Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования «Санкт-Петербургский политехнический университет Петра Великого»

Институт компьютерных наук и кибербезопасности

Высшая школа технологий искусственного интеллекта

Направление: 02.03.01. Математика и компьютерные науки

Дискретная математика
Отчет о выполнении лабораторной работы №2
Исследование двоичной функции. Вариант №5.

Студент Козловская О. А. _____
группы 5130201/30002

Преподаватель Востров А. В. _____

Санкт-Петербург, 2025

Содержание

Введение

В данной лабораторной работе требуется Построить таблицу истинности по исходной булевой функции ($f = 12978_{10}$) 4х элементов и по ней построить дерево решении и бинарную диаграмму решений, а также синтаксическое дерево для полинома Жегалкина, Реализовать программно хранение полученной бинарной диаграммы решений и вычисление по ней значения (по пользовательскому вводу значения переменных). Так же требуется По таблице истинности программно построить СДНФ и СКНФ. Вычислить по СДНФ значение булевой функции (по пользовательскому вводу). Сверить полученные значения с исходной таблицей истинности. Для заданной функции построить программно полином Жегалкина (любым способом). Вывести его на экран и вычислить значение булевой функции согласно пользовательскому вводу.

1 Математическое описание

1.1 Булевые функции

Функции $f : E_2^n \to E_2$, где $E_2 \overset{\text{Def}}{=} \{0, 1\}$, называются *функциями алгебры логики*, или *булевыми функциями* от n переменных. Множество булевых функций от n переменных обозначим P_n, $P_n \overset{\text{Def}}{=} \{f \mid f : E_2^n \to E_2\}$.

Булеву функцию от n переменных можно задать *таблицей истинности*:

x_1	$\cdots$	x_{n-1}	x_n	$f(x_1, \ldots, x_n)$
0	$\cdots$	0	0	$f(0, \ldots, 0)$
0	$\cdots$	0	1	$f(0, \ldots, 0, 1)$
0	$\cdots$	1	0	$f(0, \ldots, 1, 0)$
$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$
1	$\cdots$	1	1	$f(1, \ldots, 1)$

Если число переменных равно n, то в таблице истинности имеется 2^n строк, соответствующих всем различным комбинациям значений переменных.

Таблица истинности для функции
$f(x_1, x_2, x_3, x_4)$ = $(0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0)$

Таблица 1. Таблица истинности для 4 переменных

A	B	C	D	$f(A, B, C, D)$
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

1.2 Совершенные нормальные формы

Реализация булевой функции $f(x_1, \ldots, x_n)$ в виде формулы

$$\bigvee_{\left\{(\sigma_1,\ldots,\sigma_n)\,\middle|\, f(\sigma_1,\ldots,\sigma_n)=1\right\}} (x_1^{\sigma_1} \wedge \cdots \wedge x_n^{\sigma_n})$$

называется **совершенной дизъюнктивной нормальной формой (СДНФ)**.

СДНФ для заданной функции:

$F(x_1, x_2, x_3, x_4) = (x_1\, x_2\, x_3\, \overline{x_4}) \vee (x_1\, \overline{x_2}\, x_3\, x_4) \vee (x_1\, \overline{x_2}\, x_3\, \overline{x_4}) \vee (x_1\, \overline{x_2}\, \overline{x_3}\, x_4) \vee$
$\vee(\overline{x_1}\, x_2\, x_3\, \overline{x_4}) \vee (\overline{x_1}\, \overline{x_2}\, x_3\, x_4) \vee (\overline{x_1}\, \overline{x_2}\, x_3\, \overline{x_4}))$

Аналогично, реализация булевой функции $f(x_1, \ldots, x_n)$ в виде формулы

$$\bigwedge_{\left\{(\sigma_1,\ldots,\sigma_n)\,\middle|\, f(\sigma_1,\ldots,\sigma_n)=1\right\}} (x_1^{\sigma_1} \vee \cdots \vee x_n^{\sigma_n})$$

называется *совершенной конъюнктивной нормальной формой* (СКНФ).

СКНФ для заданной функции:

$F(x_1, x_2, x_3, x_4) = (x_1 \vee x_2 \vee x_3 \vee x_4) \wedge (x_1 \vee x_2 \vee x_3 \vee \overline{x_4}) \wedge (x_1 \vee \overline{x_2} \vee x_3 \vee x_4) \wedge (x_1 \vee \overline{x_2} \vee x_3 \vee \overline{x_4}) \wedge$
$\wedge(x_1 \vee \overline{x_2} \vee \overline{x_3} \vee \overline{x_4}) \wedge (\overline{x_1} \vee x_2 \vee x_3 \vee \overline{x_4}) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3 \vee x_4) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3 \vee \overline{x_4}) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3} \vee \overline{x_4}))$

1.3 Полином жегалкина

Представление булевой функции над базисом $\{0, 1, \wedge, +\}$ называется полиномом Жегалкина. Таким образом, всякая булева функция представима в виде:

$P(x_1, \ldots, x_n) = a_0 \oplus a_1 x_1 \oplus \cdots \oplus a_n x_n \oplus a_{n+1} x_1 x_2 \oplus \cdots \oplus a_{2^n-1} x_1 x_2 \cdots x_n,$

где $\oplus$ — сложение по модулю 2, $a_0, \ldots, a_{2^n-1} \in E_2$, n — количество входных аргументов функции.

Полином жегалкина для заданной функции:

$P(X1, X2, X3, X4) = X_3 \oplus X_2X_3X_4 \oplus X_1 \oplus X_1X_4 \oplus X_1X_3 \oplus X_1X_2X_4 \oplus X_1X_2X_3 \oplus$
$\oplus X_1X_2X_3X_4$

1.4 Бинарное дерево решений

Таблицу истинности булевой функции n переменных можно представить в виде полного бинарного дерева высоты n + 1.

Ярусы дерева соответствуют переменным, дуги дерева соответствуют значениям переменных, например, пунктирная — 0, а слошная — 1. Листья дерева на последнем ярусе хранят значение функции на кортеже, соответствующем пути из корня в этот лист. Такое дерево называется *деревом решений* (или *семантическим деревом*).

Бинарное дерево решений по заданной функции изображено на рисунке 1. Сокращенное дерево решений изображено на рисунке 2.

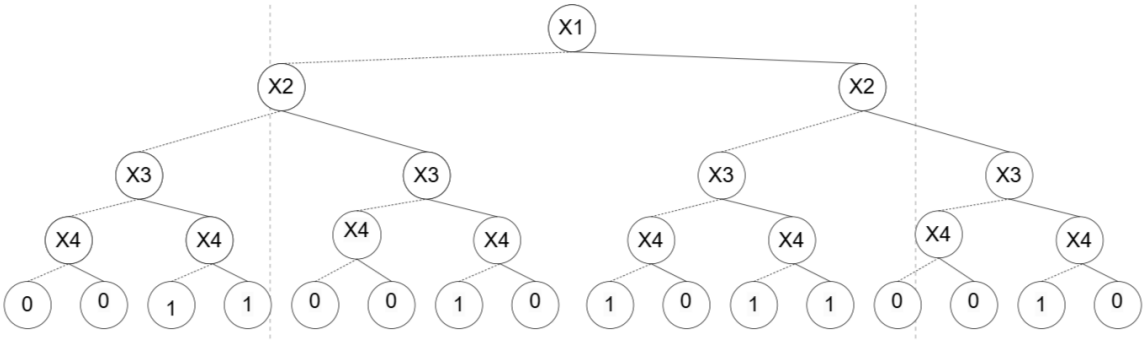


Рис. 1. Бинарное дерево решений

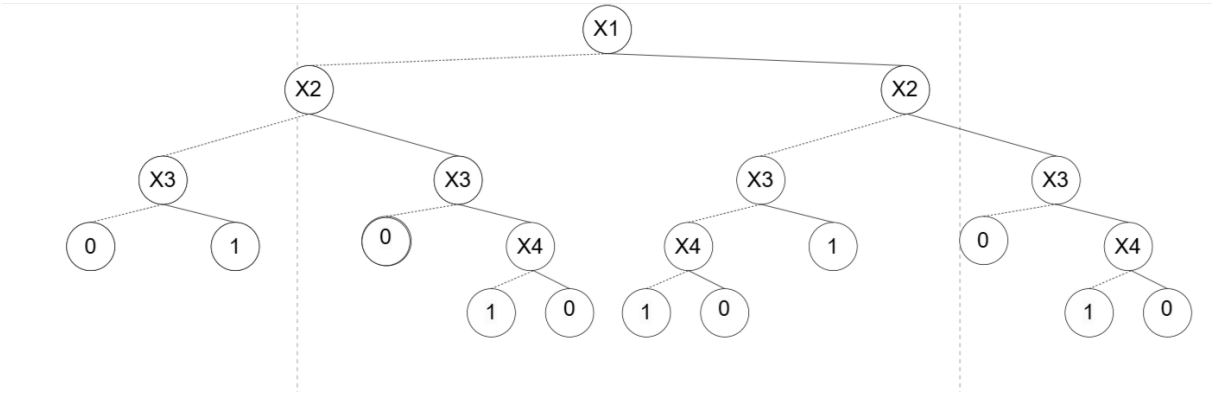


Рис. 2. Сокращенное дерево решений

Если отказаться от древовидности связей, можно получить бинарную диагармму решений.

Бинарная диаграмма решений для заданной функции изображена на рисунке 3.

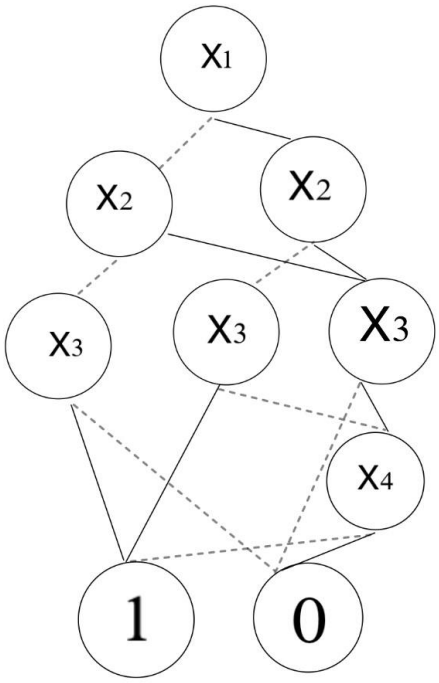


Рис. 3. Бинарная диаграмма решений

Синтаксическое дерево решений по полиному Жегалкина изображено на рисунке 4.

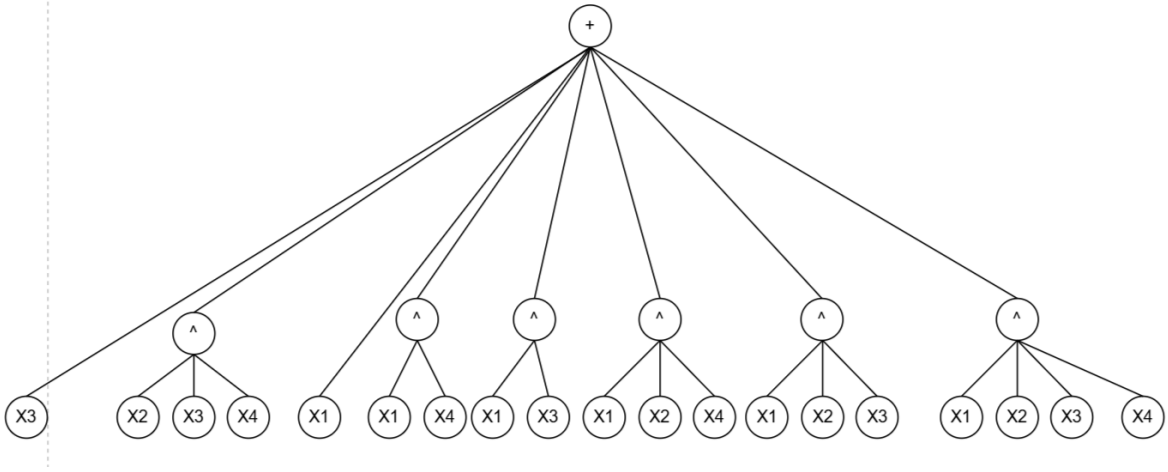


Рис. 4. Синтаксическое дерево решений

2 Особенности реализации

2.1 Пользовательские структуры данных и классы

В данной реализации были использованы следующие структуры данных и классы:

- **TreeNode**: Узел дерева решений, содержащий значение и ссылки на позитивное и негативное поддеревья.
  - `QString value`: Значение узла (может быть булевым значением или переменной, например, $X1$).
  - `TreeNode* trueBranch`: Указатель на позитивное поддерево.
  - `TreeNode* falseBranch`: Указатель на негативное поддерево.
- **DecisionTree**: Класс, представляющий дерево решений. Содержит корень дерева и методы для его построения и оценки.
  - `TreeNode* root`: Указатель на корень дерева решений.
  - `DecisionTree()`: Конструктор, который инициализирует корень и вызывает метод для построения дерева.
  - `void buildTree()`: Метод, который реализует логику построения дерева.
  - `bool evaluate(TreeNode* node, bool x1, bool x2, bool x3, bool x4)`: Метод для оценки логической функции по дереву.
- **TruthTable**: Класс, реализующий истинную таблицу с методами для преобразования в СДНФ, СКНФ и вычисления значений.
  - `DecisionTree tree`: Объект дерева решений, используемый для оценки значений.
  - `QVector<bool> truthValues`: Вектор, содержащий значения истинности для всех входных комбинаций.
  - `TruthTable(const QVector<bool> truthValues)`: Конструктор, который инициализирует класс с переданными значениями истинности и вызывает метод для заполнения значений переменных.
  - `QString toSDNF()`: Метод для преобразования в стандартную дизъюнктивную нормальную форму.
  - `QString toSKNF()`: Метод для преобразования в стандартную конъюнктивную нормальную форму.
  - `QString toZhegalkin()`: Метод для получения многочлена Жегалкина.
  - `bool evaluateSKNF(int x1, int x2, int x3, int x4)`: Метод для оценки логической функции в форме СКНФ.

2.2 Описание функций

2.2.1 Метод `evaluate` класса `DecisionTree`

Принимает указатель на узел и четыре булевые переменные, возвращает результат как булевое значение, основанное на логике дерева решений. Код представлен в «Листинг 1».

Вход: Указатель на узел дерева $node$ и четыре булевые переменные $x1, x2, x3, x4$.

Выход: Булевое значение, результат оценки логической функции на основе дерева решений.

Листинг 1. evaluate

```
bool evaluate(TreeNode* node, bool x1, bool x2, bool x3, bool x4) {
    if (node->value == "1") return true;
    if (node->value == "0") return false;

    if (node->value == "X1")
        return x1 ? evaluate(node->trueBranch, x1, x2, x3, x4) : evaluate(
            node->falseBranch, x1, x2, x3, x4);

    if (node->value == "X2")
        return x2 ? evaluate(node->trueBranch, x1, x2, x3, x4) : evaluate(
            node->falseBranch, x1, x2, x3, x4);

    if (node->value == "X3")
        return x3 ? evaluate(node->trueBranch, x1, x2, x3, x4) : evaluate(
            node->falseBranch, x1, x2, x3, x4);

    if (node->value == "X4")
        return x4 ? evaluate(node->trueBranch, x1, x2, x3, x4) : evaluate(
            node->falseBranch, x1, x2, x3, x4);

    return false;
}
```

2.2.2 Метод `evaluate` класса `DecisionTree`

Вызывает метод evaluate на корневом узле дерева, используя четыре булевые переменные. Возвращает результат как булевое значение. Код представлен в «Листинг 2».

Вход: Четыре булевые переменные $x1, x2, x3, x4$.

Выход: Булевое значение, представляющее результат оценки с использованием корня дерева решений.

Листинг 2. evaluate

```
bool evaluate(bool x1, bool x2, bool x3, bool x4) {
    return evaluate(root, x1, x2, x3, x4);
}
```

2.2.3 Функция `createXORTriangle`

Формирует хог треугольник и возвращает его левую сторону - вектор целых чисел, представляющий значения многочлена Жегалкина. Код представлен в «Листинг 3».

Вход: Строка $binary$, представляющая двоичное число.

Выход: Вектор целых чисел представляющий коэффициенты полинома Жегалкина.

Листинг 3. createXORTriangle

```
std::vector<int> createXORTriangle(const std::string& binary) {
    std::vector<int> initialRow;
    for (char bit : binary) {
        initialRow.push_back(bit - '0');
    }

    std::vector<std::vector<int>> triangle;
    triangle.push_back(initialRow);

    for (size_t row = 1; row < initialRow.size(); ++row) {
        std::vector<int> newRow;
        for (size_t col = 0; col < triangle[row - 1].size() - 1; ++col) {
            newRow.push_back(triangle[row - 1][col] ^ triangle[row - 1][col
                + 1]);
        }
        triangle.push_back(newRow);
    }

    std::vector<int> zheg;
    for (const auto& row : triangle) {
        if (!row.empty()) {
            zheg.push_back(row[0]);
        }
    }

    return zheg;
}
```

2.2.4 Метод `evaluateSDNF1` класса `TruthTable`

Выдает значение функции на основе СДНФ. Код представлен в «Листинг 4».
Вход: Строка $sdnf$ и четыре булевые переменные $x1, x2, x3, x4$.

Выход: Булевое значение на основе стандартной дизъюнктивной нормальной формы (СДНФ).

Листинг 4. evaluateSDNF1

```
bool evaluateSDNF1(const QString& sdnf, bool x1, bool x2, bool x3, bool x4)
    {
    QVector<bool> values = { x1, x2, x3, x4 };
    QStringList terms = sdnf.split("␣v␣", QString::SkipEmptyParts);

    for (const QString& term : terms) {
        bool termResult = true;

        QStringList variables = term.trimmed().mid(1, term.length() - 2).
           split("␣");
        for (const QString& variable : variables) {
            bool isNegated = variable.startsWith("!");
            int varIndex = variable.mid(isNegated ? 2 : 1).toInt() - 1;

            if (varIndex < 0  varIndex >= values.size()) {
                termResult = false;
                break;
            }

            termResult &= isNegated ? !values[varIndex] : values[varIndex];
        }

        if (termResult) {
            return true;
        }
    }

    return false;
}
```

2.2.5 Метод `evaluateXORFunction` класса `TruthTable`

Выдает значение функци на основе полинома Жегалкина. Код представлен в «Листинг 5».
Вход: Строка $function$ и четыре булевые переменные $x1, x2, x3, x4$.

Выход: Булевое значение, представляющее значение логической функции на основе полинома Жегалкина.

Листинг 5. evaluateXORFunction

```
bool evaluateXORFunction(const QString& function, bool x1, bool x2, bool x3
    , bool x4) {
    QVector<bool> values = { x1, x2, x3, x4 };
    QStringList terms = function.split("␣+␣", QString::SkipEmptyParts);
    bool result = false;

    for (const QString& term : terms) {
        bool termResult = true;

        for (const QChar& variable : term) {
            if (variable == 'x') {
                continue;
            }

            int varIndex = variable.digitValue() - 1;

            if (varIndex < 0  varIndex >= values.size()) {
                termResult = false;
                break;
            }

            termResult &= values[varIndex];
        }

        result ^= termResult;
    }

    return result;
}
```

2.2.6 Метод `evaluateSKNF` класса `TruthTable`

Выдает значение функции на основе СКНФ. Код представлен в «Листинг 6».

Вход: Четыре целых числа, представляющих булевые переменные (0 или 1).

Выход: Булевое значение, представляющее значение функции на основе СКНФ.

Листинг 6. evaluateSKNF

```
bool evaluateSKNF(int x1, int x2, int x3, int x4) {
    QString sknf = toSKNF();
    return !evaluateLogicalFormula(sknf, { x1, x2, x3, x4 });
}
```

2.2.7 Метод `toSDNF` класса `TruthTable`

Преобразует вектор булевой функции в строку, представляющую СДНФ. Код представлен в «Листинг 7».

Вход: Вектор truthValues - вектор булевой функции.

Выход: Строка, представляющая СДНФ.

Листинг 7. toSDNF

```
QString toSDNF() {
    QStringList sdnfTerms;

    for (int i = 0; i < truthValues.size(); ++i) {
        if (truthValues[i]) {
            QString term = "";
            for (int j = 0; j < 4; ++j) {
                term += ((i >> (3 - j)) & 1) ? "x" + QString::number(j + 1)
                    + "␣" : "!" + QString("x") + QString::number(j + 1) + "
                   ␣";
            }
            sdnfTerms.append("(" + term.trimmed() + ")");
        }
    }
    return sdnfTerms.join("␣v␣");
}
```

2.2.8 Метод `toSKNF` класса `TruthTable`

Преобразует вектор булевой функции в строку, представляющую СКНФ. Код представлен в «Листинг 8».

Вход: Вектор truthValues - вектор булевой функции.

Выход: Строка, представляющая СКНФ.

Листинг 8. toSKNF

```
QString toSKNF() {
    QStringList sknfTerms;

    for (int i = 0; i < truthValues.size(); ++i) {
        if (!truthValues[i]) {
            QString term = "";
            for (int j = 0; j < 4; ++j) {
                term += ((i >> (3 - j)) & 1) ? "!" + QString("x") + QString
                    ::number(j + 1) : "x" + QString::number(j + 1);
                if (j < 3) term += "␣v␣";
            }
            sknfTerms.append("(" + term.trimmed() + ")");
        }
    }
    return sknfTerms.join("␣^␣");
}
```

2.2.9 Метод `toZhegalkin` класса `TruthTable`

Преобразует булевую функцию в строку, представляющую полином Жегалкина, используя созданный ранее треугольник. Код представлен в «Листинг 9».
Вход: Константная строка - вектор булевой функции.

Выход: Строка, представляющая полином Жегалкина.

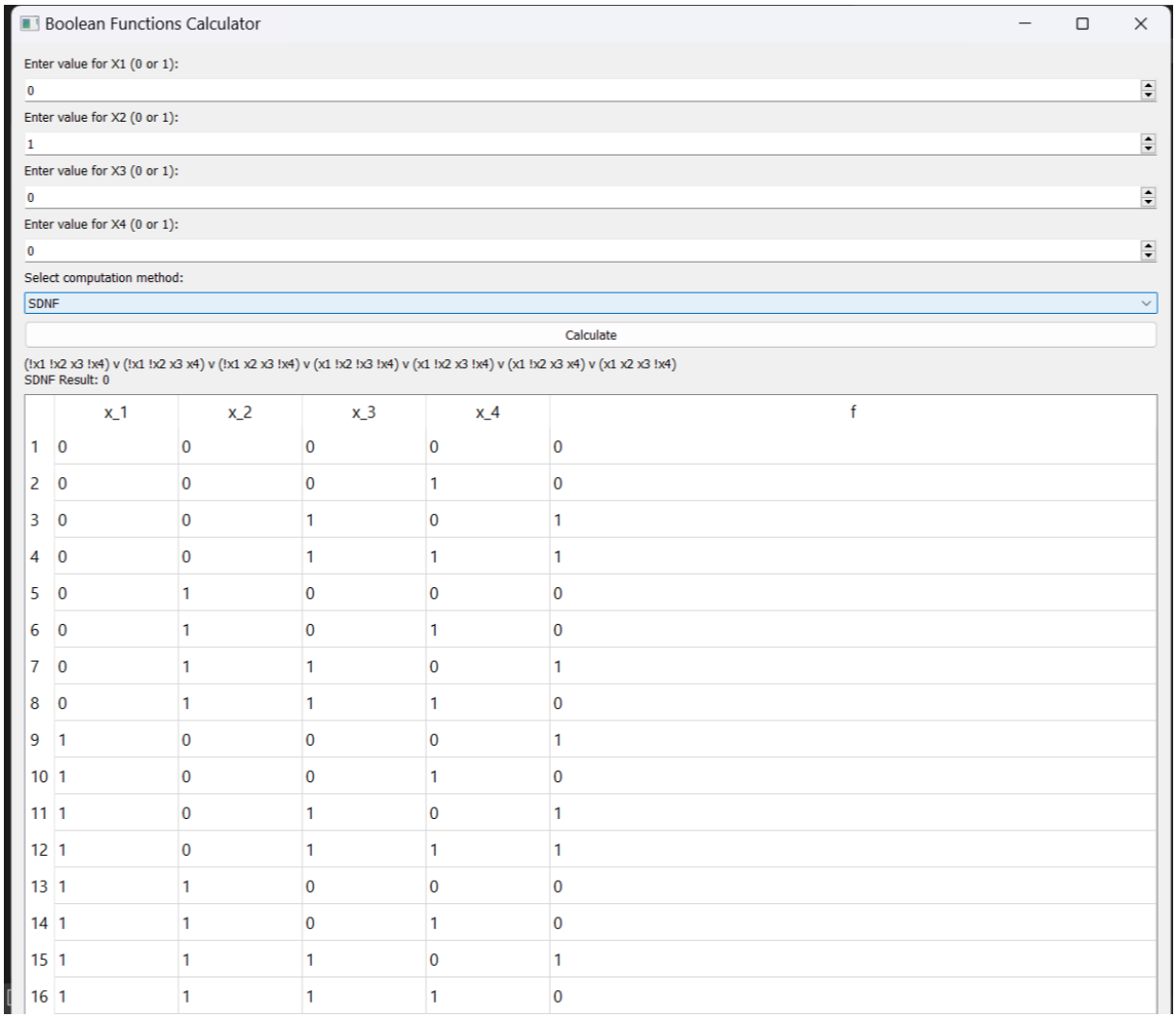
Листинг 9. toZhegalkin

```
QString toZhegalkin() {
    std::vector<int> zhegValues = createXORTriangle("0011001010110010");
    QStringList zhegTerms;

    for (int i = 0; i < zhegValues.size(); ++i) {
        if (zhegValues[i] == 1) {
            QString term;
            for (int j = 0; j <= 3; ++j) {
                if (variableValues[i][j]) {
                    term += "x" + QString::number(j + 1);
                }
            }
            zhegTerms.append(term);
        }
    }

    return zhegTerms.join("␣+␣");
}
```

3 Результаты работы

3.1 СДНФ

Пользователь может посмотреть СДНФ для заданной таблицы и вычислить значение для заданных значений переменных (см. рис. 5).

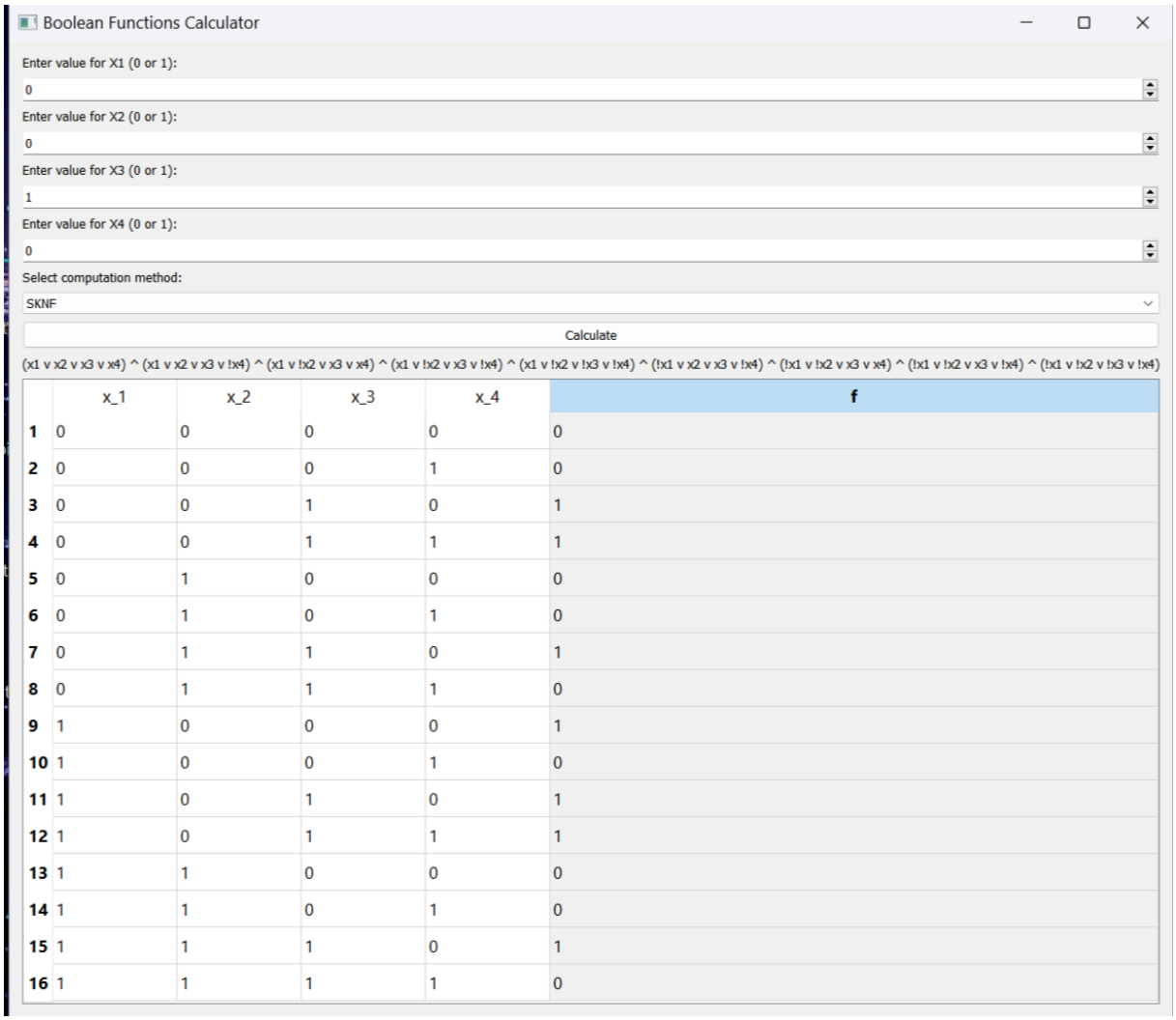


	x_1	x_2	x_3	x_4	f
1	0	0	0	0	0
2	0	0	0	1	0
3	0	0	1	0	1
4	0	0	1	1	1
5	0	1	0	0	0
6	0	1	0	1	0
7	0	1	1	0	1
8	0	1	1	1	0
9	1	0	0	0	1
10	1	0	0	1	0
11	1	0	1	0	1
12	1	0	1	1	1
13	1	1	0	0	0
14	1	1	0	1	0
15	1	1	1	0	1
16	1	1	1	1	0

Рис. 5. СДНФ

3.2 СКНФ

Пользователь может посмотреть СКНФ для данной таблицы (см. рис. 6).

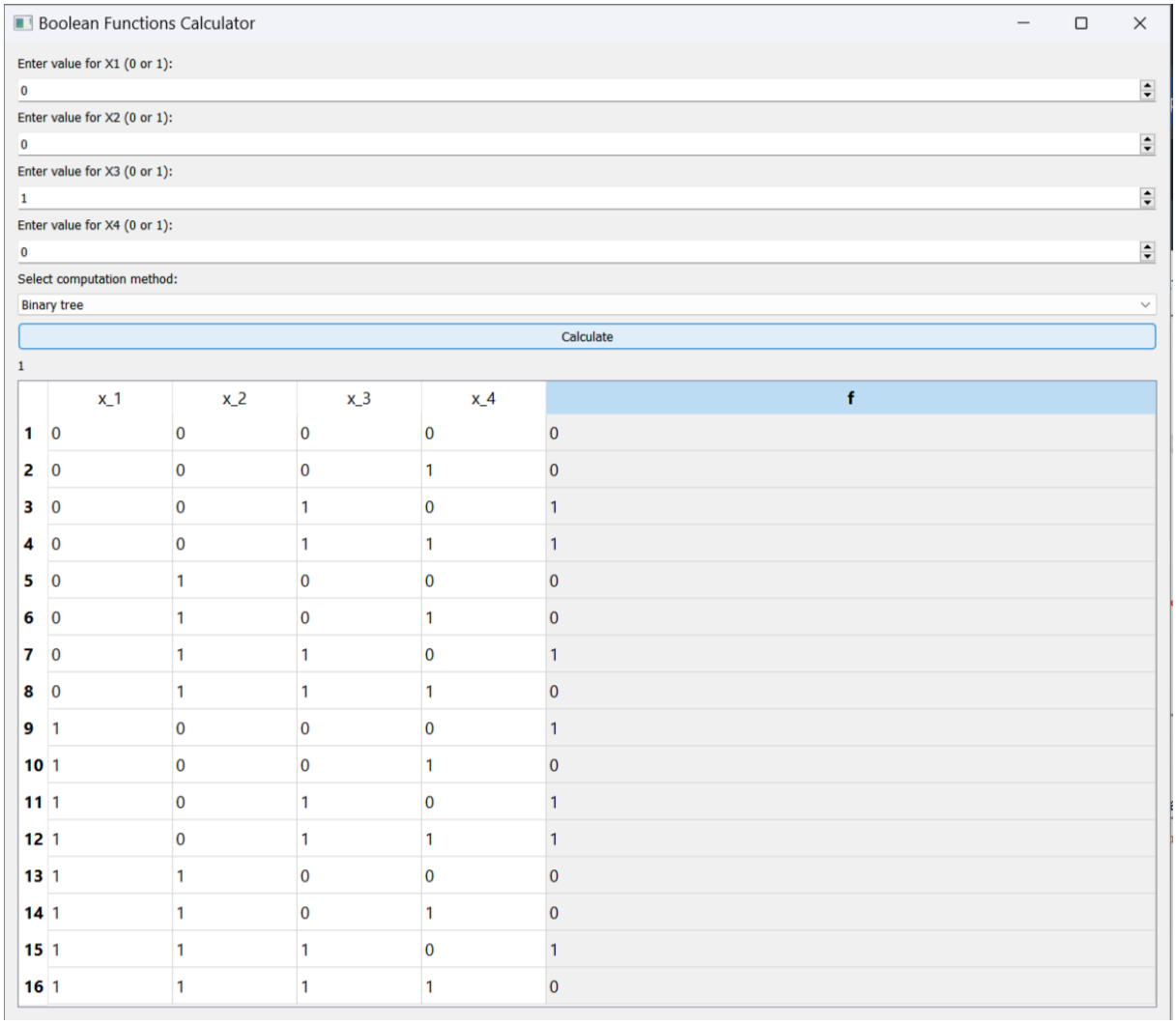


	x_1	x_2	x_3	x_4	f
1	0	0	0	0	0
2	0	0	0	1	0
3	0	0	1	0	1
4	0	0	1	1	1
5	0	1	0	0	0
6	0	1	0	1	0
7	0	1	1	0	1
8	0	1	1	1	0
9	1	0	0	0	1
10	1	0	0	1	0
11	1	0	1	0	1
12	1	0	1	1	1
13	1	1	0	0	0
14	1	1	0	1	0
15	1	1	1	0	1
16	1	1	1	1	0

Рис. 6. СКНФ

3.3 Получение решения с помощью бинарной диаграммы решений

Пользователь может получить значение функции для заданных значений переменных по бинарной диаграмме решений (см. рис. 7).

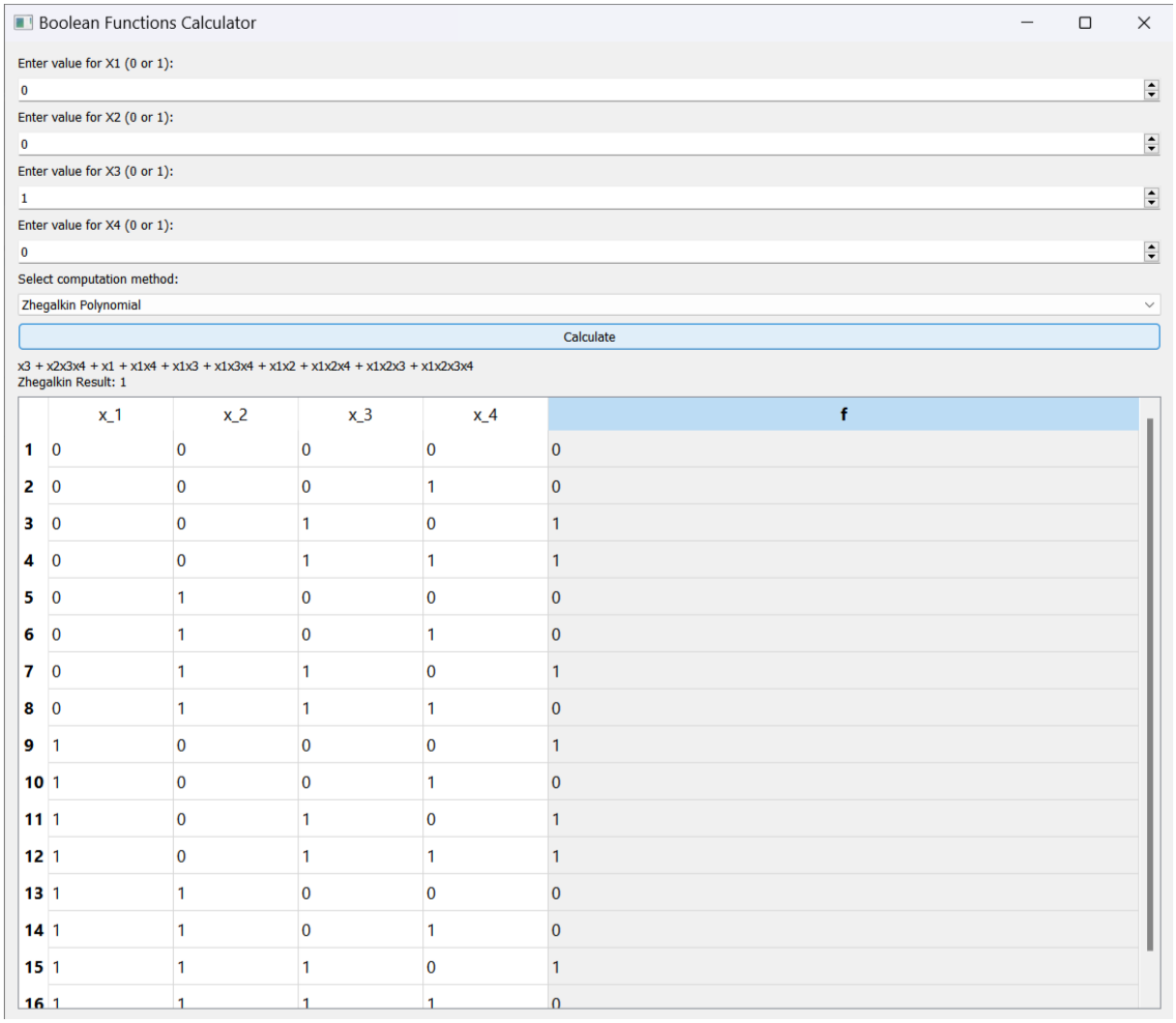


	x_1	x_2	x_3	x_4	f
1	0	0	0	0	0
2	0	0	0	1	0
3	0	0	1	0	1
4	0	0	1	1	1
5	0	1	0	0	0
6	0	1	0	1	0
7	0	1	1	0	1
8	0	1	1	1	0
9	1	0	0	0	1
10	1	0	0	1	0
11	1	0	1	0	1
12	1	0	1	1	1
13	1	1	0	0	0
14	1	1	0	1	0
15	1	1	1	0	1
16	1	1	1	1	0

Рис. 7. Бинарная диаграмма

3.4 Полином Жегалкина

Пользователь может вывести на экран полином Жегалкина для данной функции и вычислить его для заданных значений переменных (см. рис. 8).



	x_1	x_2	x_3	x_4	f
1	0	0	0	0	0
2	0	0	0	1	0
3	0	0	1	0	1
4	0	0	1	1	1
5	0	1	0	0	0
6	0	1	0	1	0
7	0	1	1	0	1
8	0	1	1	1	0
9	1	0	0	0	1
10	1	0	0	1	0
11	1	0	1	0	1
12	1	0	1	1	1
13	1	1	0	0	0
14	1	1	0	1	0
15	1	1	1	0	1
16	1	1	1	1	0

Рис. 8. Полином Жегалкина

Заключение

В данной лабораторной работе была реализована программа, способная:

- Генерировать и отображать таблицу истинности логических функций с четырьмя переменными.
- Вычислять значения булевых функций по СДНФ, БДР, полиному Жегалкина.
- Находить и выводить на экран СДНФ.

Так же вручную были построены:

- Таблица истинности
- Дерево решений
- Бинарная диаграмма решений
- Синтаксическое дерево

Достоинства программы

- Программа имеет удобный графический интерфейс, что упрощает взаимодействие с пользователем.
- Таблица истинности выводится на экран для большей наглядности.
- Использования контейнеров Qt, что помогает избежать утечек памяти и упрощает работу с кодом.
- Возможность быстрой адаптации программы под другую булевую функцию.

Недостатки программы

- В методе evaluate класса DecisionTree код для обработки переменных X1, X2, X3 и X4 повторяется.
- Код содержит много вложенных условий, что может затруднять понимание логики работы.

Масштабирование
Программа может быть улучшена путем:

- Расширения функциональности с поддержкой более чем 4 переменных.
- Дать пользователю возможность изменять значения таблицы истинности во время работы программы.
- Добавления визуализации БДР.

Список использованной литературы

[1] Новиков, Ф.А. *Н73 Дискретная математика дл я программистов: Учебник дл я вузов. 3-е изд. — СПб.: Питер, 2009. — 384 е.: ил. — (Серия «Учебник дл я вузов»))*.