#### 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»

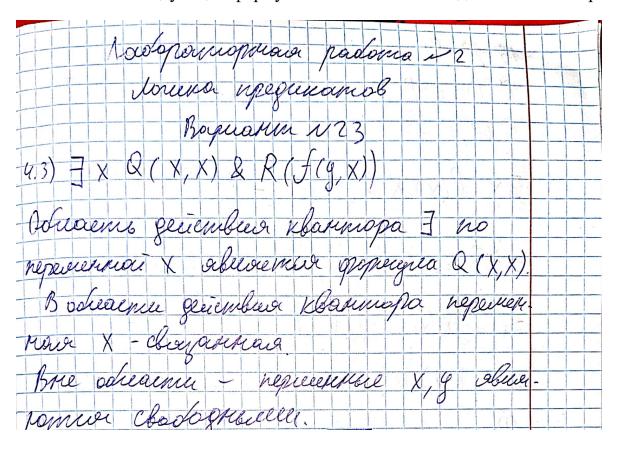
Выполнил: ст. группы ПВ-212 Степанов Степан Николаевич

Проверили: Рязанов Юрий Дмитриевич Осипов Олег Васильевич

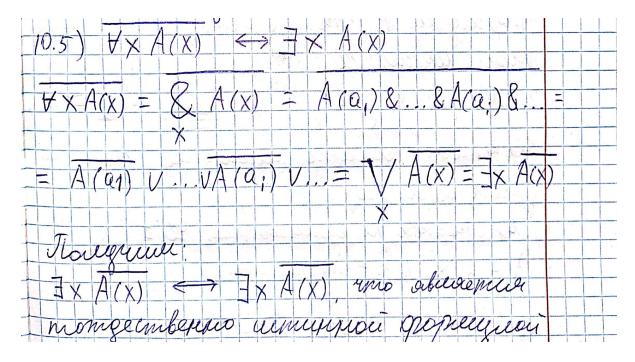
### Вариант 23

**Цель работы:** изучить логику предикатов и закрепить навык решения теоретических и практических задач.

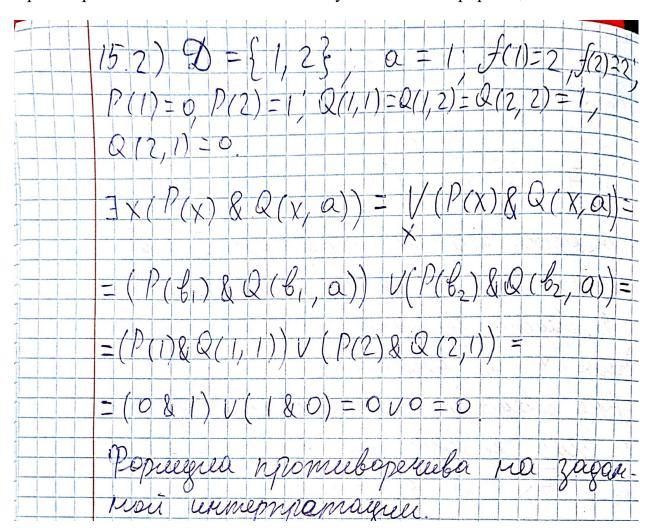
4. Определить, какие вхождения переменных являются свободными, а какие связанными в следующих формулах. Указать области действия кванторов.



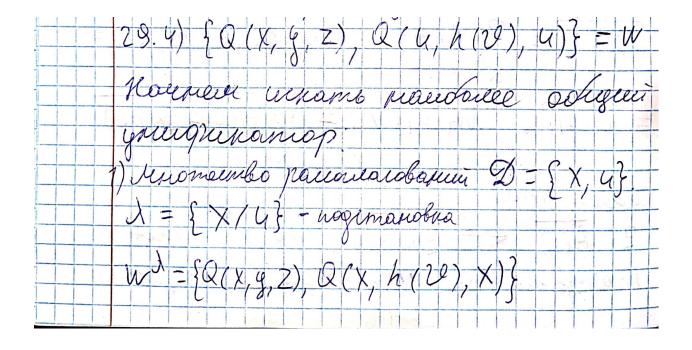
10. Проверить, являются ли следующие формулы тождественно истинными:

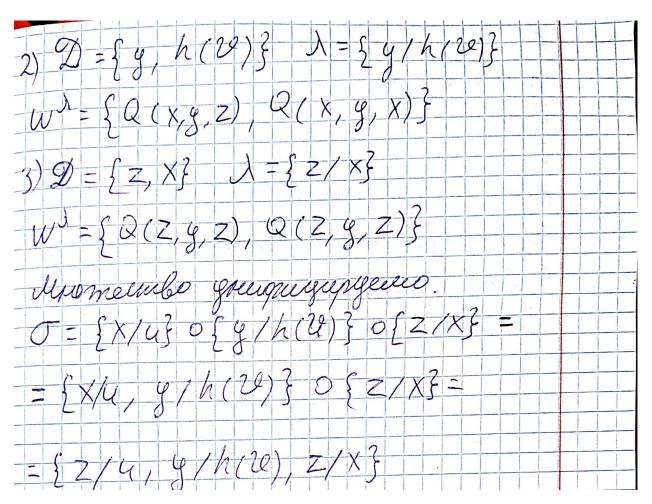


15. Определить, являются ли следующие формулы общезначимыми, противоречивыми или выполнимыми в указанной интерпретации.

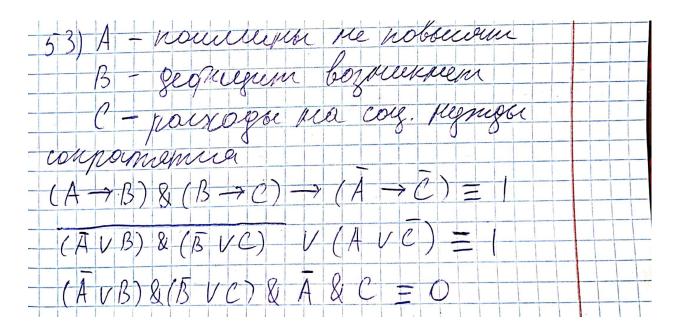


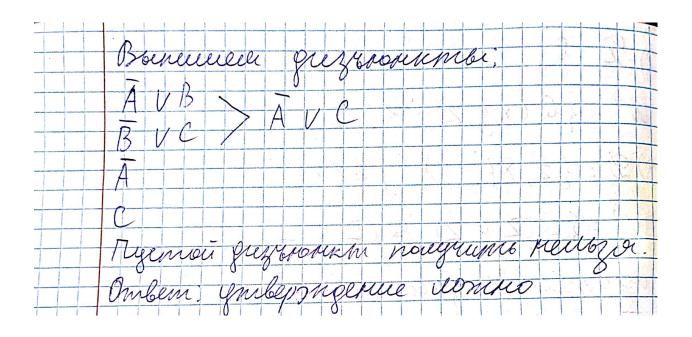
29. Определить, унифицируемо ли каждое из следующих множеств. Если да, то получить наиболее общий унификатор.





53. В бюджете возникнет дефицит, если не повысят пошлины. Если в бюджете возникнет дефицит, то расходы на социальные нужды сократятся. Следовательно, если повысят пошлины, то расходы на социальные нужды не сократятся. Проверить это с помощью метода резолюций.





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

Для решения используется алгоритм получения ОПЗ и другие функции из предыдущей работы.

```
// Возвращает множество дизъюнкций, состоящее из переменных vars,
// согласно таблице истинности table
std::set<Queue> getDisjunctionsFromTable(const TruthTable &table,
                                             const Set &vars) {
    std::set<Queue> res;
    for (const auto &row: table) {
        Queue values = row.first;
        auto var = vars.begin();
        Queue currExpr;
        while (!values.empty()) {
            Token token = values.front();
            if (token == '1')
                currExpr.push('!');
            currExpr.push(*var);
            if (std::next(var) != vars.end())
                currExpr.push('|');
            var++;
            values.pop();
        res.insert(currExpr);
    }
    return res;
}
```

```
// Выводит на экран все комбинации дизъюнкций для заданного СКНФ,
// т.е. формирует и выводит все следствия
void printConsectaries(const std::vector<std::set<Queue>>
&conjunctionSubsets) {
    for (const auto &subset: conjunctionSubsets) {
        for (auto it = subset.begin(); it != subset.end(); ++it) {
            auto expr = *it;
            std::cout << '(';
            while (!expr.empty()) {
                std::cout << expr.front();</pre>
                expr.pop();
            }
            std::cout << ')':
            if (std::next(it) != subset.end())
                std::cout << '&';
        }
        std::cout << '\n';
    }
}
// Получает сочетания для множества в пределах [currElement, end)
// размера subsetSize и записывает их в subsets
template<typename T, typename Iter>
void getCombinations(Iter currElement, Iter end, const size_t subsetSize,
                     std::set<T> &currSubset,
                     std::vector<std::set<T>> &subsets) {
    while (currElement != end) {
        currSubset.insert(*currElement);
        if (currSubset.size() == subsetSize)
            subsets.push_back(currSubset);
        else {
            getCombinations(std::next(currElement), end, subsetSize,
currSubset, subsets);
        }
        currSubset.erase(*currElement);
        currElement++:
    }
}
// Возвращает массив подмножеств множества set
template<typename T>
std::vector<std::set<T>> getSubsets(std::set<T> &set) {
    std::vector<std::set<T>> subsets(1);
    std::set<T> currSubset;
    for (int i = 1; i <= set.size(); i++)</pre>
        getCombinations(set.begin(), set.end(), i, currSubset, subsets);
   return subsets:
}
```

```
int main() {
    SetConsoleOutputCP(CP_UTF8);
    // Ввод посылок
    std::cout << "Введите количество посылок: ";
    size_t n;
    std::cin >> n;
    getchar();
    std::cout << "Введите посылки: ";
    std::string expr;
    for (int i = \emptyset; i < n; ++i) {
        auto currExpr = "(" + inputExpr() + ")";
        expr += currExpr;
        if (i + 1 != n)
            expr += '&';
    }
    Queue input = stringToSequence(expr);
    try {
        // Преобразование в ОПЗ
        Queue output = infixToPostfix(input);
        printSequence(output);
        auto vars = getVariables(output);
        // Получим строки таблицы, необходимые для построения СКНФ
        auto table = deleteRowsIf(getTruthTable(output, vars),
                                   [](const auto &pair) {
                                       return pair.second == '1';
                                                        });
        // Преобразуем строки таблицы в дизъюнкции
        auto disjunctions = getDisjunctionsFromTable(table, vars);
        // Получим все подмножества множества дизъюнкций
        auto disjunctionsSubsets = getSubsets(disjunctions);
        std::cout << "Следствия из nocылoк:";
        printConsectaries(disjunctionsSubsets);
    } catch (const String &err) {
        printErrorMessage(err);
        exit(1);
    }
   return Ø;
```

```
Введите количество посылок: 2
Введите посылки: X> (Y|Z)

Z>Y

XYZ|>ZY>&

Следствия из посылок:
(!X|Y|!Z)
(!X|Y|Z)
(!X|Y|Z)
(!X|Y|Z)
(!X|Y|Z)
(!X|Y|Z)&(!X|Y|Z)
(!X|Y|Z)&(X|Y|Z)
(!X|Y|Z)&(X|Y|Z)
(!X|Y|Z)&(X|Y|Z)
```

**Вывод:** в ходе работы была изучена логика предикатов и закреплены навыки решения теоретических и практических задач.