

Министерство образования и науки РФ  
Пермский национальный исследовательский политехнический университет  
Электротехнический факультет  
Кафедра информационных технологий и автоматизированных систем

Дискретная математика и математическая логика  
Лабораторная работа № 4

Тема: «Классы булевых функций. Полнота системы булевых функций»

Выполнил: студент группы  
ИВТ-23-1Б  
Долганова Диана Евгеньевна  
Проверил: ст. пр.  
Рустамханова Г. И.

## Оглавление

Цель работы	1
Задачи работы	2
Этапы выполнения	3
1. Проверка на булеву функцию	4
2. Функции, сохраняющие 0	4
3. Функции, сохраняющие 1	5
4. Самодвойственные функции	5
5. Монотонные функции	5
6. Линейные функции	6
7. Полнота системы функций	7
8. Таблица классов булевых функций	8
9. Меню	9
10. Главная функция	9
Тестирование программы	10
Заключение	12

## **Цель работы**

Разработать программу, которая будет выводить таблицу классов булевых функций и проверять систему функций на полноту.

## **Задачи работы**

Реализовать следующий функционал:

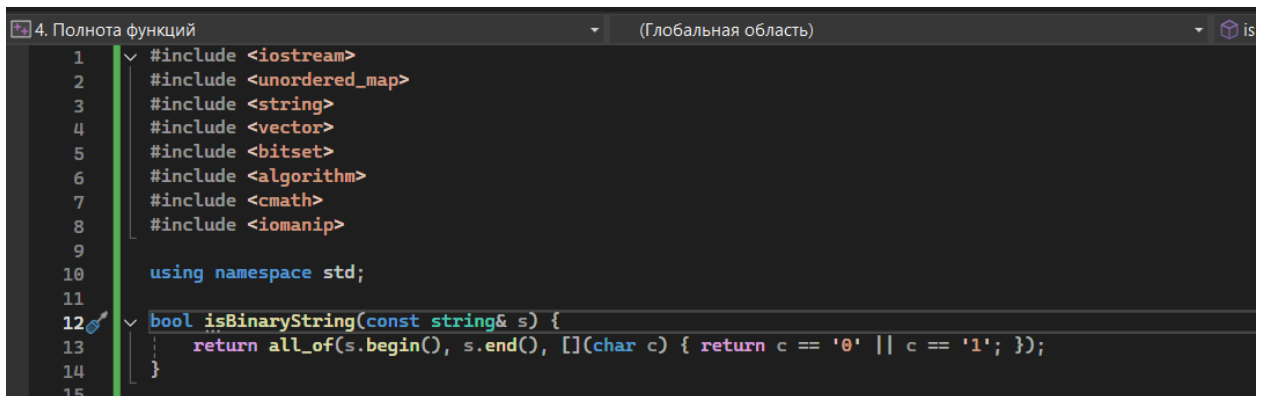
1. Возможность задать несколько функций на вход
2. Длина вектора может быть разной (2, 4, 8 и т. д.)
3. Вывести на экран таблицу классов
4. Определить полная функция или нет

## Этапы выполнения

Программа была разработана на языке C++ в программной среде Microsoft Visual Studio 2022.

### 1. Проверка на булеву функцию

Функция `isBinaryString` проверяет, является ли строка двоичной (содержит только 0 и 1). Используется `all_of` с лямбда-функцией для проверки каждого символа (рис. 1).

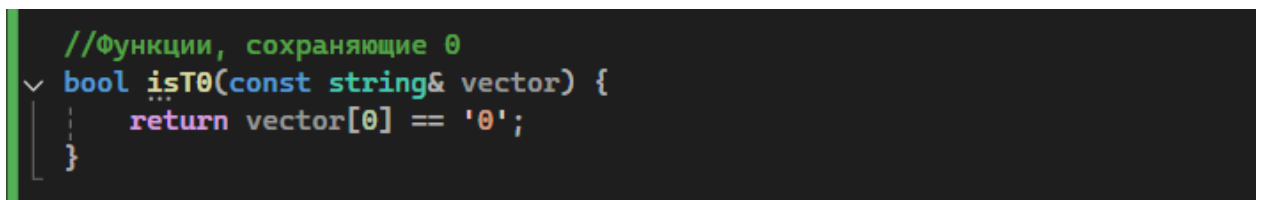


```
4. Полнота функций (Глобальная область)
1  #include <iostream>
2  #include <unordered_map>
3  #include <string>
4  #include <vector>
5  #include <bitset>
6  #include <algorithm>
7  #include <cmath>
8  #include <iomanip>
9
10 using namespace std;
11
12 bool isBinaryString(const string& s) {
13     return all_of(s.begin(), s.end(), [](char c) { return c == '0' || c == '1'; });
14 }
15
```

Рисунок 1 - функция `isBinaryString`

### 2. Функции, сохраняющие 0

Булева функция сохраняет константу 0 (принадлежит классу  $T^0$ ), если на наборе из всех нулей функция принимает значение ноль. Функция `isSave0` проверяет, сохраняет ли функция 0. Она возвращает `true`, если первый элемент (в 0-м индексе) строки (вектора) равен '0'. (рис. 2).



```
//Функции, сохраняющие 0
bool isT0(const string& vector) {
    return vector[0] == '0';
}
```

Рисунок 2 - функция `isSave0`

### 3. Функции, сохраняющие 1

Булева функция сохраняет константу 1 (принадлежит классу  $T^1$ ), если на наборе из всех единиц функция принимает значение единица. Функция `isSave1` проверяет, сохраняет ли функция 1. Возвращает `true`, если последний элемент равен '1' (рис. 3).

```
//Функции, сохраняющие 1
bool isT1(const string& vector) {
    return !vector.empty() && vector.back() == '1';
}
```

Рисунок 2 - Функция `isSave1`

### 4. Самодвойственные функции

Булева функция  $f(x_1, \dots, x_n)$  самодвойственна (принадлежит классу  $S$ ), если она равна двойственной себе функции, то есть:

$$f(x_1, \dots, x_n) = f^*(x_1, \dots, x_n) = f(\bar{x}_1, \dots, \bar{x}_n).$$

Функция `isS` проверяет, является ли функция самодвойственной. Сравниваются элементы с концов к центру. Если найдено совпадение, возвращает `false`, иначе `true`. (рис. 4).

```
//Самодвойственные функции
bool isS(const string& vector) {
    for (size_t i = 0; i < vector.size() / 2; i++) {
        if (vector[i] == vector[vector.size() - 1 - i]) {
            return false; // Если пара равна, не самодвойственный
        }
    }
    return true; // Если все пары прошли проверку, функция самодвойственная
}
```

Рисунок 4 - Функция `isS`

### 5. Монотонные функции

Булева функция  $f(x_1, \dots, x_n)$  называется монотонной (принадлежит классу  $M$ ), если для любой пары наборов  $\alpha$  и  $\beta$  таких, что  $\alpha \preceq \beta$ , выполняется условие  $f(\alpha) \leq f(\beta)$ . Функция `isM` проверяет, является ли функция монотонной.

Если для некоторых  $i$  и  $j$  флаг  $i \& j$  совпадает с  $i$  и  $\text{vector}[i]$  больше  $\text{vector}[j]$ , функция перестает быть монотонной.(рис. 5).

```
//Монотонные функции
bool isM(const string& vector) {
    // Проверка на монотонность
    for (size_t i = 0; i < vector.size(); ++i) {
        for (size_t j = 0; j < vector.size(); ++j) {
            if ((i & j) == i) {
                if (vector[i] > vector[j]) {
                    return false; // Если f(i) > f(j) при i < j, не монотонная
                }
            }
        }
    }
    return true; // Все проверки пройдены, функция монотонная
}
```

Рисунок 5 - Функция isM

## 6. Линейные функции

Булева функция называется *линейной* (принадлежит классу  $L$ ), если ее полином Жегалкина линеен. Функция isL проверяет, является ли функция линейной. Определяет количество переменных на основе длины строки. Для разных случаев (количество переменных от 2 до 4) проводятся проверки линейности (рис. 6)

```

//Линейные функции
bool isL(const string& vector) {
    size_t length = vector.length();
    int numVariables = log2(length);

    if (numVariables < 1 || length != (1 << numVariables)) {
        return false;
    }

    int c0 = vector[0] - '0';

    if (numVariables == 2) {
        int cy = c0 ^ (vector[1] - '0');
        int cx = c0 ^ (vector[2] - '0');
        int cxy = c0 ^ cx ^ cy ^ (vector[3] - '0');
        return cxy == 0; // Логическая функция является линейной
    }

    else if (numVariables == 3) {
        bool cz = c0 ^ (vector[1] - '0');
        bool cy = c0 ^ (vector[2] - '0');
        bool cx = c0 ^ (vector[4] - '0');
        bool cxy = c0 ^ cx ^ cy ^ (vector[6] - '0');
        bool cxz = c0 ^ cx ^ cz ^ (vector[5] - '0');
        bool czy = c0 ^ cz ^ cy ^ (vector[3] - '0');
        bool cxyz = c0 ^ cx ^ cy ^ cz ^ (vector[7] - '0');
        return (cxy == 0 && cxz == 0 && czy == 0 && cxyz == 0);
    }

    else if (numVariables == 4) {
        bool cx = c0 ^ (vector[1] - '0');
        bool cy = c0 ^ (vector[2] - '0');
        bool cz = c0 ^ (vector[4] - '0');
        bool cw = c0 ^ (vector[8] - '0');
        bool cxz = c0 ^ cx ^ cz ^ (vector[5] - '0');
        bool cxy = c0 ^ cx ^ cy ^ (vector[6] - '0');
        bool czy = c0 ^ cz ^ cy ^ (vector[3] - '0');
        bool cxyz = c0 ^ cx ^ cy ^ cz ^ (vector[7] - '0');
        bool cwz = c0 ^ cw ^ cz ^ (vector[9] - '0');
        bool cxwz = c0 ^ cw ^ cx ^ cz ^ (vector[10] - '0');
        return (cxy == 0 && cxz == 0 && czy == 0 && cxyz == 0 && cwz == 0 && cxwz == 0);
    }

    return false; // Для других случаев, не определённых выше, возвращаем false
}

```

Рисунок 6 - Функция isL

## 7. Полнота системы функций

Для того, чтобы система булевых функций  $N$  была функционально полной, необходимо и достаточно, чтобы она не содержалась целиком ни в одном из пяти замкнутых классов  $T^0$ ,  $T^1$ ,  $L$ ,  $S$  и  $M$ , то есть чтобы система булевых функций  $N$  содержала хотя бы одну функцию, не сохраняющую константу 0, хотя бы одну функцию, не сохраняющую константу 1, хотя бы одну нелинейную, хотя бы одну несамодвойственную и хотя бы одну немонотонную функции (теорема Поста). Функция isComplete проверяет полноту набора функций. Если в каждом столбце таблицы классов есть хотя бы 1 минус, то система функций полная. Функция возвращает true, если все классы содержат хотя бы одну функцию (рис. 7).



```

//Полнота системы функций
bool isComplete(const unordered_map<string, vector<int>>& vectors) {
    bool hasT0 = false, hasT1 = false, hasS = false, hasM = false, hasL = false;

    // Пройдем по всем векторам и проверим, есть ли хотя бы один '-' в каждом классе
    for (const auto& pair : vectors) {
        const string& vector = pair.first;
        if (!isT0(vector)) hasT0 = true; // Если в классе T0 есть '-', значит условие выполнено
        if (!isT1(vector)) hasT1 = true; // Если в классе T1 есть '-', значит условие выполнено
        if (!isS(vector)) hasS = true; // Если в классе S есть '-', значит условие выполнено
        if (!isM(vector)) hasM = true; // Если в классе M есть '-', значит условие выполнено
        if (!isL(vector)) hasL = true; // Если в классе L есть '-', значит условие выполнено
    }

    // Если в каждом классе есть хотя бы один '-', система считается полной
    return hasT0 && hasT1 && hasS && hasM && hasL;
}

```

Рисунок 7- Функция isComplete

## 8. Таблица классов булевых функций

Функция printTable выводит таблицу, показывающую принадлежность каждого вектора к классам булевых функций. Использует iomanip для форматирования вывода. После этого проверяется полнота системы с помощью isComplete. (рис. 8).

```

//Таблица классов булевых функций
void printTable(const unordered_map<string, vector<int>>& vectors) {
    cout << "\nТаблица принадлежности к классам булевых функций:\n";
    cout << setw(10) << "Вектор" << " | "
        << setw(2) << "T0" << " | "
        << setw(2) << "T1" << " | "
        << setw(2) << "S" << " | "
        << setw(2) << "M" << " | "
        << setw(2) << "L" << "\n";
    cout << string(45, '-') << "\n";

    for (const auto& pair : vectors) {
        const string& vector = pair.first;
        cout << setw(10) << vector << " | "
            << setw(2) << (isT0(vector) ? "+" : "-") << " | "
            << setw(2) << (isT1(vector) ? "+" : "-") << " | "
            << setw(2) << (isS(vector) ? "+" : "-") << " | "
            << setw(2) << (isM(vector) ? "+" : "-") << " | "
            << setw(2) << (isL(vector) ? "+" : "-") << "\n";
    }

    // Проверяем полноту после вывода таблицы
    if (isComplete(vectors)) {
        cout << "Система функций является полной.\n";
    }
    else {
        cout << "Система функций не является полной.\n";
    }
}

```

Рисунок 8 - Функция printTable

## 9. Меню

Функция `menu` запрашивает у пользователя количество булевых выражений (векторов) и затем по одному вводит их. Если введенный вектор соответствует двоичному формату и отсутствует в словаре, он добавляется. В противном случае выводит сообщение об ошибке (рис. 9).

```
void menu(unordered_map<string, vector<int>>& vectors) {
    int expressionNumber;
    cout << "Введите количество выражений: ";
    cin >> expressionNumber;
    cin.ignore(); // для очищения буфера ввода после ввода числа

    for (int i = 0; i < expressionNumber; ++i) {
        string vector;
        cout << "Введите вектор: ";
        getline(cin, vector);

        if (isBinaryString(vector) && vectors.find(vector) == vectors.end()) {
            vectors[vector] = {}; // добавляем вектор в словарь
        }
        else {
            cout << "Неверный формат ввода вектора\n";
            return; // выходим из функции на ошибке
        }
    }
}
```

Рисунок 9 - Функция `menu`

## 10. Главная функция

Основная функция `main` создает ассоциативный массив `vectors`, вызывает функцию `menu` для ввода данных, а затем выводит результат через `printTable` (рис. 10).

```
int main() {
    setlocale(LC_ALL, "RU");
    unordered_map<string, vector<int>> vectors;
    menu(vectors);
    printTable(vectors);

    return 0;
}
```

Рисунок 10 - Функция `main`

## Тестирование программы

Для проверки программы введем векторы и проверим таблицу классов и полноту системы на корректность (рис. 11 - 13).

```
cs Консоль отладки Microsoft Visual Studio
Введите количество выражений: 2
Введите вектор: 0111
Введите вектор: 1101

Таблица принадлежности к классам булевых функций:
Вектор | T0 | T1 | S | M | L
-----
0111 | + | + | - | + | -
1101 | - | + | - | - | -
Система функций не является полной.
```

Рисунок 11 - Первая система функций

```
cs Консоль отладки Microsoft Visual Studio
Введите количество выражений: 3
Введите вектор: 11110111
Введите вектор: 01100101
Введите вектор: 10001110

Таблица принадлежности к классам булевых функций:
Вектор | T0 | T1 | S | M | L
-----
11110111 | - | + | - | - | -
10001110 | - | - | + | - | -
01100101 | + | + | - | - | -
Система функций является полной.
```

Рисунок 12 - Вторая система функций

Введите количество выражений: 3

Введите вектор: 0001

Введите вектор: 0000

Введите вектор: 1001

Таблица принадлежности к классам булевых функций:

Вектор	T0	T1	S	M	L
--------	----	----	---	---	---

0001	+	+	-	+	-
------	---	---	---	---	---

0000	+	-	-	+	+
------	---	---	---	---	---

1001	-	+	-	-	+
------	---	---	---	---	---

Система функций является полной.

Рисунок 13 - Третья система функций

## **Заключение**

В ходе работы была разработана программа, которая позволяет ввести на вход несколько функций с разной длиной вектора, выводит таблицу с принадлежностью функций к замкнутым классам и проверяет систему функций на полноту.

## **Список используемой литературы**

1. 16. Функциональная полнота системы булевых функций URL:  
[https://ido.tsu.ru/iop\\_res/bulevfunc/text/g16\\_1.html](https://ido.tsu.ru/iop_res/bulevfunc/text/g16_1.html) (дата обращения:  
10.12.2024).
2. 15. Важнейшие замкнутые классы булевых функций URL:  
[https://ido.tsu.ru/iop\\_res/bulevfunc/text/g15\\_3\\_3.html](https://ido.tsu.ru/iop_res/bulevfunc/text/g15_3_3.html) (дата обращения:  
10.12.2024).