

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 № 2**

по дисциплине: Математическая логика и теория алгоритмов

тема: «Логика предикатов»

Выполнил: ст. группы ПВ-202

Аладиб Язан

Проверил:

Куценко Дмитрий Александрович

Бондаренко Татьяна Владимировна

Белгород 2022 г.

## Теоретическая часть :

### Задание варианта №2:

Номер варианта	Номера задач				
2	18.1	6.3	10.14	24.2	45

1)

18. Дана следующая интерпретация:  
 $D = \{1, 2\}$ ;  $a = 1$ ,  $b = 1$ ;  $f(1) = 2$ ,  $f(2) = 1$ ;  
 $P(1, 1) = P(1, 2) = 1$ ,  $P(2, 1) = P(2, 2) = 0$ .

Определите, являются ли следующие формулы  
общезначимыми, противоречивыми или  
выполнимыми в указанной интерпретации.

1)  $P(a, f(a)) \& P(b, f(b))$ .

Решение:

при  $a = 1, b = 1$ :  $P(1, f(1)) \& P(1, f(1)) \equiv 1 \rightarrow P(2, 2) \equiv 1 \rightarrow 0 \equiv 0$ .  
при  $a = 1, b = 2$ :  $P(1, f(1)) \& P(2, f(2)) \equiv 1 \rightarrow P(2, 1) \equiv 1 \rightarrow 0 \equiv 0$ .  
при  $a = 2, b = 1$ :  $P(2, f(2)) \& P(1, f(1)) \equiv 0 \rightarrow P(1, 2) \equiv 0 \rightarrow 1 \equiv 1$ .  
при  $a = 2, b = 2$ :  $P(2, f(2)) \& P(2, f(2)) \equiv 0 \rightarrow P(1, 1) \equiv 0 \rightarrow 1 \equiv 1$ .

Итак, данной интерпретации формула может  
принимать значение как 0, так и 1, поэтому формула  
является выполнимой на данной интерпретации.

2)

6. определить истинность, отрицательность или выполнимость следующих формул:

3)  $P(a) \rightarrow \exists x P(x)$

решение:

$$\begin{aligned} P(a) &\rightarrow \exists x P(x) \\ P(a) &\rightarrow \forall x \overline{P(x)} \\ \overline{P(a)} \vee \forall x \overline{P(x)} \\ \forall x (\overline{P(a)} \vee \overline{P(x)}) \end{aligned}$$

применим замену  $\langle x/a \rangle$

$$\begin{aligned} \overline{P(a)} \vee \overline{P(a)} \\ \overline{P(a)} \end{aligned}$$

$\overline{P(a)} \neq 1$  и  $\overline{P(a)} \neq 0$ , значит формула не является истинной. мы получили значение отрицания  $\Rightarrow P(a) \rightarrow \exists x P(x)$  является истинным.

3)

10. проверить, являются ли следующие формулы тождественно истинными:

14)  $\exists x (P(x) \vee Q(x)) \Leftrightarrow \exists x P(x) \vee \exists x Q(x)$

решение:

$\exists x (P(x) \vee Q(x))$  истинно, то существует значение  $x$  такое, что  $P(x) \vee Q(x)$  истинно. таким образом,  $P(x)$  истинно или  $Q(x)$  истинно.

Когда  $\exists x P(x)$  истинно или  $\exists x Q(x)$  истинно, что означает, что  $\exists x P(x) \vee \exists x Q(x)$  истинно.

$\exists x (P(x) \vee Q(x))$  ложно, тогда для всех значений  $x$  имеем  $P(x) \vee Q(x)$  ложно, следовательно,  $P(x)$  ложно и  $Q(x)$  ложно. тогда  $\exists x P(x)$  ложно и  $\exists x Q(x)$  ложно, что означает, что  $\exists x P(x) \vee \exists x Q(x)$  ложно.

вывод: эти формулы, где выражения всегда имеют одно и то же значение истинности и потому они логически эквивалентны.

4)

24. привести к предикатной нормальной форме:

$$2) \exists z \forall x (G(x, y) \& F(x, z)) \rightarrow \forall y H(y, z)$$

решение:

$$\exists z \forall x (G(x, y) \& F(x, z)) \rightarrow \forall y H(y, z)$$

$$\forall z \forall x (G(x, y) \& F(x, z)) \rightarrow \forall y H(y, z)$$

$$\forall z \forall x (G(x, y) \& F(x, z)) \rightarrow \forall y H(y, z)$$

$$\forall z \forall x (G(x, y) \& F(x, z)) \rightarrow \forall y H(y, z)$$

$$\forall z \forall x (G(x, y) \& F(x, z)) \rightarrow \forall y H(y, z)$$

5)

45. Все студенты нашей группы - болельщики «Спартака», причем некоторые занимаются спортом. Следовательно, некоторые из болельщиков «Спартака» занимаются спортом. Доказать с помощью метода резолюций.

решение:

используем символы одноместных предикатов  $F, G$  и  $S$  интерпретируемых следующим образом:

$F(x)$ : « $x$  - болельщик «Спартака»»,  
 $G(x)$ : « $x$  - студент нашей группы»,  
 $S(x)$ : « $x$  - спортсмен».

тогда:  $F = (\forall x) (G(x) \rightarrow F(x)) \& (\exists y) (G(y) \& S(y))$ ,  
 $G = (\exists x) (F(x) \& S(x))$ . составим множество формул  $\{F, \bar{G}\}$  и каждую из них приведем к скановской нормальной форме. получим формулы:

$$H_1 = (\forall x) (G(x) \vee F(x)) \& G(a) \& S(a)$$

$$H_2 = (\forall x) (F(x) \vee S(x)).$$

множество дизъюнктов  $S$  равно  $\{G(x) \vee F(x) \vee S(x), G(a) \vee S(a)\}$ . исключаем дизъюнкт из множества  $S$  вводящий алгебраический образ.

## Практическая часть :

1) Разработать программу, способную считывать несколько формул-посылок логики высказываний и выводить на экран все формулы-следствия из этих посылок.

### 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>

using namespace std;

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;
}

//ставит в соответствие каждой переменной из var значение из a
Map inputByArray(const Set &var, int *a) {
    Token val;

```

```

Map res;
for (Set::const_iterator i = var.begin(); i != var.end(); ++i) {
    val = (*a ? '1' : '0');
    a++;

    res.insert(VarVal(*i, val));
}
return res;
}

//записывает в a таблицу истинности для выражения input
//с набором переменных vars
void getTruthTable(Queue &input, Set &vars, std::vector<std::vector<int>> &a)
{
    static int D[100] = {0};
    static int i = 0;
    static int z = 0; //номер столбца в таблице истинности
    for (int x = 0; x <= 1; x++) {
        D[i] = x;
        if (i == vars.size() - 1) {
            //вычисление значения выражения:
            Map Vars = inputByArray(vars, D); //каждой переменной
соответствует свое значение из D
            Token res = evaluate(substValues(input, Vars)); //вычисление
значения для строки таблицы
            for (int k = 0; k <= i; k++)
                a[z][k] = D[k];
            a[z][vars.size()] = res == '1';
            z++;
        } else {
            i++;
            getTruthTable(input, vars, a);
            i--;
        }
    }
    if (i == 0) {
        z = 0;
    }
}

//вывод СКНФ по заданному набору дизъюнктов и всех ее сочетаний
void writeExpr(std::vector<String> &d) {
    static int D[100]; //формирование сочетания
    static int i = 0;
    for (int x = 0; x <= 1; x++) {
        D[i] = x;
        if (i == d.size() - 1) {
            int end = 1;
            for (int v = 0; v < d.size(); v++) { //по всем элементам из
набора дизъюнктов
                if (D[v]) {
                    if (!end) { //если не дошли до последнего дизъюнкта, то
ВЫВОДИМ
                        std::cout << "&" << d[v];
                    } else { //если дошли до конца -- выводим последний
дизъюнкт и утверждаем, что это конец
                        end = 0;

```

```

        std::cout << d[v];
    }
}
}
if (!end) //если есть еще дизъюнкты
    std::cout << "\n";
} else {
    i++;
    writeExpr(d);
    i--;
}
}
}

//вывод всех формул-следствий для всех формул-посылок, наход. в s
void OutputConsequences(String s) {
    Queue input = stringToSequence(s);
    Queue output = infixToPostfix(input); //преобразовать последовательность
    токенов в ОПЗ
    Set vars = getVariables(output); //множество переменных
    int table_size = 1 << vars.size();
    std::vector<std::vector<int>> a(table_size); //матрица = таблица
    истинности
    for (int i = 0; i < table_size; i++)
        a[i].resize(vars.size() + 1);
    getTruthTable(output, vars, a); //в a записывается таблица истинности для
    выражения output
    std::vector<String> result;
    for (int i = 0; i < a.size(); i++) { //по таблице истинности для
    выражения СКНФ
        if (a[i][vars.size()] == 0) { //формирование дизъюнкта по
        соответствующей строке таблицы
            String d = "(";
            if (a[i][0] == 1)
                d = d + "-";
            d = d + *(vars.begin()); //значение первого элемента в множестве
            элементов выражения
            auto ptr_curr = vars.begin(); //указатель на элементы из
            множества переменных
            for (int j = 1; j < a[i].size() - 1; j++) {
                d = d + "|";
                if (a[i][j] == 1)
                    d = d + "-";

                d = d + *(++ptr_curr);
            }
            d = d + ")";
            result.push_back(d);
        }
    }
    writeExpr(result);
}

int main() {
    SetConsoleOutputCP(CP_UTF8); //Подключение русского языка
    std::cout << "Введите количество формул-посылок:"; //считывание кол-ва
    посылок

```

```

    int n;
    std::cin >> n;
    String res = "(";
    String p;
    std::cout << "Введите формулы-посылки:";
    std::cin >> p; //считывание первой посылки
    res = res + p + ")";
    for (int i = 1; i < n; i++) { //считывание оставшихся посылок
        String s;
        std::cin >> s;
        res = res + "&" + "(" + s + ")";
    }
    std::cout << "Все формулы-следствия :\n"; //вывод формул-следствий
    OutputConsequences(res);
    return 0;
}

```



