map>
#include <stack>
#include <queue>
#include <cassert>
#include <cstdlib>
#include <vector>

// Объявление типов.
// Токен (лексема):
typedef char Token;
// Стек токенов:
typedef std::stack<Token> Stack;
// Последовательность токенов:
typedef std::queue<Token> Queue;
// Множество различных токенов:
typedef std::set<Token> Set;
// Таблица значений переменных:
typedef std::map<Token, Token> Map;
// Пара переменная—значение:
typedef std::pair<Token, Token> VarVal;
// Строка символов:
typedef std::string String;

// Является ли токен числом?
inline bool isNumber(Token t) {
    return t == '0' || t == '1';
}

// Является ли токен переменной?
inline bool isVariable(Token t) {
    return (t >= 'A' && t <= 'Z') || (t >= 'a' && t <= 'z');
}

// Является ли токен операцией?
inline bool isOperation(Token t) {
    return (t == '|' || t == '&' || t == '-' || t == '>' || t == '~');
}

// Является ли токен открывающей скобкой?
inline bool isOpeningPar(Token t) {
    return t == '(';
}
```

```
// Является ли токен закрывающей скобкой?
inline bool isClosingPar(Token t) {
    return t == ')';
}

// Вернуть величину приоритета операции
// (чем больше число, тем выше приоритет)
inline int priority(Token op) {
    assert (isOperation(op));
    int res = 0;
    switch (op){
        case '-':
            // Отрицание — наивысший приоритет
            res = 5;
            break;
        case '&':
            // Конъюнкция
            res = 4;
            break;
        case '|':
            // Дизъюнкция
            res = 3;
            break;
        case '>':
            // Импликация
            res = 2;
            break;
        case '~':
            // Эквивалентность — наинизший приоритет
            res = 1;
            break;
    }
    return res;
}

// Преобразовать последовательность токенов,
// представляющих выражение в инфиксной записи,
// в последовательность токенов, представляющих
// выражение в обратной польской записи
// (алгоритм Дейкстры «Сортировочная станция»)
Queue infixToPostfix(Queue input);

// Напечатать последовательность токенов
void printSequence(Queue q);

// Является ли символ пробельным?
inline bool isSpace(char c) {
    return c <= ' ';
}

// Если символ — маленькая буква, преобразовать её в большую,
// иначе просто вернуть этот же символ
inline char toUpperCase(char c) {
    if (c >= 'a' && c <= 'z') {
        return c - 'a' + 'A';
    } else {
        return c;
```

```
    }
}

// Преобразовать строку с выражением в последовательность токенов
// (лексический анализатор)
Queue stringToSequence(const String &s);

// Напечатать сообщение об ошибке
inline void printErrorMessage(const String &err) {
    std::cerr << "*** ОШИБКА! " << err << std::endl;
}

// Ввести выражение с клавиатуры
inline String inputExpr() {
    String expr;
    std::cout << "Формула логики высказываний: ";
    std::getline(std::cin, expr);
    return expr;
}

// Выделить из последовательности токенов переменные
Set getVariables(Queue s);

// Получить значения переменных с клавиатуры
Map inputVarValues(const Set &var);

// Заменить переменные их значениями
Queue substValues(Queue expr, Map &varVal);

// Является ли операция бинарной?
inline bool isBinOp(Token t) {
    return t == '&' || t == '|' || t == '>' || t == '~';
}

// Является ли операция унарной?
inline bool isUnarOp(Token t) {
    return t == '-';
}

// Получить bool-значение токена-числа (true или false)
inline bool logicVal(Token x) {
    assert (isNumber(x));
    return x == '1';
}

// Преобразовать bool-значение в токен-число
inline Token boolToToken(bool x) {
    if (x) {
        return '1';
    } else {
        return '0';
    }
}

// Вычислить результат бинарной операции
inline Token evalBinOp(Token a, Token op, Token b) {
    assert (isNumber(a) && isBinOp(op) && isNumber(b));
```

```
    bool res;
    // Получить bool-значения операндов
    bool left = logicVal(a);
    bool right = logicVal(b);
    switch (op){
        case '&':
            // Конъюнкция
            res = left && right;
            break;
        case '|':
            // Дизъюнкция
            res = left || right;
            break;
        case '>':
            // Импликация
            res = !left || right;
            break;
        case '~':
            // Эквивалентность
            res = (!left || right) && (!right || left);
            break;
    }
    return boolToToken(res);
}

// Вычислить результат унарной операции
inline Token evalUnarOp(Token op, Token a) {
    assert (isUnarOp(op) && isNumber(a));
    bool res = logicVal(a);
    switch (op){
        case '-':
            // Отрицание
            res = !res;
            break;
    }
    return boolToToken(res);
}

// Вычислить значение операции, модифицируя стек.
// Результат помещается в стек
void evalOpUsingStack(Token op, Stack &s);

// Вычислить значение выражения, записанного в обратной польской записи
Token evaluate(Queue expr);

// Вывести результат вычисления на экран
void printResult(Token r);

//выводит на экран все значения вектора a до n-го элемента
void output_array(std::vector<Token> a, int n);

//выводит таблицу истинности для формулы input
void printTruthTable(Queue input);

//определяет, является ли формула в постфиксной записи output общезначимой
// (тождественно истинной) на всех интерпретациях
bool isValid(Queue output);
```

```
//выводит все вектора для формулы в постфиксной записи output,
//состоящие из всех возможных комбинаций значений её переменных,
//при подстановке которых в output, формула принимает истинное значение
void printTrueVectors(Queue output);

#endif //MATHLOGIC
```

Файл main.cpp :

```
#include "mathlogic.h"
#include <windows.h>

Queue infixToPostfix(Queue input) {
    // Выходная последовательность (очередь вывода):
    Queue output;
    // Рабочий стек:
    Stack s;
    // Текущий входной токен:
    Token t;
    while (!input.empty()) {
        // Получить токен из начала входной последовательности
        t = input.front();
        input.pop();
        if (isNumber(t) || isVariable(t)) {
            output.push(t);
        } else if (isOperation(t)) { // Если токен — операция op1, то:
            // Пока на вершине стека присутствует токен-операция op2
            // и у op1 приоритет меньше либо равен приоритету op2, то:
            while (!s.empty() && isOperation(s.top())
                   && priority(t) <= priority(s.top())
                    ) {
                // переложить op2 из стека в выходную очередь
                output.push(s.top());
                s.pop();
            }
            // Положить op1 в стек
            s.push(t);
            // Если токен — открывающая скобка, то:
        } else if (isOpeningPar(t)) {
            // Положить его в стек
            s.push(t);
            // Если токен — закрывающая скобка, то:
        } else if (isClosingPar(t)) {
            // Пока токен на вершине стека не является открывающей скобкой:
            while (!s.empty() && !isOpeningPar(s.top())) {
                // Перекладывать токены-операции из стека
                // в выходную очередь
                assert (isOperation(s.top()));
                output.push(s.top());
                s.pop();
            }
            // Если стек закончился до того,
```

```
            // как был встречен токен-«открывающая скобка», то:
            if (s.empty()) {
                // В выражении пропущена открывающая скобка
                throw String("Пропущена открывающая скобка!");
            } else {
                // Иначе выкинуть открывающую скобку из стека
                // (но не добавлять в очередь вывода)
                s.pop();
            }
        } else {
            // В остальных случаях входная последовательность
            // содержит токен неизвестного типа
            String msg("Неизвестный символ \'");
            msg += t + String("\'!");
            throw msg;
        }
    }
    // Токенов на входе больше нет, но ещё могут остаться токены в стеке.
    // Пока стек не пустой:
    while (!s.empty()) {
        // Если токен на вершине стека — открывающая скобка, то:
        if (isOpeningPar(s.top())) {
            // В выражении присутствует незакрытая скобка
            throw String("Незакрытая скобка!");
        } else {
            // Иначе переложить токен-операцию из стека в выходную очередь
            assert (isOperation(s.top()));
            output.push(s.top());
            s.pop();
        }
    }
    // Конец алгоритма.
    // Выдать полученную последовательность
    return output;
}

void printSequence(Queue q) {
    while (!q.empty()) {
        std::cout << q.front();
        q.pop();
    }
}


Queue stringToSequence(const String &s) {
    Queue res;
    for (size_t i = 0; i < s.size(); ++i) {
        if (!isSpace(s[i])) {
            res.push(toUpperCase(s[i]));
        }
    }
    return res;
}

Set getVariables(Queue s) {
    Set res;
    while (!s.empty()) {
```

```
        if (isVariable(s.front()) && res.count(s.front()) == 0) {
            res.insert(s.front());
        }
        s.pop();
    }
    return res;
}

Map inputVarValues(const Set &var) {
    Token val;
    Map res;
    for (Set::const_iterator i = var.begin(); i != var.end(); ++i) {
        do {
            std::cout << *i << " = ";
            std::cin >> val;
            if (!isNumber(val)) {
                std::cerr << "Введите 0 или 1!" << std::endl;
            }
        } while (!isNumber(val));
        res.insert(VarVal(*i, val));
    }
    return res;
}

Queue substValues(Queue expr, Map &varVal) {
    Queue res;
    while (!expr.empty()) {
        if (isVariable(expr.front())) {
            res.push(varVal[expr.front()]);
        } else {
            res.push(expr.front());
        }
        expr.pop();
    }
    return res;
}

void evalOpUsingStack(Token op, Stack &s) {
    assert (isOperation(op));
    // Если операция бинарная, то:
    if (isBinOp(op)) {
        // В стеке должны быть два операнда
        if (s.size() >= 2) {
            // Если это так, то извлекаем правый операнд-число
            Token b = s.top();
            if (!isNumber(b)) {
                throw String("Неверное выражение!");
            }
            s.pop();
            // Затем извлекаем левый операнд-число
            Token a = s.top();
            if (!isNumber(a)) {
                throw String("Неверное выражение!");
            }
            s.pop();
            // Помещаем в стек результат операции
            s.push(evalBinOp(a, op, b));
```

```
        } else {
            throw String("Неверное выражение!");
        }
        // Иначе операция унарная
    } else if (isUnarOp(op) && !s.empty()) {
        // Извлекаем операнд
        Token a = s.top();
        if (!isNumber(a)) {
            throw String("Неверное выражение!");
        }
        s.pop();
        // Помещаем в стек результат операции
        s.push(evalUnarOp(op, a));
    } else {
        throw String("Неверное выражение!");
    }
}

Token evaluate(Queue expr) {
    // Рабочий стек
    Stack s;
    // Текущий токен
    Token t;
    // Пока входная последовательность содержит токены:
    while (!expr.empty()) {
        // Считать очередной токен
        t = expr.front();
        assert (isNumber(t) || isOperation(t));
        expr.pop();
        // Если это число, то:
        if (isNumber(t)) {
            // Поместить его в стек
            s.push(t);
            // Если это операция, то:
        } else if (isOperation(t)) {
            // Вычислить её, модифицируя стек
            // (результат также помещается в стек)
            evalOpUsingStack(t, s);
        }
    }
    // Результат — единственный элемент в стеке
    if (s.size() == 1) {
        // Вернуть результат
        return s.top();
    } else {
        throw String("Неверное выражение!");
    }
}

void printResult(Token r) {
    assert (isNumber(r));
    std::cout << "Значение выражения: " << r << std::endl;
}

void output_array(std::vector<Token> a, int n) {
    std::cout << " ";
    for (int i = 0; i < n; i++)
```

```
        printf(" %c |", a[i]);
}

void printTruthTableInner(Queue output, Set variables, int i, int n) {
    static std::vector<Token> D(100, '0');
    for (int j = '0'; j <= '1'; j++) {
        D[i] = j;
        if (i == n - 1) {
            output_array(D, n);
            //вычисление значения выражения:
            Map varVal;
            int k = 0; //по значениям D (вектору 1ц и 0й)
            for (Set::const_iterator x = variables.begin(); x !=
variables.end(); ++x) { //по именам переменных
                varVal.insert(VarVal(*x, D[k]));
                k++;
            }
            Queue rpn = substValues(output, varVal);

            Token res = evaluate(rpn);
            printf(" %c\n", res);
        } else
            printTruthTableInner(output, variables, i + 1, n);
    }
}

void printTruthTable(Queue input) {
    //получение множества всех переменных из исходной формулы
    Set variables = getVariables(input);
    for (auto var: variables)
        printf(" %c |", var);
    printSequence(input);
    std::cout << std::endl;
    //преобразование послед-ти токенов в постфиксную запись
    Queue output = infixToPostfix(input);
    printTruthTableInner(output, variables, 0, variables.size());
}

void output_vector(std::vector<Token> a, int n) {
    printf("{");
    for (int i = 0; i < n; i++)
        printf("%c, ", a[i]);
    printf("\b\b}\n");
}

bool isValidInner(Queue output, Set variables, int i, int n) {
    static std::vector<Token> D(100, '0');
    static int n_truth = 0;
    for (int j = '0'; j <= '1'; j++) {
        D[i] = j;
        if (i == n - 1) {
            //вычисление значения выражения:
            Map varVal;
            int k = 0; //по значениям D (вектору 1ц и 0й)
            for (Set::const_iterator x = variables.begin(); x !=
variables.end(); ++x, ++k) {
                varVal.insert(VarVal(*x, D[k]));
```

```
            }
            Queue rpn = substValues(output, varVal);
            Token res = evaluate(rpn);
            if (res == '1')
                n_truth++;
        } else
            isValidInner(output, variables, i + 1, n);
    }
    return n_truth == (1 << n); //n_truth == 2^n
}

bool isValid(Queue output) {
    Set variables = getVariables(output);
    return isValidInner(output, variables, 0, variables.size());
}

void printTrueVectorsInner(Queue output, Set variables, int i, int n) {
    static std::vector<Token> D(100, '0');
    for (int j = '0'; j <= '1'; j++) {
        D[i] = j;
        if (i == n - 1) {
            //вычисление значения выражения:
            Map varVal;
            int k = 0; //по значениям D (вектору 1ц и 0й)
            for (Set::const_iterator x = variables.begin(); x !=
variables.end(); ++x, ++k)
                varVal.insert(VarVal(*x, D[k]));

            Queue rpn = substValues(output, varVal);
            Token res = evaluate(rpn);
            if (res == '1')
                output_vector(D, n);
        } else
            printTrueVectorsInner(output, variables, i + 1, n);
    }
}

void printTrueVectors(Queue output) {
    Set variables = getVariables(output);
    printTrueVectorsInner(output, variables, 0, variables.size());
}

int main() {
    SetConsoleOutputCP(CP_UTF8);//Подключение русского языка
    std::string expr = inputExpr();
    Queue input = stringToSequence(expr);
    try{
        printTruthTable(input);
        Queue output = infixToPostfix(input);
        if (isValid(output)) {
            std::cout << "Формула действительна,т.к. истинна на всех
интерпретациях.";
        } else {
            std::cout << "Формула недействительна,т.к. истинна не во всех
интерпретациях.\n";
        }
    } catch (const String &err){
```

```
            printErrorMessage(err);
            exit(1);
        }
        return 0;
}
```