Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Основы алгоритмизации и программирования

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

КАЛЬКУЛЯТОР ЛОГИЧЕСКИХ ВЫРАЖЕНИЙ С ВОЗМОЖНОСТЬЮ МИНИМИЗАЦИИ БУЛЕВЫХ ФУНКЦИЙ

БГУИР КР 6-05-0612-01 130 ПЗ

Студент: гр. 451001 Цурко В. Е.

Руководитель:

асс. Фадеева Е.Е.

Минск 2025

       Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой ПОИТ

–––––––––––––Лапицкая Н.В.–––––––––––

(подпись)

––11.02.2025 г.

ЗАДАНИЕ

по курсовому проектированию

Студенту   Цурко Владиславу Евгеньевичу  –––––––––––––––––––

1. Тема работы  Калькулятор логических выражений с возможностью минимизации булевых функций. –––––––––––––––––––––––                  ––   ––

2. Срок сдачи студентом законченной работы––02.06.2025 г.–––

3. Исходные данные к работе: язык программирования Delphi, алгоритм получения СДНФ и СКНФ, алгоритм создания полинома Жегалкина, алгоритм минимизации булевых функций с использованием метода Квайна-МакКласки и Карты Карно, работа с файлами формата .txt

4. Содержание расчётно-пояснительной записки (перечень вопросов, которые подлежат разработке)

Введение.

1. Анализ прототипов, литературных источников и формирование требований к проектируемому программному средству;

2. Анализ требований к программному средству и разработка функциональных требований;

3. Проектирование программного средства;

4. Создание (конструирование) программного средства;

5. Тестирование, проверка работоспособности и анализ полученных результатов;

6. Руководство по установке и использованию;

Список используемой литературы

Заключение

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

"Калькулятор логически выражений с возможностью минимизации булевых функций. ", А1, схема программы, чертеж.

6. Консультант по курсовой работе

Фадеева Е.Е.

7. Дата выдачи задания 11.02.2025 ––––––––––––––––––––   –

8. Календарный график работы над курсовой работой на весь период проектирования (с обозначением сроков выполнения и процентом от общего объёма работы):

раздел 1 к 24.02.2025 – 15 % готовности работы;

разделы 2, 3 к 20.03.2025 – 30 % готовности работы;

разделы 4, 5 к 21.04.2025 – 60 % готовности работы;

раздел 6 к 19.05.2025 – 90 % готовности работы;

оформление пояснительной записки и графического материала к 02.06.2025 – 100 % готовности работы.

Защита курсовой работы с 02.06.2025 по 06.06.2025 г.––––––––––––––––––––

РУКОВОДИТЕЛЬ–––––– Е.Е.Фадеева

(подпись)

Задание принял к исполнению –––\_\_\_\_––

(дата и подпись студента)

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 5](#_Toc199189323)

[1. Анализ прототипов, литературных источников и формирование требований к проектируемому программному средству 6](#_Toc199189324)

[2. Анализ требований к программному средству и разработка функциональных требований 10](#_Toc199189325)

[3. Проектирование программного средства 14](#_Toc199189326)

[4. Создание (конструирование) программного средства 21](#_Toc199189327)

[5. Тестирование, проверка работоспособности и анализ полученных результатов 25](#_Toc199189328)

[6. Руководство по установке и использованию 29](#_Toc199189329)

[ЗАКЛЮЧЕНИЕ 37](#_Toc199189330)

[ПРИЛОЖЕНИЕ А 39](#_Toc199189331)

[ПРИЛОЖЕНИЕ Б 54](#_Toc199189332)

# ВВЕДЕНИЕ

В современном мире логические выражения и булевы функции играют важную роль в технических сферах деятельности. Наиболее явно их можно наблюдать в программировании. Калькулятор логических выражений с возможностью минимизации булевых функций очень полезен в данной сфере. Его использование позволяет снизить сложность цифровых схем, уменьшить объем используемой памяти и повысить производительность вычислений. В настоящее время существует множество программных решений, связанных с булевыми функциями. Однако многие из них не имеют достаточного функционала, требуют значительных вычислительных ресурсов, а также подключения к сети Интернет.

Целью данного курсового проекта является разработка калькулятора булевых выражений с возможностью минимизации булевых функций, который позволит пользователю получить нужное ему решение в удобной форме. Основное внимание уделяется разработке логических алгоритмов, применению методов карт Карно и Квайна-МакКласки.

В ходе работы разрабатывается программная реализация калькулятора и проводится тестирование его функциональности.

Курсовой проект состоит из нескольких разделов. В первом разделе рассматривается предметная область булевой алгебры и методов минимизации логических функций, анализ готовых решений, на основе которого перечислены основные функции, прописаны их недостатки. Также прописаны требования к данной курсовой работе. Второй раздел содержит перечень функций, предусмотренных в калькуляторе. В следующем пункте продемонстрированы схемы алгоритмов и описание программы. В четвертом разделе содержится описание основных структур решений, которые использовались при создании калькулятора. Далее проводится тестирование программы, анализируются результаты ее работы и оценивается эффективность примененных алгоритмов. Заключительный раздел содержит подробное руководство по установке и использованию калькулятора.

Данный курсовой проект выполнен мной лично, проверен на заимствования, процент оригинальности составляет ХХ% (отчет о проверке на заимствования прилагается).

# 1. Анализ прототипов, литературных источников и формирование требований к проектируемому программному средству

Назначение разработки – создание программного средства для вычисления логических выражений и минимизации булевых функций.

Были проанализированы известные решения, некоторые из которых описаны далее.

Из достоинств первой программы [1] можно выделить интуитивно понятный интерфейс, возможность получения таблицы истинности для основных логических операций (рисунок 1.1). Недостатки: непропорциональное масштабирование окон, очень ограниченные возможности для решения задач, отсутствие каких-либо возможностей минимизации булевых выражений и остальных критериев отличных от таблицы истинности, ограничение в количестве вводимых значений, работа приложения осуществляется только при наличии Интернет-соединения, отсутствие инструкции к использованию, а также описания его функционала.

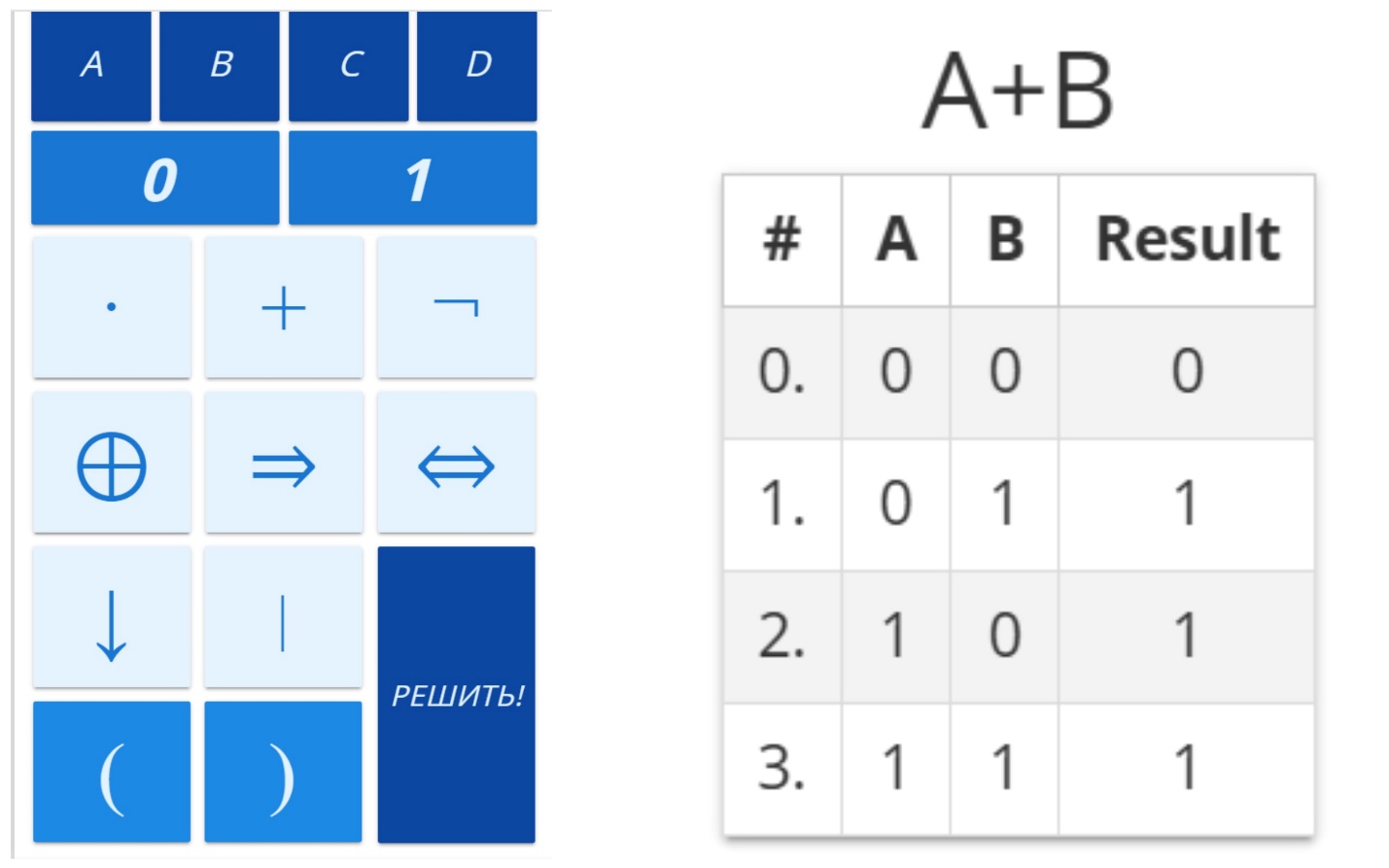


Рисунок 1.1 – Интерфейс первого найденного аналога

Из достоинств второй программы [2] можно выделить возможность получения результатов для большого количества значений операндов, расширенный функционал, удобный интерфейс, наличие видео инструкции по использованию приложения, наличие описания решения задачи. Недостатки: работа осуществляется только при наличии Интернет-соединения, огромный теоретический материал, в коде не обрабатывается событие неправильного ввода выражения (в таком случае калькулятор пытается решить неправильную строку в следствии чего сайт вылетает).

На основе проведенного анализа и с учетом требований, указанных в задании на курсовое проектирование, я выбрал следующий функционал моего калькулятора:

1. Поддержка основных логических операций, а также генерация таблицы истинности для логического выражения. Логическое выражение – выражение, в котором операндами являются объекты, над которыми выполняются логические операции. Результатом его выполнения является одно из двух логических значений: либо Истина (true, 1), либо Ложь (false, 0). Основные логические операции: конъюнкция, дизъюнкция, инверсия, импликация, эквивалентность и исключающее ИЛИ. Каждая из них задается таблицей истинности, которая показывает результаты их выполнения для всех возможных логических значений исходных выражений. Количество вариантов зависит от количества операндов. Таким образом, если число аргументов в логическом выражении равно N, то таблица истинности будет содержать 2N различных строк [3]. Рассматриваются результаты операций для одного и двух выражений.

Конъюнкция – это сложное логическое выражение, которое считается истинным в том и только том случае, когда оба значения операндов являются истинными, во всех остальных случаях – ложно. Обозначения: &, and (в Delphi), ∧, И.

Таблица 1.1 – Таблица истинности конъюнкции

|  |  |  |
| --- | --- | --- |
| А | В | Результат |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Дизъюнкция – это сложное логическое выражение, которое истинно, если значение хотя бы одного операнда истинно и ложно тогда и только тогда, когда оба из них ложны. Обозначения: V, |, или, +, or (в Delphi).

Таблица 1.2 – Таблица истинности дизъюнкции

|  |  |  |
| --- | --- | --- |
| А | В | Результат |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

Инверсия – это сложное логическое выражение, если значение операнда истинно, то результат отрицания будет ложным, и наоборот. Обозначения: НЕ, not (в Delphi), ¬, ¯.

Таблица 1.3 – Таблица истинности инверсии

|  |  |
| --- | --- |
| А | Результат |
| 0 | 0 |
| 1 | 1 |
| 0 | 1 |
| 1 | 1 |

Импликация – это сложное логическое выражение, которое истинно при всех значениях операндов, кроме случая, когда первый – истина, второй – ложь. Обозначения: →, ⊃, (not A) or B (в Delphi).

Таблица 1.4 – Таблица истинности импликации

|  |  |  |
| --- | --- | --- |
| А | В | Результат |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Эквивалентность – это сложное логическое выражение, которое истинно при одинаковых значениях операндов.

Таблица 1.5 – Таблица истинности эквивалентности

|  |  |  |
| --- | --- | --- |
| А | В | Результат |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

2. Вычисление значений логических выражений. Поддержка способов представления совершенная дизъюнктивная (СДНФ) и конъюнктивная (СКНФ) нормальные формы, полином Жегалкина.

Дизъюнктивная нормальная форма, ДНФ – нормальная форма, в которой булева функция имеет вид дизъюнкции нескольких простых конъюнктов. Пример ДНФ: f(x,y,z)=(x∧y)∨(y∧¬z). Совершенная дизъюнктивная нормальная форма, СДНФ – ДНФ, удовлетворяющая условиям: в ней нет одинаковых простых конъюнкций, каждая простая конъюнкция полная. Пример СДНФ: f(x,y,z)=(x∧¬y∧z)∨(x∧y∧¬z) [4].

Конъюнктивная нормальная форма, КНФ – нормальная форма, в которой булева функция имеет вид конъюнкции нескольких простых дизъюнктов. Пример КНФ: f(x,y,z)=(x∨y)∧(y∨¬z). Совершенная конъюнктивная нормальная форма, СКНФ – это такая КНФ, которая удовлетворяет условиям: в ней нет одинаковых простых дизъюнкций, каждая простая дизъюнкция полная. Пример СКНФ: f(x,y,z)=(x∨¬y∨z)∧(x∨y∨¬z) [5]

Полином Жегалкина – полином с коэффициентами вида 0 и 1, где в качестве произведения берётся конъюнкция, а в качестве сложения исключающее или. Полином был предложен в 1927 году И. И. Жегалкиным в качестве средства для представления функций булевой логики. Полином Жегалкина имеет следующий вид: P=a000…000⊕a100…0x1⊕a010…0x2⊕…⊕a00…01xn⊕a110…0x1x2⊕…⊕a00…011xn−1xn⊕…⊕a11…1x1x2…xn

3. Минимизация булевых функций. Метод Квайна-МакКласки,

Карта Карно.

Метод Квайна-МакКласки — это алгоритмический способ минимизации булевых функций, который подходит для автоматизации упрощения логических выражений и используется в компьютерных программах. Он основан на объединении минтермов (элементарных конъюнкций) по определённым правилам. Алгоритм метода Квайна-МакКласки:

1. Составить таблицу минтермов. Найти все минтермы (элементарные конъюнкции), для которых функция принимает значение 1. Привести их к двоичной форме.
2. Группировка минтермов. Разделить их на группы по количеству единиц в двоичном представлении.
3. Объединение и сокращение. Объединять пары минтермов, отличающиеся на один бит, заменяя различающийся бит на "-". Повторять процесс, пока не останутся термы, которые нельзя объединить.
4. Построение таблицы покрытия. Составить таблицу, показывающую, какие минтермы покрываются какими термами. Выбрать минимальное покрытие.

Карта Карно — это графический метод минимизации булевых функций, удобный для функций с 2–4 переменными. Она представляет таблицу, в которой логические выражения группируются для их упрощения.

Алгоритм работы с картой Карно:

1. Построить карту. Определить количество переменных. Нарисовать таблицу с ячейками, где каждая ячейка представляет минтерм.
2. Заполнить карту. Вписать 1 в ячейки, соответствующие минтермам, для которых функция равна 1.
3. Группировка единиц. Объединять соседние 1 (по степеням двойки: 1, 2, 4, 8). Каждая группа даёт упрощённое выражение.
4. Записать минимизированную функцию. Группированные термы дают итоговое логическое выражение.

Входные данные: логическое выражение, введенное пользователем в строковом формате, выбранные им функции, которые необходимо выполнить. Ввод будет происходить в текстовое поле путем нажатия кнопок, расположенных на экране.

Выходные данные: решение задачи, поставленной пользователем, в файле формата .txt.

# 2. Анализ требований к программному средству и разработка функциональных требований

Генерация таблицы истинности является одной из ключевых функций калькулятора. Пользователь должен ввести выражение, после чего программа должна вычислить его. Для того, чтобы калькулятор считал то, что ввел пользователь, используется приведение булевого выражения к польской записи (также известной как префиксная запись), который осуществляется с использованием алгоритма сортировочной станции Дейкстры [6].

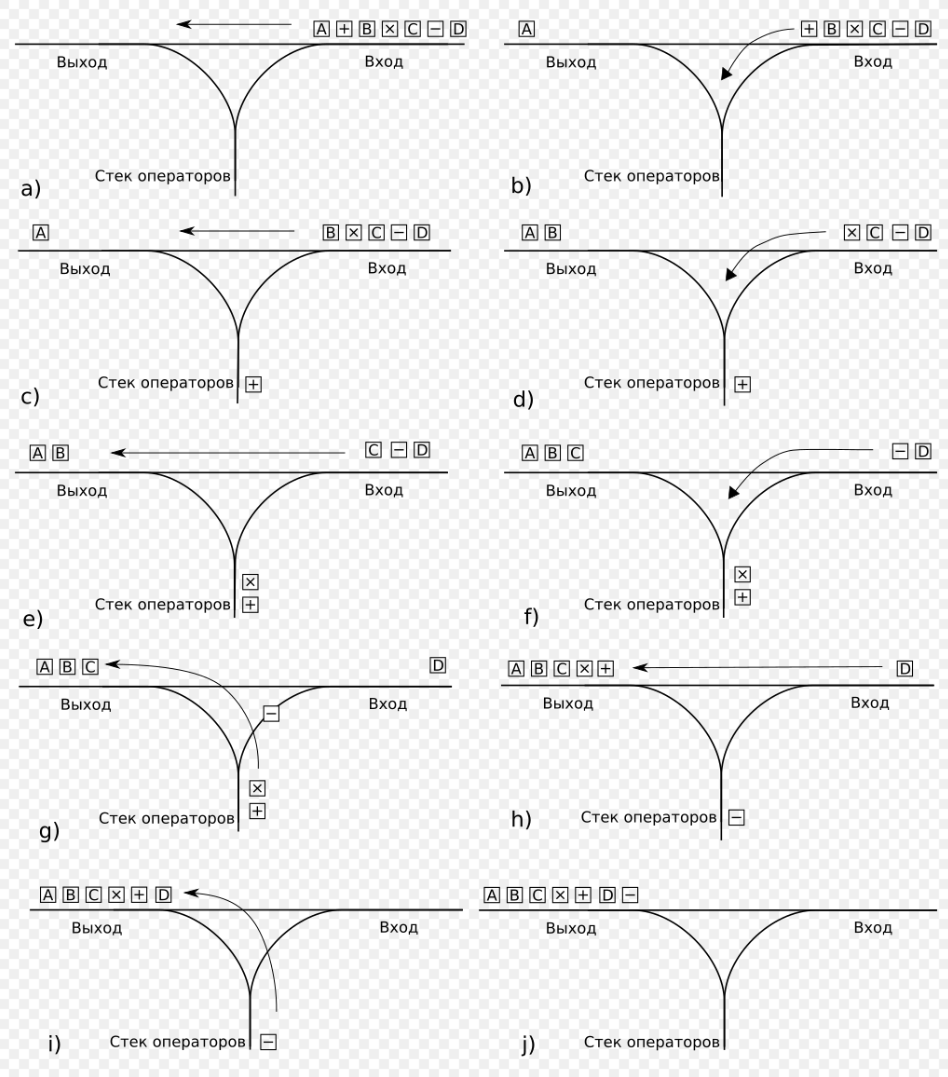


Рисунок 2.1 – Алгоритм дейкстры

Для первого действия в данном алгоритме необходимо создать пустые массивы для операторов и для конечного результата. Далее мы идем по выражению по порядку перебирая введенные значения:

1. если текущий символ – операнд (переменная), то его нужно добавить в массив конечного результата по текущему индексу,
2. если это оператор (булева функция), то нужно определить его приоритет и проверить массив операторов. Если в нем нет булевых функций с большим приоритетом, то поместить в массив. Если есть, переместить их в конечный результат, а текущий оператор записать в пустой массив,
3. после обработки всего выражения переместить оставшиеся функции из массива для операторов в конечный результат.

После данных действий мы получим польскую запись. В зависимости от количества переменных генерируются всевозможные значения true или false для каждой из них. Далее, перебирая и подставляя вместо переменных сгенерированные значения, происходит считывание массива с конечным результатом:

1. если элемент является булевым значением, поместить в массив булевых значений,
2. если это оператор, то определяем какая это булева функция и в зависимости от того, одноместная она или двухместная, берем нужное количество операндов из массива булевых значений, а их результат записываем вместо них,
3. в самом конце алгоритма останется лишь одно булево значение, которое является ответом выражения,

После получения таблицы истинности мы можем реализовать алгоритм получения СДНФ и СКНФ. Для них необходимо знать значение всего выражения при всевозможных значениях параметров.

Согласно работе [7], для получения СДНФ необходим следующий алгоритм:

1. из таблицы всевозможных значений берем только те, в которых значение выражения будет истинным,
2. в данной выборке проверяем значения каждой из переменных: если переменная равна 1, оставить ее в исходном виде (Например: A). Иначе приписать ее отрицание (Например: not A). В математике формирование таких выборок называется минтермами,
3. все минтермы объединяются с помощью логической операции AND.

Алгоритм для получения СКНФ:

1. из таблицы всевозможных значений берем только те, в которых значение выражения будет ложным,
2. в данной выборке проверяем значения каждой из переменных: если переменная равна 0, оставить ее в исходном виде (Например: A). Иначе приписать ее отрицание (Например: not A). В математике формирование таких выборок называется макстермами,
3. все макстермы объединяются с помощью логической операции OR.

Следующий алгоритм булевого калькулятора – построение полинома Жегалкина используя таблицу истинности, полученную ранее. Его коэффициенты вычисляются при помощи операции сложения по модулю 2 (⊕). Каждый коэффициент отвечает за вклад определенного набора переменных (или их комбинации) в итоговое выражение. При этом их вычисление идет по определенным наборам значений, чтобы соблюсти одно из ключевых свойств полинома Жегалкина: его значения должны совпадать с результатом исходного выражения для всех комбинаций входных переменных.

Шаги вычисления коэффициентов:

1. вводится начальный коэффициент A0 – значение выражения, когда все переменные равны нулю (false). В таблице истинности это верхняя строка,
2. расчет вклада каждой отдельной переменной. Берется разность по модулю 2 между значениями выражения для выборок с переменными по отдельности; сначала, где значение текущей переменной равно 1 (true), а остальных 0 (false), а затем, где все переменные равны 0 (false),
3. этот процесс повторяется для всех групп переменных (от одной до всех переменных выражения), пока не будут рассчитаны коэффициенты для всех мономов полинома Жегалкина.

Следующий алгоритм – минимизация булевых функций с использованием метода Квайна-МакКласки (Карты Карно). Он подходит для минимизации булевых функций, особенно когда число переменных больше четырех. Он состоит из нескольких этапов:

1. составляется таблица из строк таблицы, где выражение принимает значение 1 (true). Затем они представляются в виде чисел в двоичной системе,
2. далее идет разделение по группам в зависимости от количества 1 в числе. Например, числа 0001 и 0010 будут относится к одной группе, а число 1100 нет, так как имеет две единицы в записи,
3. происходит сравнение чисел в соседних выборках (рисунок 2.2). Если они отличаются ровно на одну позицию, например, 0001 и 0011, нужно их объединить в новую группу с отличительной чертой, которая размешается на месте различающихся позиций данных чисел, для чисел из примера это записи 00-1 и 00-1. Хочется отметить, что при повторе чисел можно оставить только одно из них в группе, а второе же стереть. Данный шаг нужно повторять до тех пор, пока не получим неприводимые выборки,
4. составление таблицы покрытий, которая показывает какие из неприводимых групп налаживаются на значения шага 1,
5. выбирается минимальный набор, покрывающий каждый из начальной выборки, с использованием метода исключений.

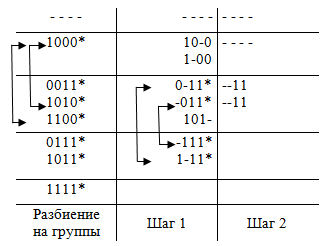


Рисунок 2.2 – Демонстрация 3 шагов алгоритма Квайна-МакКласки (Карты Карно).

При запуске программы открывается окно калькулятора. В нем будут находится строка ввода данных и кнопки, отвечающие за различные действия. Есть некоторые правила для взаимодействия с ними:

1. Вначале нужно ввести выражение. Кнопки, которые вы не можете нажать в данный момент из-за правил булевой логики будут неактивны. Для ввода выражения используются кнопки, представленные в калькуляторе. Среди кнопок есть кнопки для стирания последнего введенного элемента (операнда или операции) и стирания полностью всей строки.
2. После ввода выражения нажать кнопку Calculate. После ее нажатия она станет неактивной до того момента, пока вы не измените выражение на новое. Также будут произведены все алгоритмы булевого калькулятора и все функции вывода (таблица истинности, полином Жегалкина и т.д.) станут активными. При их нажатии будет открываться соответствующий файл .txt с результатом алгоритма.

При закрытии калькулятора нужно воспользоваться крестиком в правом верхнем углу окна калькулятора.

# 3. Проектирование программного средства

Вначале кнопки вычисления СДНФ, СКНФ, таблицы истинности, полинома Жегалкина, минимизации методом Квайна-МакКласки (Карты Карно) будут неактивны. Пользователю необходимо ввести исходное булево выражение, после чего нажать кнопку с значком “=”. Только после ее нажатия происходят вычисления всех функций и кнопки с выводом в формате .txt (СДНФ, таблицы истинности и т.д.) становятся доступными для нажатия. После того как пользователь выберет одну из них, перед ним откроется нужное ему представление. Формат выходного файла – .txt.

Рассматриваю далее структуру всех алгоритмов булевого калькулятора. На рисунке 3.1 изображена блок-схема генерации строк таблицы истинности. Подсчет количества строк таблицы путем побитового сдвига единицы влево на количество переменных, которые введет пользователь. Задание значения true или false переменной происходит засчет побитового сдвига вправо и операцией “and” c 1.

На рисунке 3.2 и 3.3 представлена блок-схема алгоритма вычисления результата выражения для каждой строки таблицы истинности. Основная идея данного алгоритма заключается в формировании польской записи выражения, которое ввел пользователь, и после ее чтение программой.

На рисунке 3.4 представлена блок-схема алгоритма создания СКНФ и СДНФ. Его реализация осуществляется после того, как будет реализована функция генерации строк таблицы истинности и вычисления результата для каждой из ее строк, так как основа данного алгоритма заключается в использовании готовой таблицы истинности для выражения.

На рисунке 3.5 изображена блок схема алгоритма создания полинома Жегалкина. Этот алгоритм реализуется после выполнения функции создания СДНФ и СКНФ. Определение того, влияет ли коэффициент на A i-ый элемент осуществляется за счёт операции “xor”, что равносильно определению операции сложения по модулю 2 (⊕).

На рисунке 3.6 представлена блок-схема алгоритма минимизации функции методом Квайна-МакКласки. Группой определенных минтермов является группа, в которой одинаковое количество 1 (значений true переменных). Импликанты – результаты сравнения соседних групп, где на месте различия, которое должно быть единственным среди двух минтермов, ставится “-“, а использованные минтермы убираются из массива для использования.

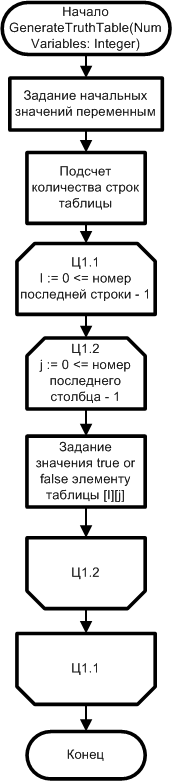


Рисунок 3.1 – Алгоритм генерации строк таблицы истинности

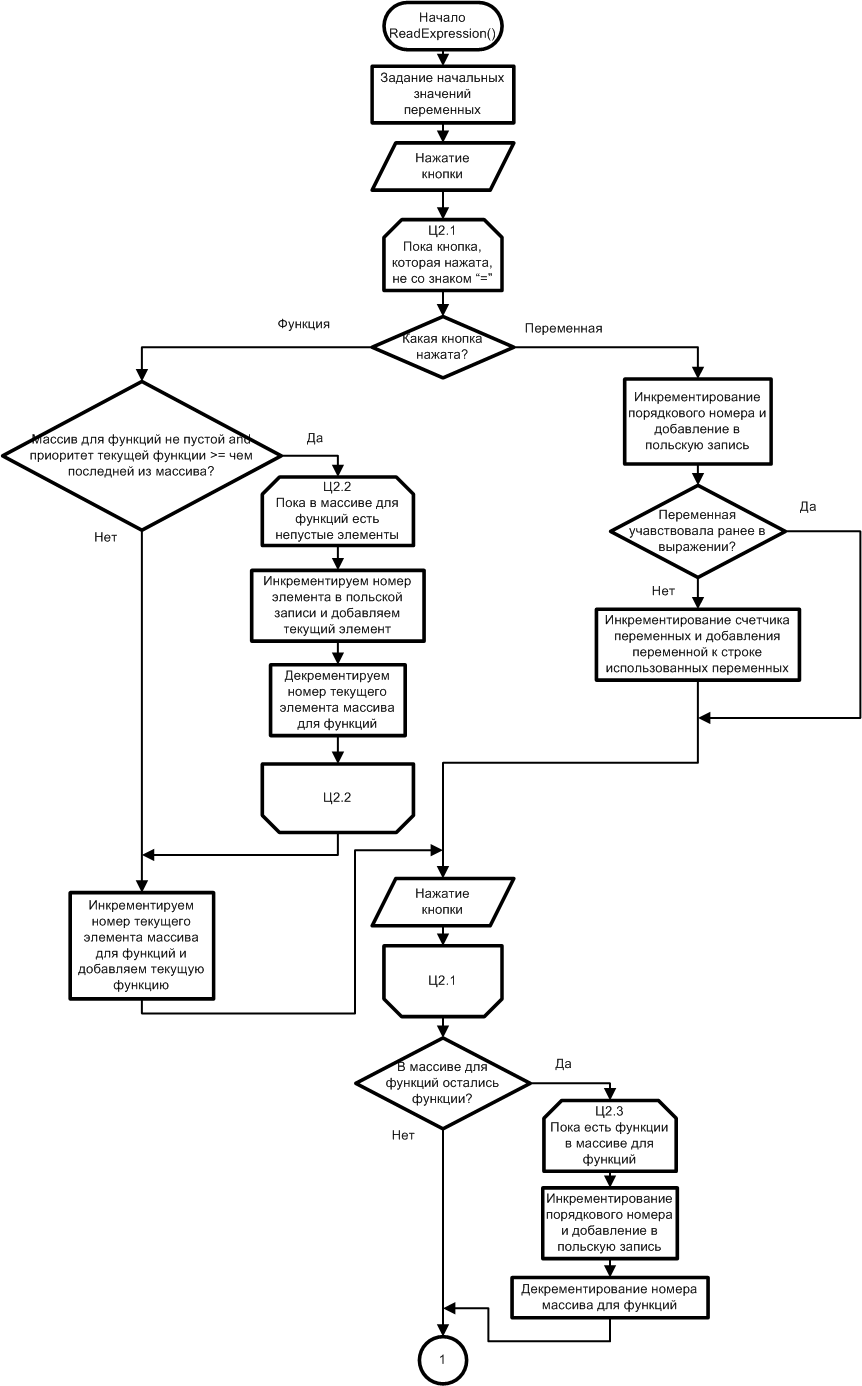


Рисунок 3.2 – Алгоритм вычисления результата выражения для каждой строки таблицы истинности (часть 1)

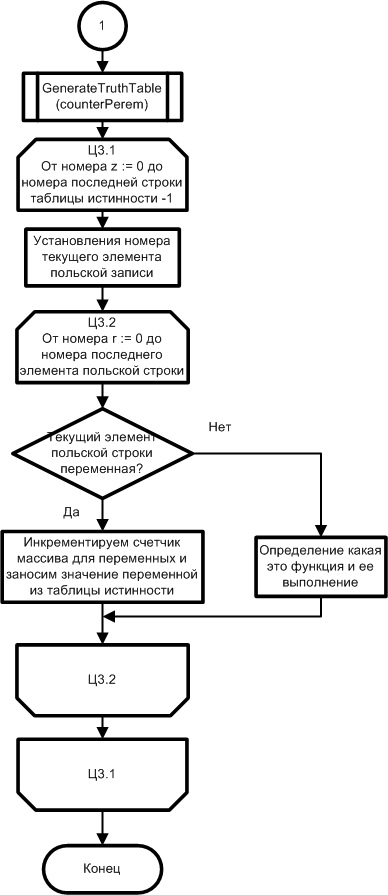


Рисунок 3.3 – Алгоритм вычисления результата выражения для каждой строки таблицы истинности (часть 2)

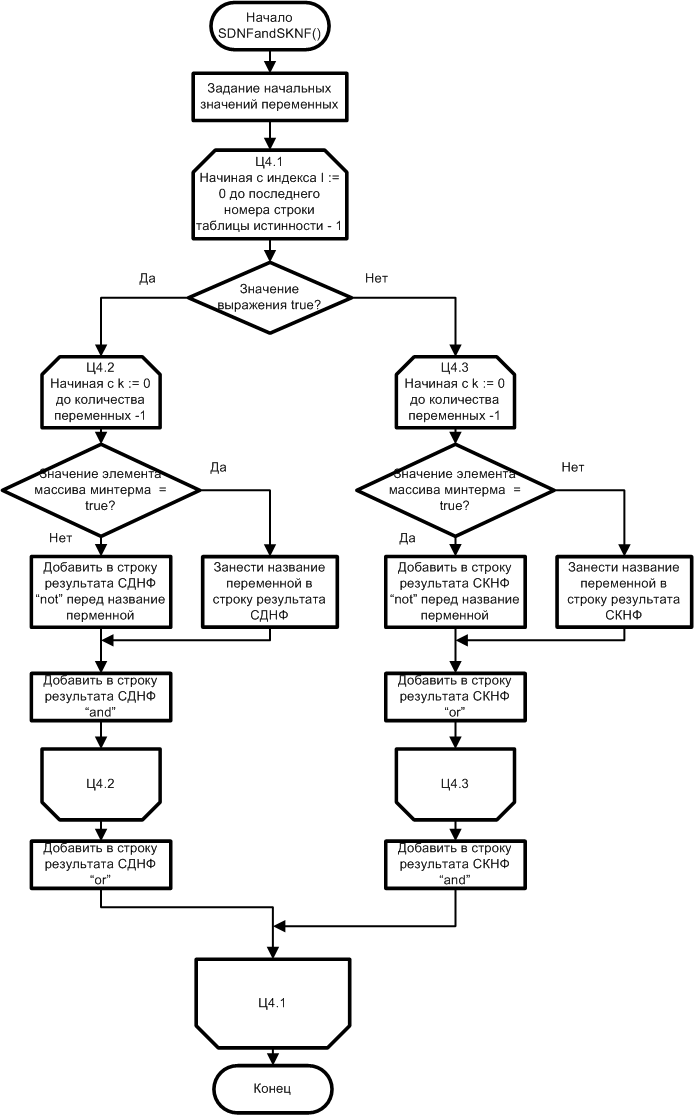


Рисунок 3.4 – Алгоритм создания СКНФ и СДНФ.

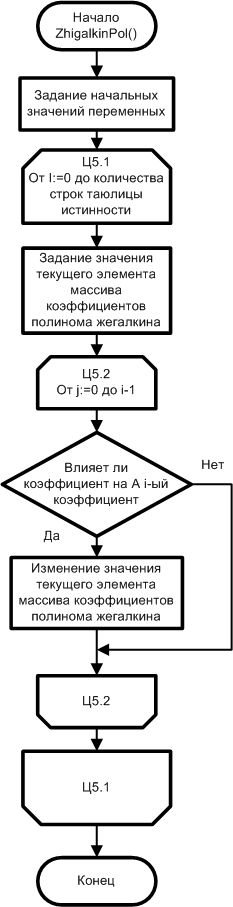


Рисунок 3.5 – Алгоритм создания полинома Жегалкина.

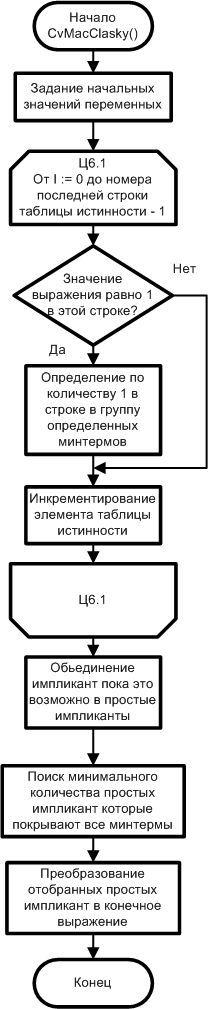


Рисунок 3.6 – Алгоритм минимизации функции методом Квайна-МакКласки (Карты Карно).

# 4. Создание (конструирование) программного средства

Разрабатываемое программное средство состоит из следующих модулей:

1. UnitMain.pas – модуль, который содержит в себе основной функционал формы программы (обработчики событий при нажатии на кнопки, открытие и закрытие формы и т.д.);
2. UnitTruthTable.pas – модуль, в котором реализованы функции и процедуры для создания таблицы истинности для выражения, введенного пользователем;
3. UnitSdnfSknf.pas – модуль, отвечающий за создание СДНФ и СКНФ;
4. UnitPolinomZh.pas – модуль, в котором реализован алгоритм получения полинома Жегалкина;
5. UnitMacClasky.pas – модуль, содержащий в себе процедуры и методы необходимые для реализации минимизации функции методом Квайна-МакКласки.

Далее приведены таблицы с описаниями подпрограмм данных модулей. Код всей программы находится в приложении А.

Таблица 4.1 – Описание подпрограмм UnitMain.pas

|  |  |  |
| --- | --- | --- |
| **Имя и тип под-**  **программы** | **Назначение подпрограммы** | **Параметры, назначение и возвращаемые значения** |
| procedure LockAfterBtnLetters; | Блокировка кнопок, которые нельзя нажимать после нажатия кнопок с именем переменных | Отсутствуют |
| procedure LockAfterBtnOperations; | Блокировка кнопок, которые нельзя нажимать после нажатия кнопок с названиями операций | Отсутствуют |
| procedure btnAClick | Обработчик события щелчка по кнопке с именем “A”. Ввод в строку Edit1 переменной с именем “A”. | (Sender: TObject);  Sender – объект работы процедуры (кнопка) |
| procedure btnBClick | Обработчик события щелчка по кнопке с именем “B”. Ввод в строку Edit1 переменной с именем “B”. | (Sender: TObject);  Sender – объект работы процедуры (кнопка) |
| procedure btnCClick | Обработчик события щелчка по кнопке с именем “C”. Ввод в строку Edit1 переменной с именем “C”. | (Sender: TObject);  Sender – объект работы процедуры (кнопка) |
| procedure btnDClick | Обработчик события щелчка по кнопке с именем “D”. Ввод в строку Edit1 переменной с именем “D”. | (Sender: TObject);  Sender – объект работы процедуры (кнопка) |

Таблица 4.1 – Продолжение

|  |  |  |
| --- | --- | --- |
| procedure btnAndClick | Обработчик события щелчка по кнопке с именем “AND”. Ввод в строку Edit1 операции с именем “AND”. | (Sender: TObject);  Sender – объект работы процедуры (кнопка) |
| procedure btnOrClick | Обработчик события щелчка по кнопке с именем “OR”. Ввод в строку Edit1 операции с именем “OR”. | (Sender: TObject);  Sender – объект работы процедуры (кнопка) |
| procedure btnXorClick | Обработчик события щелчка по кнопке с именем “XOR”. Ввод в строку Edit1 операции с именем “XOR”. | (Sender: TObject);  Sender – объект работы процедуры (кнопка) |
| procedure btnNotClick | Обработчик события щелчка по кнопке с именем “NOT”. Ввод в строку Edit1 операции с именем “NOT”. | (Sender: TObject);  Sender – объект работы процедуры (кнопка) |
| procedure btnMacClaskyClick | Обработчик события щелчка по кнопке с именем “MacClasky”. Вывод результата минимизации функции методом Квайна-МакКласки в виде открытия .txt файла | (Sender: TObject);  Sender – объект работы процедуры (кнопка) |
| procedure btnTruthTableClick | Обработчик события щелчка по кнопке с именем “Truth table”. Вывод результата создания таблицы истинности для выражения в виде открытия .txt файла | (Sender: TObject);  Sender – объект работы процедуры (кнопка) |
| procedure btnSdnfClick | Обработчик события щелчка по кнопке с именем “SDNF”. Вывод результата создания СДНФ для выражения в виде открытия .txt файла | (Sender: TObject);  Sender – объект работы процедуры (кнопка) |
| procedure btnSknfClick | Обработчик события щелчка по кнопке с именем “SKNF”. Вывод результата создания СКНФ для выражения в виде открытия .txt файла | (Sender: TObject);  Sender – объект работы процедуры (кнопка) |
| procedure btnPolinomZhClick | Обработчик события щелчка по кнопке с именем “Zhigalkin Polynomial”. Вывод результата создания полинома Жегалкина для выражения в виде открытия .txt файла | (Sender: TObject);  Sender – объект работы процедуры (кнопка) |

Таблица 4.1 – Продолжение

|  |  |  |
| --- | --- | --- |
| procedure btnCalcClick | Обработчик события щелчка по кнопке с именем “Calculate”. Запускает процедуры рассчетов всех функций для введенного выражения | (Sender: TObject);  Sender – объект работы процедуры (кнопка) |
| procedure FormCreate | Обработчик события начала включения программы (подготовка формы к использованию) | (Sender: TObject);  Sender – объект работы процедуры (кнопка) |
| procedure btnBackspaseClick | Обработчик события щелчка по кнопке с именем “Backspase”. Стирает запись после последней нажатой пользователем кнопки | (Sender: TObject);  Sender – объект работы процедуры (кнопка) |
| procedure btnCleanAllClick | Обработчик события щелчка по кнопке с именем “Clean ALL”. Очищает строку ввода выражения | (Sender: TObject);  Sender – объект работы процедуры (кнопка) |

Таблица 4.2 – Описание подпрограмм UnitTruthTable.pas

|  |  |  |
| --- | --- | --- |
| **Имя и тип под-**  **программы** | **Назначение подпрограммы** | **Параметры, назначение и возвращаемые значения** |
| procedure GenerateTruthTable; | Создание пустой таблицы истинности | (NumVariables: Integer; var TruthTable: TTruth; var NumRows: Integer);  NumVariables – количество переменных в выражении;  TruthTable – таблица истинности;  NumRows – количество строк таблицы истинности. |
| procedure FillTruthTable | Заполнение таблицы истинности используя алгоритм сведения выражения к обратной польской записи | (Edit1: TEdit);  Edit1 – текстовое поле с выражением |
| procedure OpenTextFile | Открытие файла | (const FileName: string) FileName – имя файла для открытия. |

Таблица 4.3 – Описание подпрограмм UnitSdnfSknf.pas

|  |  |  |
| --- | --- | --- |
| **Имя и тип под-**  **программы** | **Назначение подпрограммы** | **Параметры, назначение и возвращаемые значения** |
| procedure SDNFandSKNF; | Создание СДНФ и СКНФ и запись в соответствующий файл | (TruthTable: TTruth; StrOfPerem: String);  TruthTable – таблица истинности;  StrOfPerem – строка, содержащая все переменные в выражении |

Таблица 4.4 – Описание подпрограмм UnitPolinomZh.pas

|  |  |  |
| --- | --- | --- |
| **Имя и тип под-**  **программы** | **Назначение подпрограммы** | **Параметры, назначение и возвращаемые значения** |
| procedure ZhigalkinPol; | Создание полинома Жегалкина и запись в соответствующий файл | (TruthTable: TTruth; StrOfPerem: String);  TruthTable – таблица истинности;  StrOfPerem – строка, содержащая все переменные в выражении |

Таблица 4.5 – Описание подпрограмм UnitMacClasky.pas

|  |  |  |
| --- | --- | --- |
| **Имя и тип под-**  **программы** | **Назначение подпрограммы** | **Параметры, назначение и возвращаемые значения** |
| procedure ComapisonMinterms; | Сравнение минтермов для минимизации функции методом Квайна-МакКласки с проверкой на прайм импликанту и добавлением в соответствующий массив данных. | (FirstMinterm, SecondMinterm: RMinterm;  var CounterCovering: Integer;  var CheckPrime: Boolean;  var MasMinterm2: TMinterms; var ChangesElements: Boolean;  var CounterMas2: Integer;  var StringOfMinterms: String);  FirstMinterm – первый минтерм сравнения;  SecondMinterm – второй минтерм сравнения;  CounterCovering – подсчет покрытия минтермом вариантов значений переменных;  CheckPrime – проверка на прайм импликанту;  MasMinterm2 – массив минтермов после очередного шага алгоритма;  ChangesElements – проверка на то, есть ли еще не прайм импликанты;  CounterMas2 – счетчик элементов второго массива;  StringOfMinterms – строка обработанных минтермов для проверки того, повторяются ли они. |
| procedure GroupingMinterms; | Группирует минтермы по количеству 1 (Truth) в них и считает количество групп. | (var MasMinterm1: TMinterms; var CounterGroups: Integer);  MasMinterm1 – массив необработанных минтермов; |

Таблица 4.5 – Продолжение

|  |  |  |
| --- | --- | --- |
| Procedure CvMacClasky; | Минимизация функции методом Квайна-МакКласки | TruthTable – таблица истинности |

# 5. Тестирование, проверка работоспособности и анализ полученных результатов

Таблица 5.1 – Тестирование программного средства

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Специфика тестирования | Номер теста | Вводимые данные | Ожидаемый результат | Полученный результат |
| Запуск программы | 1 | - | Отображение основного меню с некоторыми отключенными кнопками | Тест пройден |
| Проверка правильности обработчика событий для кнопки | 2 | Нажатие кнопки “A”, когда она активна | Добавление символа “A” в строку ввода | Тест пройден |
| Проверка правильности обработчика событий для кнопки | 3 | Нажатие кнопки “B”, когда она активна | Добавление символа “B” в строку ввода | Тест пройден |
| Проверка правильности обработчика событий для кнопки | 4 | Нажатие кнопки “C”, когда она активна | Добавление символа “C” в строку ввода | Тест пройден |
| Проверка правильности обработчика событий для кнопки | 5 | Нажатие кнопки “D”, когда она активна | Добавление символа “D” в строку ввода | Тест пройден |

Таблица 5.1 – Продолжение

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Проверка правильности обработчика событий для кнопки | 6 | Нажатие кнопки “AND”, когда она активна | Добавление символа “AND” в строку ввода | Тест пройден |
| Проверка правильности обработчика событий для кнопки | 7 | Нажатие кнопки “OR”, когда она активна | Добавление символа “OR” в строку ввода | Тест пройден |
| Проверка правильности обработчика событий для кнопки | 8 | Нажатие кнопки “XOR”, когда она активна | Добавление символа “XOR” в строку ввода | Тест пройден |
| Проверка правильности обработчика событий для кнопки | 9 | Нажатие кнопки “NOT”, когда она активна | Добавление символа “NOT” в строку ввода | Тест пройден |
| Проверка правильности доступа нажатия кнопок | 10 | Нажатие кнопки “AND”, когда она активна | Кнопки со всеми операциями кроме “NOT” и переменных блокируются | Тест пройден |
| Проверка правильности доступа нажатия кнопок | 11 | Нажатие кнопки “OR”, когда она активна | Кнопки со всеми операциями кроме “NOT” и переменных блокируются | Тест пройден |
| Проверка правильности доступа нажатия кнопок | 12 | Нажатие кнопки “XOR”, когда она активна | Кнопки со всеми операциями кроме “NOT” и переменных блокируются | Тест пройден |
| Проверка правильности доступа нажатия кнопок | 13 | Нажатие кнопки “NOT”, когда она активна | Кнопки со всеми операциями кроме ввода операнда блокируются | Тест пройден |

Таблица 5.1 – Продолжение

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Проверка правильности доступа нажатия кнопок | 14 | Нажатие кнопки “A”, когда она активна | Кнопки со всеми операциями кроме операций блокируются | Тест пройден |
| Проверка правильности доступа нажатия кнопок | 15 | Нажатие кнопки “B”, когда она активна | Кнопки со всеми операциями кроме операций блокируются | Тест пройден |
| Проверка правильности доступа нажатия кнопок | 16 | Нажатие кнопки “C”, когда она активна | Кнопки со всеми операциями кроме операций блокируются | Тест пройден |
| Проверка правильности доступа нажатия кнопок | 17 | Нажатие кнопки “D”, когда она активна | Кнопки со всеми операциями кроме операций блокируются | Тест пройден |
| Проверка обработчика событий Calculate | 18 | Нажатие кнопки “Calculate”, когда она активна | Кнопки со всеми операциями кроме операций блокируются и открываются функции таблицы истинности и т.д. | Тест пройден |
| Проверка корректности файлов | 19 | Нажатие кнопки “Truth table”, когда она активна | Открытие .txt файла с таблицей истинности. Таблица совпадает с таблицей выражения, полученной из математической программы | Тест пройден |
| Проверка корректности файлов | 20 | Нажатие кнопки “SDNF”, когда она активна | Открытие .txt файла с СДНФ. Строка совпадает с строкой, полученной из математической программы | Тест пройден |

Таблица 5.1 – Продолжение

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Проверка корректности файлов | 21 | Нажатие кнопки “SKNF”, когда она активна | Открытие .txt файла с СКНФ. Строка совпадает с строкой, полученной из математической программы | Тест пройден |
| Проверка корректности файлов | 22 | Нажатие кнопки “Zhigalkin Polynomial”, когда она активна | Открытие .txt файла с полиномом Жегалкина. Совпадает с полиномом, полученным из математической программы | Тест пройден |
| Проверка корректности файлов | 23 | Нажатие кнопки “MacClasky”, когда она активна | Открытие .txt файла с минимизированной функцией. Совпадает с выражением, полученным из математической программы | Тест пройден |
| Проверка правильности удаления | 24 | Нажатие кнопки “Backspace”, когда она активна | Удаляет последнее введенное выражение и блокирует некоторые кнопки | Тест пройден |
| Проверка правильности удаления | 25 | Нажатие кнопки “CleanALL”, когда она активна | Очищает строку и делает активными такие же кнопки, что и при запуске программы | Тест пройден |

Программное средство работает корректно. Формы правильно отображают данные, взаимодействие с кнопками происходит по заданной логике, корректно реагируя на действия пользователя. Все окна открываются полноценно, взаимодействие с ними соответствует ожиданию.

# 6. Руководство по установке и использованию

Установить и использовать программу в том виде, в котором она предусмотрена в курсовой работе, можно только на разных версиях операционной системы (ОС) Windows, для остальных же (Linux, MacOS и др.) необходимо изменять файлы вручную в формат, предусмотренный системой (.deb, .rpm, .dmg и др.). В данном руководстве рассматривается установка только для версий ОС Windows (.exe), однако использование продукта одинаковое для всех других ОС.

Вначале необходимо скачать ZIP-архив с названием “Калькулятор булевых выражений”. В названии в конце может быть формат .zip. Далее необходимо распаковать архив. Для этого необходимо зайти в папку, в которой непосредственно находится сам архив, который был скачен. Далее кликнуть по нему левой кнопкой мыши, после чего кликнуть правой. Откроется меню с выбором опций. На разных версиях ОС меню может отличаться, я рассматриваю на примере Windows 11 (Рисунок 6.1). Если у вас другая версия ОС, то различия никак не влияют на следующие шаги по руководству. Также у меня включено сохранение на Google Drive, если у вас оно не включено, то иконки файлов и папок будут без кругов в нижнем левом углу. Это ни на что не влияет, кроме как вида иконки.

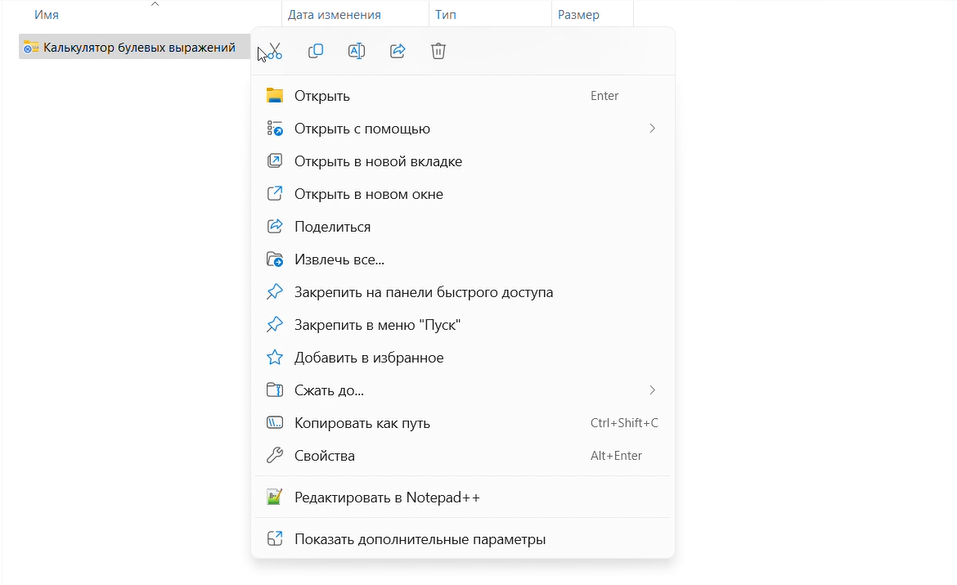


Рисунок 6.1 – Опции, применимые к архиву

Теперь необходимо распаковать архив. Это можно сделать двумя способами:

1. Скачать стороннюю программу, которой необходимо предоставить доступ к своим файлам, после разобраться как взаимодействовать с ней и распаковать архив.
2. Воспользоваться встроенной опцией Windows.

Воспользуемся вторым способом, так как опция распаковки архива формата .zip без сторонних программ (ПО) есть в любой версии ОС Windows. Для этого необходимо выбрать опцию “Извлечь все…”, на которую наведен указатель мыши на рисунке 6.2.

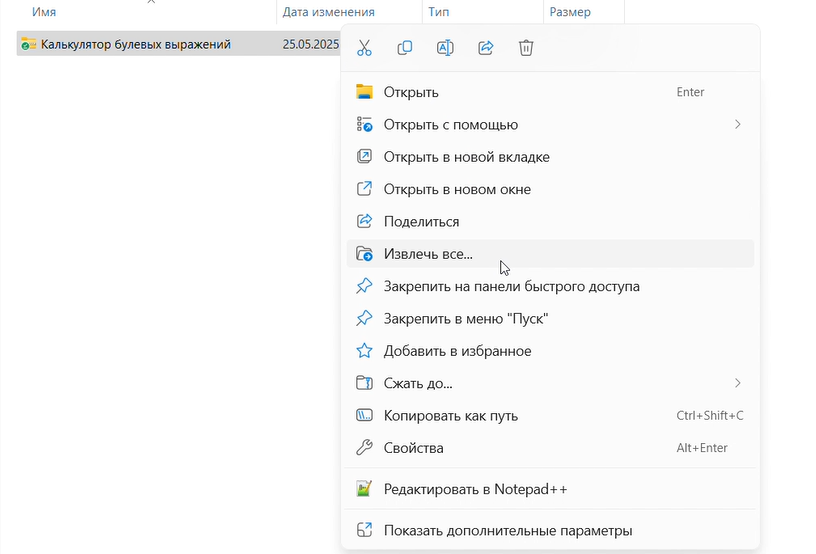


Рисунок 6.2 – Опция для распаковки ZIP-архива

После выбора данной опции откроется окно распаковки (Рисунок 6.3), в которой синим будет подсвечен путь, в который будет распакован архив. По умолчанию он распакует архив в папку, в которой вы сейчас находитесь, поэтому можно оставить путь неизменным, однако вы можете нажать кнопку обзор справа и выбрать папку самостоятельно.

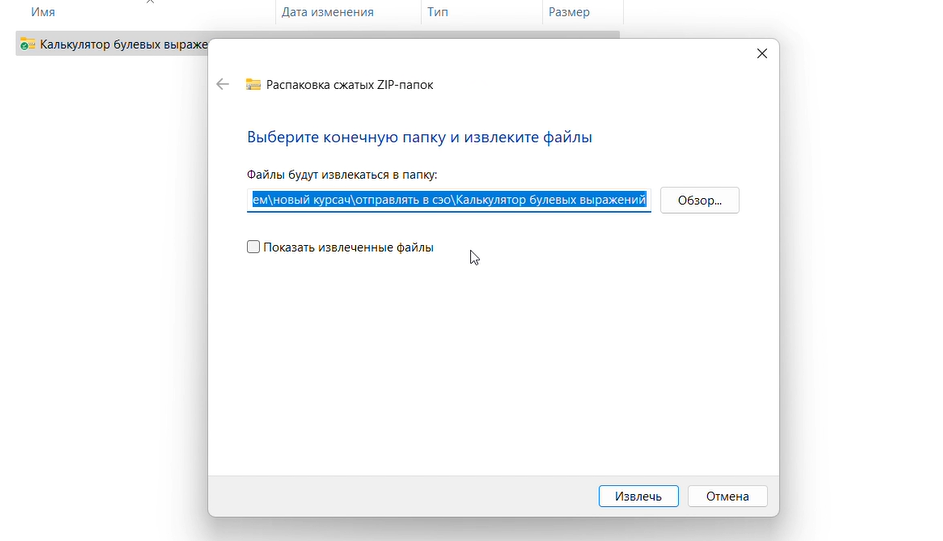


Рисунок 6.3 – Окно опции “Извлечь все…”

В руководстве путь оставлен по умолчанию. После завершения распаковки, рядом с ZIP-архивом появится новая папка с таким же именем (Рисунок 6.4), однако ее иконка будет видоизменена на ту, которая используется для папок в вашей версии ОС.

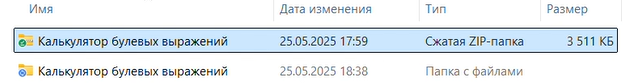


Рисунок 6.4 – Файлы: сверху – ZIP-архив, снизу – файл, результат распаковки

Теперь необходимо открыть данный папку с помощью двойного клика левой кнопкой мыши по ней. Теперь будет отображено ее содержимое. Среди всех файлов вам необходимо найти самый первый по алфавиту из них с названием “Calculator” (Рисунок 6.5). В названии может присутствовать расширение .exe. У меня включена сортировка файлов по алфавиту, поэтому он у меня сверху.

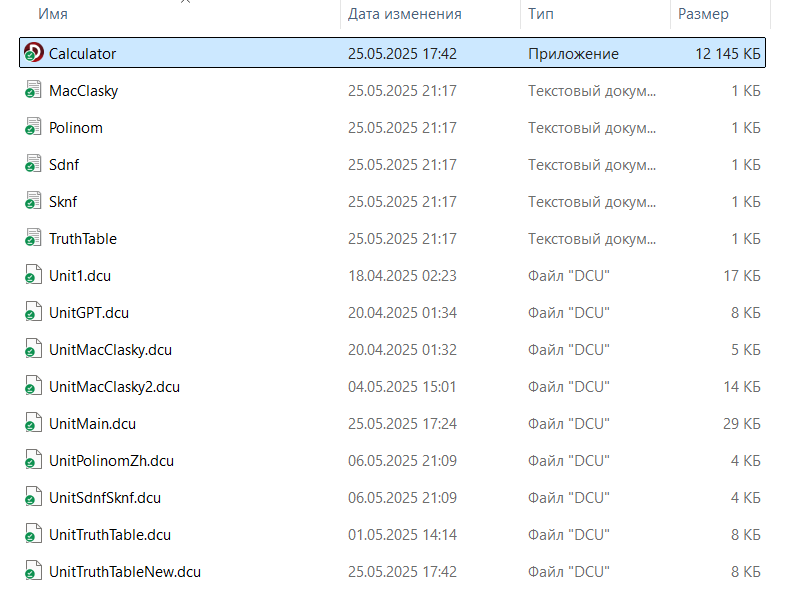


Рисунок 6.5 – Содержимое папки “Калькулятор булевых выражений” с выделенным .exe файлом

Далее необходимо дважды кликнуть по нему левой кнопкой мыши, откроется главное окно программы, которое продемонстрировано на рисунке 6.6. В верхнем левом углу подписано название приложения. В правом верхнем – кнопка закрытия приложения. В верхней строке находятся опции “Файл”, “Инструменты” и “Справка”. Ниже находится строка ввода, ввод осуществляется только с помощью кнопок, которые находятся на синем фоне.

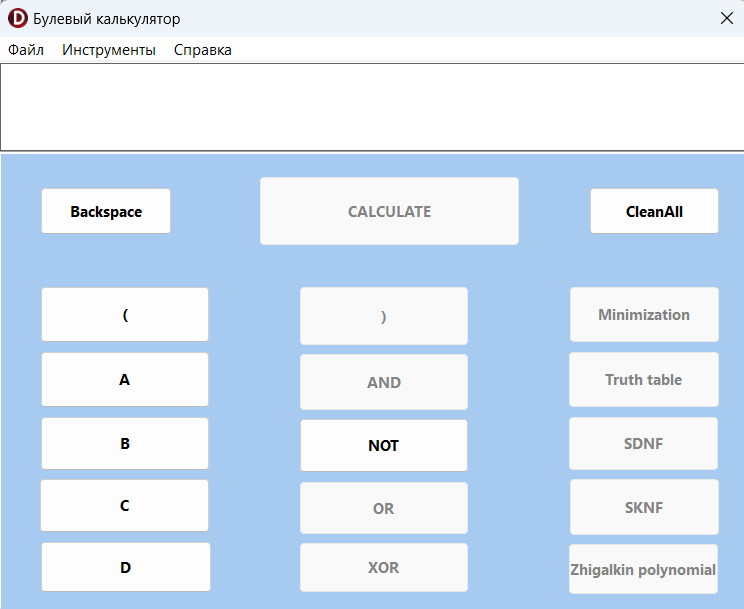


Рисунок 6.6 – Стартовое окно калькулятора

Слева находятся кнопка, отвечающая за стирание последнего введенного символа “Backspace”, кнопка открывающей скобки “(“, а также кнопки ввода переменных: “A”, “B”, “C” и “D”. В середине находятся кнопка подсчета выражения “CALCULATE”, под ней кнопка закрывающей скобки “)”, а также кнопки операций: “AND”, “NOT”, “OR” и “XOR”. Справа располагаются кнопка очистки всего ввода “CleanAll”, а под ней кнопки всевозможных результатов подсчета выражения: “Minimization”, “Truth table”, “SDNF”, “SKNF” и “Zhegalkin polynomial”.

При запуске программы некоторые кнопки имеют серый оттенок, который означает, что пользователь не может на данный момент нажать эти кнопки.

Рассмотрю, что происходит при нажатии различных кнопок. При нажатии кнопки “Файл” откроется подменю с опциями (Рисунок 6.7). В нем находятся две опции:

1. “Выход” – опция, при нажатии которой приложение закрывается. Также ее можно активировать с помощью нажатия клавиши “Esc”
2. “Сохранить как” – опция, при нажатии которой открывается проводник и можно выбрать файлы, которые являются результатами работы калькулятора, для сохранения. Сочетание клавиш: “Ctrl + S”.

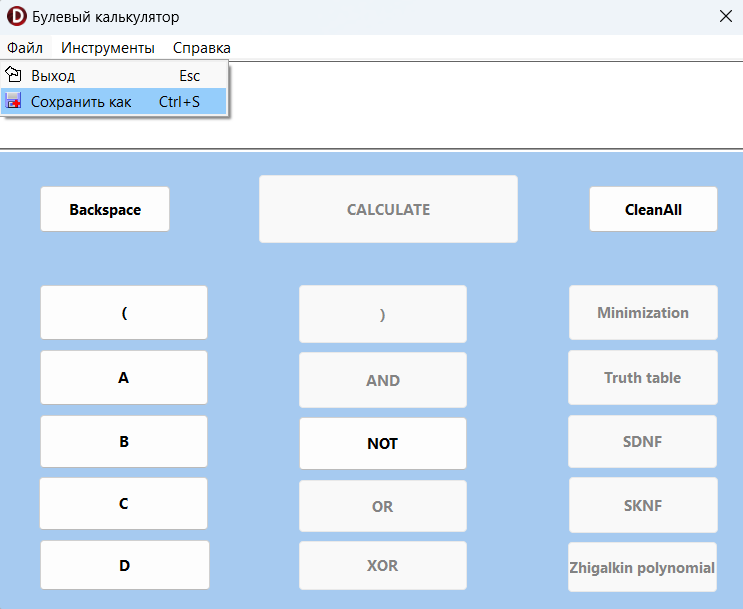


Рисунок 6.7 – Подменю опции “Файл”

При нажатии на “Инструменты” появиться подменю (Рисунок 6.8) с следующими опциями:

1. “Вычислить” – опция, эквивалентная по действию кнопке “CALCULATE”, выполняет подсчет выражения, которое ввел пользователь и которое будет отображено в строке ввода. Делает активными кнопки “Minimization”, “Truth table”, “SDNF”, “SKNF” и “Zhigalkin polynomial”. Горячая клавиша: “Enter”.
2. “Минимизировать” – опция, эквивалентная по действию кнопке “Minimization”, открывает файл .txt, в котором записан результат минимизации выражения, введенного пользователем. Опция активна только после нажатия кнопки “CALCULATE” либо опции вычислить. Сочетание клавиш: “Alt + M”.
3. “Таблица истинности” – опция, эквивалентная по действию кнопке “Truth table”, открывает файл .txt, в котором записана таблица истинности выражения, введенного пользователем. Опция активна только после нажатия кнопки “CALCULATE” либо опции вычислить. Сочетание клавиш: “Alt + T”.
4. “СДНФ” – опция, эквивалентная по действию кнопке “SDNF”, открывает файл .txt, в котором записана совершенная дизъюнктивная нормальная форма выражения, введенного пользователем. Опция активна только после нажатия кнопки “CALCULATE” либо опции вычислить. Сочетание клавиш: “Alt + D”.
5. “СКНФ” – опция, эквивалентная по действию кнопке “SKNF”, открывает файл .txt, в котором записана совершенная конъюнктивная нормальная форма выражения, введенного пользователем. Опция активна только после нажатия кнопки “CALCULATE” либо опции вычислить. Сочетание клавиш: “Alt + K”.
6. “Полином” – опция, эквивалентная по действию кнопке “Zhigalkin polynomial”, открывает файл .txt, в котором полином Жегалкина для выражения, введенного пользователем. Опция активна только после нажатия кнопки “CALCULATE” либо опции вычислить. Сочетание клавиш: “Alt + P”.

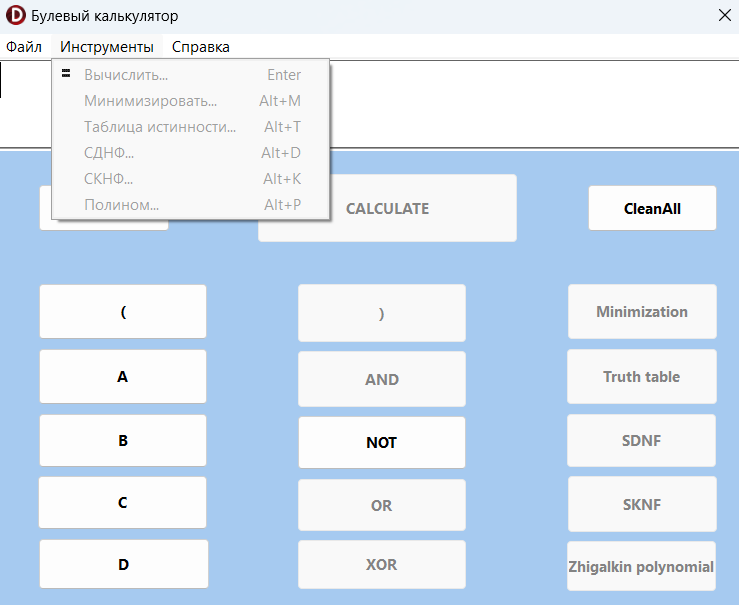


Рисунок 6.8 – Подменю опции “Инструменты”

При нажатии опции “Справка” выпадает подменю (Рисунок 6.9) с следующими опциями:

1. “О программе” – опция, при выборе которой появится всплывающее окно с краткими сведениями о том, что такое калькулятор булевых выражений.
2. “Помощь” – опция, при выборе которой появится всплывающее окно с краткими правилами пользования калькулятором.

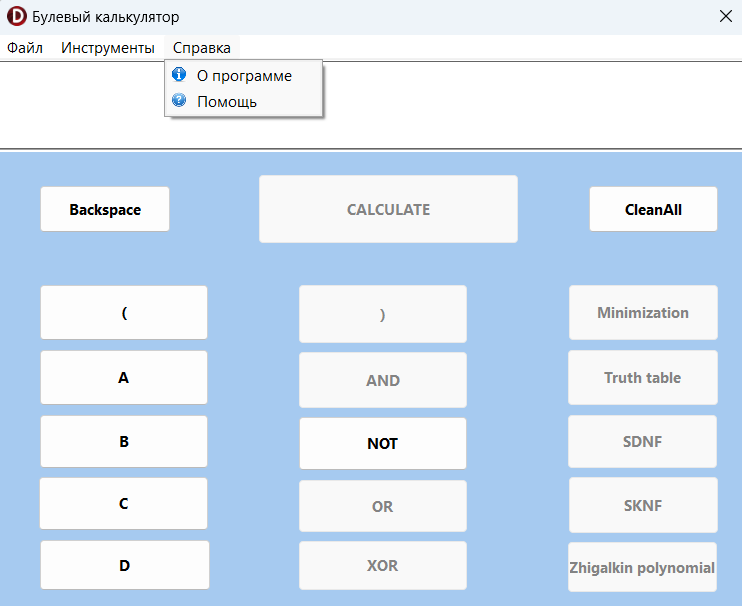


Рисунок 6.9 – Подменю опции “Справка”

Кнопки “Calculate”, “Minimization”, “Truth table”, “SDNF”, “SKNF” и “Zhigalkin polynomial” имеют функции аналогичные тем, которые описаны выше в опциях подменю “Инструменты”. “Backspace” и “CleanAll” – стереть последний ввод и все выражение соответственно. Кнопки “A”, “B”, “C” и “D” добавляют в строку ввода переменные с их названиями соответственно. “AND”, “NOT”, “OR” и “XOR” добавляют операции с их названиями соответственно.

Правила ввода выражения такие же, как и в булевой алгебре:

1. Открытая скобка может ставится вначале либо после операций (and, not и т.д), закрытая скобка ставится после переменной (A, B и т.д.). Вначале ввода также может быть переменная (A, B и т.д.).
2. После операндов (A, B и т.д.) вводится либо двуместная операция (and, or, xor) либо активируется подсчет выражения (кнопка “CALCULATE” либо опция “Вычислить” в подменю “Инструменты”).
3. После двуместных операндов (and, or, xor) можно ввести либо переменную (A, B и т.д.) либо одноместный операнд not.
4. После not всегда идет операнд (A, B и т.д.) либо еще один not.
5. Выражение всегда заканчивается либо операндом (A, B и т.д.) либо закрытой скобкой (если до этого была незакрытая открытая скобка и больше таких в выражении не имеется).

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы была достигнута основная цель – разработка программного средства «Калькулятор логических выражений с возможностью минимизации булевых функций». В процессе проектирования были последовательно выполнены этапы анализа предметной области, изучения существующих аналогов, постановки функциональных требований, алгоритмического и структурного проектирования, реализации программы на языке Delphi, а также её тестирования и анализа результатов.

Программа предоставляет пользователю удобный графический интерфейс и широкий набор функций, включая построение таблицы истинности, формирование СДНФ и СКНФ, построение полинома Жегалкина и минимизацию логических выражений с использованием метода Квайна-МакКласки. Проведённые тесты подтвердили корректность работы всех реализованных функций, соответствие логики обработки заданным алгоритмам и удобство взаимодействия с интерфейсом.

Разработанный калькулятор может быть полезен в учебной среде как вспомогательный инструмент для изучения булевой алгебры и логических преобразований. Также он может служить основой для дальнейшего развития, расширения функционала и адаптации под другие платформы.

Таким образом, все поставленные задачи были успешно выполнены, а программное средство показало свою эффективность, надёжность и практическую применимость. СПИСОК ЛИТЕРАТУРЫ

[1] [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://boolean-calculator.ru/>.

[2] [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://programforyou.ru/calculators/postroenie-tablitci-istinnosti-sknf-sdnf>.

[3] Сальников В., Построение систем принятия решений на основе операций алгебры логики / В. Сальников // Ученые записки Санкт-Петербургского имени В.Б. Бобкова филиала Российской таможенной академии. 2019. № 2 (70). С. 101–107

[4] [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://neerc.ifmo.ru/wiki/index.php?title=%D0%94%D0%9D%D0%A4>

[5] [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://neerc.ifmo.ru/wiki/index.php?title=%D0%9A%D0%9D%D0%A4>

[6] [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D1%81%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BE%D1%87%D0%BD%D0%BE%D0%B9_%D1%81%D1%82%D0%B0%D0%BD%D1%86%D0%B8%D0%B8>

[7] Андулько Д.С., Алгоритм формирования функций алгебры логики из таблицы истинности / Aндулько Д.С. // Орловский государственный университет им. И. С. Тургенева Международный научный журнал “Вестник науки” Июнь 2024 г. №6 (75) Том 2.

# ПРИЛОЖЕНИЕ А

UnitMain:

unit UnitMain;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.ExtCtrls, UnitTruthTable, UnitSdnfSknf,

UnitPolinomZh, UnitMacClasky2;

type

TMasBuf = array of String[255];

TForm1 = class(TForm)

Panel1: TPanel;

Edit1: TEdit;

Panel2: TPanel;

btnA: TButton;

btnB: TButton;

btnC: TButton;

btnD: TButton;

btnAnd: TButton;

btnNot: TButton;

btnOr: TButton;

btnXor: TButton;

btnTruthTable: TButton;

btnCalc: TButton;

btnCleanAll: TButton;

btnBackspase: TButton;

btnSdnf: TButton;

btnSknf: TButton;

btnPolinomZh: TButton;

btnMacClasky: TButton;

procedure LockAfterBtnLetters;

procedure LockAfterBtnOperations;

procedure btnMacClaskyClick(Sender: TObject);

procedure btnAClick(Sender: TObject);

procedure btnBClick(Sender: TObject);

procedure btnCClick(Sender: TObject);

procedure btnDClick(Sender: TObject);

procedure btnAndClick(Sender: TObject);

procedure btnOrClick(Sender: TObject);

procedure btnXorClick(Sender: TObject);

procedure btnNotClick(Sender: TObject);

procedure btnBackspaseClick(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure btnCleanAllClick(Sender: TObject);

procedure btnCalcClick(Sender: TObject);

procedure btnTruthTableClick(Sender: TObject);

procedure btnSdnfClick(Sender: TObject);

procedure btnSknfClick(Sender: TObject);

procedure btnPolinomZhClick(Sender: TObject);

private

//LastButtonText: string[255];

InputBuffer: TMasBuf;

{ Private declarations }

public

{ Public declarations }

end;

var

Form1: TForm1;

implementation

{$R \*.dfm}

procedure TForm1.LockAfterBtnLetters;

begin

btnPolinomZh.Enabled := false;

btnCalc.Enabled := true;

btnTruthTable.Enabled := false;

btnA.Enabled := false;

btnB.Enabled := false;

btnC.Enabled := false;

btnD.Enabled := false;

btnNot.Enabled := false;

btnAnd.Enabled := true;

btnOr.Enabled := true;

btnXor.Enabled := true;

btnSdnf.Enabled := false;

btnSknf.Enabled := false;

end;

procedure TForm1.LockAfterBtnOperations;

begin

btnPolinomZh.Enabled := false;

btnCalc.Enabled := false;

btnTruthTable.Enabled := false;

btnAnd.Enabled := false;

btnOr.Enabled := false;

btnXor.Enabled := false;

btnNot.Enabled := true;

btnA.Enabled := true;

btnB.Enabled := true;

btnC.Enabled := true;

btnD.Enabled := true;

btnSdnf.Enabled := false;

btnSknf.Enabled := false;

end;

procedure TForm1.btnMacClaskyClick(Sender: TObject);

begin

CvMacClasky(TruthTable);

end;

procedure TForm1.btnAClick(Sender: TObject);

begin

LockAfterBtnLetters;

SetLength(InputBuffer, Length(InputBuffer) + 1);

InputBuffer[Length(InputBuffer) - 1] := 'A';

Edit1.Text := Edit1.Text + 'A';

end;

procedure TForm1.btnAndClick(Sender: TObject);

begin

LockAfterBtnOperations;

SetLength(InputBuffer, Length(InputBuffer) + 1);

InputBuffer[Length(InputBuffer) - 1] := ' and ';

Edit1.Text := Edit1.Text + ' and ';

end;

procedure TForm1.btnBClick(Sender: TObject);

begin

LockAfterBtnLetters;

SetLength(InputBuffer, Length(InputBuffer) + 1);

InputBuffer[Length(InputBuffer) - 1] := 'B';

Edit1.Text := Edit1.Text + 'B';

end;

procedure TForm1.btnCalcClick(Sender: TObject);

begin

btnCalc.Enabled := false;

FillTruthTable(Edit1);

btnTruthTable.Enabled := true;

SDNFandSKNF(TruthTable, StrOfPerem);

btnSdnf.Enabled := true;

btnSknf.Enabled := true;

btnPolinomZh.Enabled := true;

ZhigalkinPol(TruthTable, StrOfPerem);

end;

procedure TForm1.btnCClick(Sender: TObject);

begin

LockAfterBtnLetters;

SetLength(InputBuffer, Length(InputBuffer) + 1);

InputBuffer[Length(InputBuffer) - 1] := 'C';

Edit1.Text := Edit1.Text + 'C';

end;

procedure TForm1.btnCleanAllClick(Sender: TObject);

begin

Edit1.Text := '';

SetLength(InputBuffer, 0);

LockAfterBtnOperations;

end;

procedure TForm1.btnDClick(Sender: TObject);

begin

LockAfterBtnLetters;

SetLength(InputBuffer, Length(InputBuffer) + 1);

InputBuffer[Length(InputBuffer) - 1] := 'D';

Edit1.Text := Edit1.Text + 'D';

end;

procedure TForm1.btnNotClick(Sender: TObject);

begin

LockAfterBtnOperations;

SetLength(InputBuffer, Length(InputBuffer) + 1);

InputBuffer[Length(InputBuffer) - 1] := 'not ';

Edit1.Text := Edit1.Text + 'not ';

end;

procedure TForm1.btnOrClick(Sender: TObject);

begin

LockAfterBtnOperations;

SetLength(InputBuffer, Length(InputBuffer) + 1);

InputBuffer[Length(InputBuffer) - 1] := ' or ';

Edit1.Text := Edit1.Text + ' or ';

end;

procedure TForm1.btnPolinomZhClick(Sender: TObject);

begin

OpenTextFile('Polinom.txt');

end;

procedure TForm1.btnSdnfClick(Sender: TObject);

begin

OpenTextFile('Sdnf.txt');

end;

procedure TForm1.btnSknfClick(Sender: TObject);

begin

OpenTextFile('Sknf.txt');

end;

procedure TForm1.btnTruthTableClick(Sender: TObject);

begin

OpenTextFile('TruthTable.txt');

end;

procedure TForm1.btnXorClick(Sender: TObject);

begin

LockAfterBtnOperations;

SetLength(InputBuffer, Length(InputBuffer) + 1);

InputBuffer[Length(InputBuffer) - 1] := ' xor ';

Edit1.Text := Edit1.Text + ' xor ';

end;

procedure TForm1.btnBackspaseClick(Sender: TObject);

var

tempStrEdit: string[255];

tempLastTextInput: string[255];

begin

if length(InputBuffer) <> 0 then

begin

tempStrEdit := Edit1.Text; //выражение которое введено

tempLastTextInput := InputBuffer[Length(InputBuffer) - 1]; //последний введенный текст с кнопки

Delete(tempStrEdit, Length(Edit1.Text) - Length(tempLastTextInput) + 1, Length(tempLastTextInput));

Edit1.Text := tempStrEdit;

if Length(Edit1.Text) = 0 then

begin

LockAfterBtnOperations;

end

else if (tempLastTextInput = ' and ') or (tempLastTextInput = ' or ') or (tempLastTextInput = ' xor ') then

begin

LockAfterBtnLetters;

btnCalc.Enabled := true;

end

else

begin

LockAfterBtnOperations;

end;

SetLength(InputBuffer, Length(InputBuffer) - 1);

end;

end;

procedure TForm1.FormCreate(Sender: TObject);

begin

LockAfterBtnOperations;

end;

end.

UnitTruthTable:

unit UnitTruthTable;

interface

uses

ShellApi, SysUtils, Vcl.StdCtrls, Windows;

type

TTruth = array of array of Boolean;

PeremMAS = array of String;

FuncMAS = array of Integer;

const

allFunc: array [1..4] of String = ('not', 'and', 'or', 'xor');

var

TruthTable: TTruth;

BooleanFunc: String = 'not and or xor'; //доступные операции

ValueOfFunc: String[4]; //приоритет операций

StrOfPerem: String; //все переменные в выражении

NumRows: Integer; //количество строк таблицы

procedure GenerateTruthTable(NumVariables: Integer; var TruthTable: TTruth; var NumRows: Integer);

procedure FillTruthTable(Edit1: TEdit);

procedure OpenTextFile(const FileName: string);

implementation

procedure OpenTextFile(const FileName: string);

begin

ShellExecute(0, 'open', PChar(FileName), nil, nil, SW\_SHOWNORMAL);

end;

procedure GenerateTruthTable(NumVariables: Integer; var TruthTable: TTruth; var NumRows: Integer);

var

i, j: Integer;

begin

// Количество строк в таблице истинности = 2^NumVariables

NumRows := 1 shl NumVariables; // 2^NumVariables

SetLength(TruthTable, NumRows, NumVariables + 1); //NumVariables

// Генерация всех комбинаций

for i := 0 to NumRows - 1 do

for j := 0 to NumVariables - 1 do

TruthTable[i][j] := (i shr (NumVariables - 1 - j)) and 1 = 1;

end;

procedure FillTruthTable(Edit1: TEdit);

type

TElExpr = array of String; //массив элементов выражения

var

MasOfEl: TElExpr;

Expression: String;

Element: String;

position: Integer;

counterPerem: Integer;

PolandStr: PeremMas; //польская запись

StackFunc: FuncMAS;

NumOfPolandEl, NumOfStackFuncEl, NumOfStackPerEl: Integer;

StackPerem: array of Boolean;

f: TextFile;

//Virazh: String[255];

begin

SetLength(StackFunc, 20); // Резервируем 100 элементов для стека функций

SetLength(PolandStr, 40); // Резервируем 100 элементов для вывода

StrOfPerem := '';

Expression := Edit1.Text; //

MasOfEl := TElExpr(Expression.Split([' '])); //

NumOfPolandEl := -1;

NumOfStackFuncEl := -1;

counterPerem := 0;

//WriteLn('Введите логическое выражение, заканчивающееся "stop":');

//Readln(Virazh);

// Основной цикл обработки

for ELement in MasOfEl do

begin

position := 0;

// Проверка, является ли ELement функцией

for var i := Low(allFunc) to High(allFunc) do

if ELement = allFunc[i] then

position := i;

if position = 0 then // Если это не функция

begin

Inc(NumOfPolandEl);

if NumOfPolandEl = Length(PolandStr) then

SetLength(PolandStr, Length(PolandStr) + 10);

PolandStr[NumOfPolandEl] := ELement; // Добавляем в польскую запись

if pos(ELement, StrOfPerem) = 0 then //проверка на то, была ли эта переменная в выражении до этого

begin

Inc(counterPerem);

StrOfPerem := StrOfPerem + ELement;

end;

end

else // Если это функция

begin

// Обрабатываем стек функций

if (NumOfStackFuncEl >= 0) and (StackFunc[NumOfStackFuncEl] < position) then //проверить на баги <= !!! сделать not

begin

while (NumOfStackFuncEl >= 0) and (StackFunc[NumOfStackFuncEl] < position) do //изменил

begin

Inc(NumOfPolandEl);

if NumOfPolandEl = Length(PolandStr) then

SetLength(PolandStr, Length(PolandStr) + 10);

PolandStr[NumOfPolandEl] := IntToStr(StackFunc[NumOfStackFuncEl]);

Dec(NumOfStackFuncEl);

end;

end;

Inc(NumOfStackFuncEl);

if NumOfStackFuncEl = Length(StackFunc) then

SetLength(StackFunc, Length(StackFunc) + 10);

StackFunc[NumOfStackFuncEl] := position; // Добавляем функцию в стек

end;

end;

if NumOfStackFuncEl <> -1 then

begin

while (NumOfStackFuncEl >= 0) and (StackFunc[NumOfStackFuncEl] >= position) do //одно лишнее условие?

begin

Inc(NumOfPolandEl);

if NumOfPolandEl = Length(PolandStr) then

SetLength(PolandStr, Length(PolandStr) + 10);

PolandStr[NumOfPolandEl] := IntToStr(StackFunc[NumOfStackFuncEl]);

Dec(NumOfStackFuncEl);

end;

end;

// Вывод результатов

//WriteLn('Результат преобразования:');

//for var i := 0 to NumOfPolandEl do

// Write(PolandStr[i], ' ');

//WriteLn;

ValueOfFunc := '1234'; //приоритет операций

GenerateTruthTable(counterPerem, TruthTable, NumRows);

SetLength(StackPerem, counterPerem); //размер стека равен количеству переменных

AssignFile(f, 'TruthTable.txt');

Rewrite(f);

for var j := 1 to counterPerem do //i = 5????

begin

write(f, StrOfPerem[j] + ' ');

end;

writeln(f, '| Result');

for var z := 0 to NumRows - 1 do

begin

NumOfStackPerEl := -1;

for var r := 0 to NumOfPolandEl do

begin

position := pos(PolandStr[r], ValueOfFunc);

if position = 0 then

begin

Inc(NumOfStackPerEl);

if NumOfStackPerEl = Length(StackPerem) then

SetLength(StackPerem, Length(StackPerem) + 5);

StackPerem[NumOfStackPerEl] := TruthTable[z][pos(PolandStr[r], StrOfPerem) - 1]; //занесение значения переменной в стек переменных

end

else

begin

//('not', 'and', 'or', 'xor')

case position of

1:

begin

StackPerem[NumOfStackPerEl] := not StackPerem[NumOfStackPerEl];

end;

2:

begin

Dec(NumOfStackPerEl);

StackPerem[NumOfStackPerEl] := StackPerem[NumOfStackPerEl] and StackPerem[NumOfStackPerEl + 1];

StackPerem[NumOfStackPerEl + 1] := false;

end;

3:

begin

Dec(NumOfStackPerEl);

StackPerem[NumOfStackPerEl] := StackPerem[NumOfStackPerEl] or StackPerem[NumOfStackPerEl + 1];

StackPerem[NumOfStackPerEl + 1] := false;

end;

4:

begin

Dec(NumOfStackPerEl);

StackPerem[NumOfStackPerEl] := StackPerem[NumOfStackPerEl] xor StackPerem[NumOfStackPerEl + 1];

StackPerem[NumOfStackPerEl + 1] := false;

end;

end;

end;

end;

for var k := 0 to counterPerem - 1 do

begin

write(f, TruthTable[z][k]);

write(f, ' ');

end;

TruthTable[z][counterPerem] := StackPerem[NumOfStackPerEl];

write(f, '| ');

writeln(f, TruthTable[z][counterPerem]);

//writeln(StackPerem[stackPer]);

end;

CloseFile(f);

//OpenTextFile('TruthTable.txt');

end;

end.

UnitSdnfSknf:  
unit UnitSdnfSknf;

interface

uses

UnitTruthTable;

procedure SDNFandSKNF(TruthTable: TTruth; StrOfPerem: String);

implementation

procedure SDNFandSKNF(TruthTable: TTruth; StrOfPerem: String);

var

h: Integer;

fSdnf: TextFile;

fSknf: TextFile;

StartFileSdnf, StartFileSknf: Boolean;

//CheckStartFile: String;

begin

AssignFile(fSdnf, 'Sdnf.txt');

Rewrite(fSdnf);

write(fSdnf, 'SDNF: ');

StartFileSdnf := true;

AssignFile(fSknf, 'Sknf.txt');

Rewrite(fSknf);

write(fSknf, 'SKNF: ');

StartFileSknf := true;

for var i := 0 to Length(TruthTable) - 1 do //количество строк

begin

{проверка значения выражения. Если true, то Сднф, иначе - Скнф}

if TruthTable[i][Length(TruthTable[i]) - 1] = true then

begin

//Read(fSdnf, CheckStartFile);

if StartFileSdnf <> true then

write(fSdnf, ' or ');

StartFileSdnf := false;

write(fSdnf, '(');

for var j := 0 to Length(TruthTable[i]) - 2 do //количество столбцов не считая столбец результата

begin

if TruthTable[i][j] = true then

begin

if j = 0 then

write(fSdnf, StrOfPerem[j + 1])

else

write(fSdnf, ' and ' + StrOfPerem[j + 1]);

end

else

begin

if j = 0 then

write(fSdnf, 'not ' + StrOfPerem[j + 1])

else

write(fSdnf, ' and ' + 'not ' + StrOfPerem[j + 1]);

end;

end;

write(fSdnf, ')');

end

else

begin

//Read(fSknf, CheckStartFile);

if StartFileSknf <> true then

write(fSknf, ' and ');

StartFileSknf := false;

write(fSknf, '(');

for var j := 0 to Length(TruthTable[i]) - 2 do //количество столбцов не считая столбец результата

begin

if TruthTable[i][j] = false then

begin

if j = 0 then

write(fSknf, StrOfPerem[j + 1])

else

write(fSknf, ' or ' + StrOfPerem[j + 1]);

end

else

begin

if j = 0 then

write(fSknf, 'not ' + StrOfPerem[j + 1])

else

write(fSknf, ' or ' + 'not ' + StrOfPerem[j + 1]);

end;

end;

write(fSknf, ')');

end;

end;

CloseFile(fSdnf);

CloseFile(fSknf);

end;

end.

UnitPolinomZh:

unit UnitPolinomZh;

interface

uses

UnitTruthTable;

type

TBoolStrPair = record

Koef: Boolean;

Perems: string;

end;

TMasOfKf = array of TBoolStrPair;

procedure ZhigalkinPol(TruthTable: TTruth; StrOfPerem: String);

implementation

(\*procedure ZhigalkinPol(TruthTable: TTruth; StrOfPerem: String);

var

ZhigalkinKf: TMasOfKf;

NumTableStrings: Integer;

NumTableСolumn: Integer;

NumKfPerem: Integer;

ZnachLastKf: Boolean;

IndexOfPerem, CounterTruthPerem: Integer;

begin

NumTableStrings := Length(TruthTable) - 1;

NumTableСolumn := Length(TruthTable[0]) - 1;

SetLength(ZhigalkinKf, NumTableStrings); //коэффициентов столько же, сколько и строк таблицы

ZhigalkinKf[0].Koef := TruthTable[0][NumTableСolumn];

NumKfPerem := 1;

for var i := 1 to NumTableStrings - 1 do

begin

ZnachLastKf := ZhigalkinKf[Length(ZhigalkinKf) - 1].Koef;

for var j := 1 to NumTableStrings do

begin

CounterTruthPerem := 0;

for var k := 0 to NumTableСolumn - 1 do

begin

if TruthTable[j][k] = true then

begin

IndexOfPerem := k + 1; //для поиска в строке переменных, где инд. с 1 идет

Inc(CounterTruthPerem);

end;

end;

if CounterTruthPerem = i then

begin

ZhigalkinKf[length(ZhigalkinKf)].Perems := Copy(StrOfPerem, 1, IndexOfPerem);

ZhigalkinKf[length(ZhigalkinKf)].Koef := TruthTable[j][NumTableСolumn] xor ZnachLastKf;

end;

end;

end;

end;\*)

procedure ZhigalkinPol(TruthTable: TTruth; StrOfPerem: String);

var

Values: array of Boolean;

Coeffs: array of Boolean;

N, i, j: Integer;

counterZhEl: Integer;

f: TextFile;

begin

N := Length(TruthTable); // Кол-во строк

SetLength(Values, N);

SetLength(Coeffs, N);

// Копируем значения функции (последний столбец)

for i := 0 to N - 1 do

Values[i] := TruthTable[i][High(TruthTable[i])];

// Строим треугольник Жегалкина — XOR-цепочка

for i := 0 to N - 1 do

Coeffs[i] := Values[i];

for i := 0 to N - 2 do

for j := N - 1 downto i + 1 do

Coeffs[j] := Coeffs[j] xor Coeffs[j - 1];

// Вывод коэффициентов

//Writeln('Zhigalkin polynomial coefficients:');

counterZhEl := 0;

AssignFile(f, 'Polinom.txt');

Rewrite(f);

write(f, 'Zhegalkin polynomial: ');

for i := 0 to N - 1 do

begin

if Coeffs[i] then

begin

// Определим моном по бинарному представлению номера i

if i = 0 then

begin

write(f, Coeffs[i]);

write(f, ' xor ');

end;

if counterZhEl <> 0 then

write(f, ' xor ');

for j := 0 to Length(TruthTable[0]) - 2 do

begin

if TruthTable[i][j] then

begin

write(f, StrOfPerem[j + 1]); // переменные индексируются с 1

Inc(counterZhEl);

end;

end;

end;

end;

CloseFile(f);

end;

end.

UnitMacClasky:

unit UnitMacClasky2;

interface

uses

UnitTruthTable;

type

RMinterm = record

Value: String;

NumOfTruth: Integer;

end;

TMinterms = array of RMinterm;

var

PrimeMinterms: array of String;

procedure ComapisonMinterms(FirstMinterm, SecondMinterm: RMinterm;

var CounterCovering: Integer; var CheckPrime: Boolean;

var MasMinterm2: TMinterms; var ChangesElements: Boolean;

var CounterMas2: Integer; var StringOfMinterms: String);

procedure GroupingMinterms(var MasMinterm1: TMinterms; var CounterGroups: Integer);

procedure CvMacClasky(TruthTable: TTruth);

implementation

procedure ComapisonMinterms(FirstMinterm, SecondMinterm: RMinterm;

var CounterCovering: Integer; var CheckPrime: Boolean;

var MasMinterm2: TMinterms; var ChangesElements: Boolean;

var CounterMas2: Integer; var StringOfMinterms: String);

var

CounterDiffEl: Integer;

NewElOfMas2: RMinterm;

begin

CounterDiffEl := 0;

NewElOfMas2.Value := '';

NewElOfMas2.NumOfTruth := 0;

for var i := 1 to Length(FirstMinterm.Value) do // Исправлено: индексация строки с 1

begin

if FirstMinterm.Value[i] <> SecondMinterm.Value[i] then

begin

Inc(CounterDiffEl);

NewElOfMas2.Value := NewElOfMas2.Value + '-';

end

else

begin

if FirstMinterm.Value[i] = '1' then

Inc(NewElOfMas2.NumOfTruth);

NewElOfMas2.Value := NewElOfMas2.Value + FirstMinterm.Value[i];

end;

end;

if (CounterDiffEl = 1) and (Pos(NewElOfMas2.Value, StringOfMinterms) = 0) then

begin

ChangesElements := true;

CheckPrime := false;

if CounterMas2 >= Length(MasMinterm2) then // Исправлено условие: >= вместо =

SetLength(MasMinterm2, Length(MasMinterm2) + 10);

MasMinterm2[CounterMas2] := NewElOfMas2; // Исправлено смещение индекса -1

Inc(CounterMas2);

StringOfMinterms := StringOfMinterms + NewElOfMas2.Value + ' ';

end;

end;

procedure GroupingMinterms(var MasMinterm1: TMinterms; var CounterGroups: Integer);

var

last: Integer;

Sorted: Boolean;

tempChangeMinterm: RMinterm;

begin

Sorted := false;

last := Length(MasMinterm1) - 1;

while not Sorted do

begin

Sorted := true;

for var i := 0 to last - 1 do

begin

if MasMinterm1[i].NumOfTruth > MasMinterm1[i + 1].NumOfTruth then

begin

tempChangeMinterm := MasMinterm1[i];

MasMinterm1[i] := MasMinterm1[i + 1];

MasMinterm1[i + 1] := tempChangeMinterm;

Sorted := false;

end;

end;

Dec(last);

end;

CounterGroups := 1;

for var i := 0 to Length(MasMinterm1) - 2 do

begin

if MasMinterm1[i].NumOfTruth <> MasMinterm1[i + 1].NumOfTruth then

Inc(CounterGroups);

end;

end;

procedure CvMacClasky(TruthTable: TTruth);

var

MasTableTruthMinterms: TMinterms;

CounterMinterms: Integer;

MasMinterm1: TMinterms;

MasMinterm2: TMinterms;

//PrimeMinterms: array of String;

BinaryCombination: String;

ChangesElements: Boolean;

CounterGroups: Integer;

CurrElementGroup, NextElementGroup: Integer;

CounterSr1: Integer;

CounterSr2, CounterSr2temp: Integer;

FindNewGroup: Boolean;

CounterMas2: Integer;

StringOfMinterms: String;

CounterCovering: Integer;

CheckPrime: Boolean;

CounterPrimeMinterms: Integer;

NewPrimeMinterm: RMinterm;

begin

SetLength(MasTableTruthMinterms, Length(TruthTable));

SetLength(MasMinterm1, Length(MasTableTruthMinterms));

SetLength(MasMinterm2, Length(MasTableTruthMinterms));

SetLength(PrimeMinterms, Length(MasTableTruthMinterms));

CounterMinterms := 0;

for var i := 0 to Length(TruthTable) - 1 do

begin

if TruthTable[i][High(TruthTable[i])] then // Исправлено: Length(TruthTable[0]) - 1 заменено на High()

begin

MasTableTruthMinterms[CounterMinterms].Value := '';

MasTableTruthMinterms[CounterMinterms].NumOfTruth := 0;

for var j := 0 to High(TruthTable[i]) - 1 do // Исправлено: High вместо Length - 2

begin

if TruthTable[i][j] then

begin

MasTableTruthMinterms[CounterMinterms].Value := MasTableTruthMinterms[CounterMinterms].Value + '1';

Inc(MasTableTruthMinterms[CounterMinterms].NumOfTruth);

end

else

MasTableTruthMinterms[CounterMinterms].Value := MasTableTruthMinterms[CounterMinterms].Value + '0';

end;

Inc(CounterMinterms);

end;

end;

SetLength(MasMinterm1, CounterMinterms); // Исправлено: урезание под количество

for var i := 0 to CounterMinterms - 1 do

MasMinterm1[i] := MasTableTruthMinterms[i];

ChangesElements := true;

CounterSr1 := 0;

CounterMas2 := 0;

CounterPrimeMinterms := 0;

StringOfMinterms := '';

while ChangesElements do

begin

ChangesElements := false;

GroupingMinterms(MasMinterm1, CounterGroups);

if CounterGroups = 1 then // Если только одна группа — это уже простые минтермы

begin

for var i := 0 to Length(MasMinterm1) - 1 do

begin

PrimeMinterms[CounterPrimeMinterms] := MasMinterm1[i].Value;

Inc(CounterPrimeMinterms);

end;

ChangesElements := false;

end

else

begin

if Length(MasMinterm1) > 0 then

CurrElementGroup := MasMinterm1[0].NumOfTruth;

CounterSr1 := 0;

for var i := 1 to CounterGroups - 1 do

begin

CounterSr2 := CounterSr1;

FindNewGroup := false;

while (CounterSr2 < Length(MasMinterm1)) and not FindNewGroup do

begin

if MasMinterm1[CounterSr2].NumOfTruth <> CurrElementGroup then

begin

FindNewGroup := true;

NextElementGroup := MasMinterm1[CounterSr2].NumOfTruth;

end

else

Inc(CounterSr2);

end;

while (CounterSr1 < Length(MasMinterm1)) and

(MasMinterm1[CounterSr1].NumOfTruth = CurrElementGroup) do

begin

CheckPrime := true;

CounterSr2temp := CounterSr2;

while (CounterSr2temp < Length(MasMinterm1)) and

(MasMinterm1[CounterSr2temp].NumOfTruth = NextElementGroup) do

begin

ComapisonMinterms(MasMinterm1[CounterSr1], MasMinterm1[CounterSr2temp], CounterCovering,

CheckPrime, MasMinterm2, ChangesElements, CounterMas2, StringOfMinterms);

Inc(CounterSr2temp);

end;

{if CheckPrime then

begin

PrimeMinterms[CounterPrimeMinterms] := MasMinterm1[CounterSr1].Value;

Inc(CounterPrimeMinterms);

end;}

Inc(CounterSr1);

end;

//if CounterSr2temp < Length(MasMinterm1) then

// CurrElementGroup := MasMinterm1[CounterSr2temp - 1].NumOfTruth; // изменил -1

CurrElementGroup := MasMinterm1[CounterSr1].NumOfTruth;

//CounterSr1 := CounterSr2temp;

end;

MasMinterm1 := Copy(MasMinterm2, 0, CounterMas2);

CounterMas2 := 0;

end;

{for var i := 0 to Length(MasMinterm1) - 1 do

begin

PrimeMinterms[CounterPrimeMinterms] := MasMinterm1[i].Value;

Inc(CounterPrimeMinterms);

end; }

end;

for var i := 0 to Length(MasMinterm2) - 1 do

begin

PrimeMinterms[CounterPrimeMinterms] := MasMinterm2[i].Value;

Inc(CounterPrimeMinterms);

end;

end;

end.

# ПРИЛОЖЕНИЕ Б

