Министерство образования и науки РФ

Пермский национальный исследовательский политехнический университет

Электротехнический факультет

Кафедра информационных технологий и автоматизированных систем

Дискретная математика

Лабораторная работа № 3

Тема: «Минимизация булевой функции методом Квайна»

Выполнил: студент группы ИВТ-22-1б

Игошев Матвей Иванович

Проверил: Ст. Преподаватель кафедры ИТАС

Рустамханова Г.И.

г. Пермь – 2023

Оглавление

Цель работы 3

Задачи работы 4

Этапы выполнения 5

Заключение 6

Список используемой литературы 7

# Цель работы

Создать консольное приложение, которое будет выполнять минимизацию функции согласно методу Квайна - Мак-Класки.

# Задачи работы

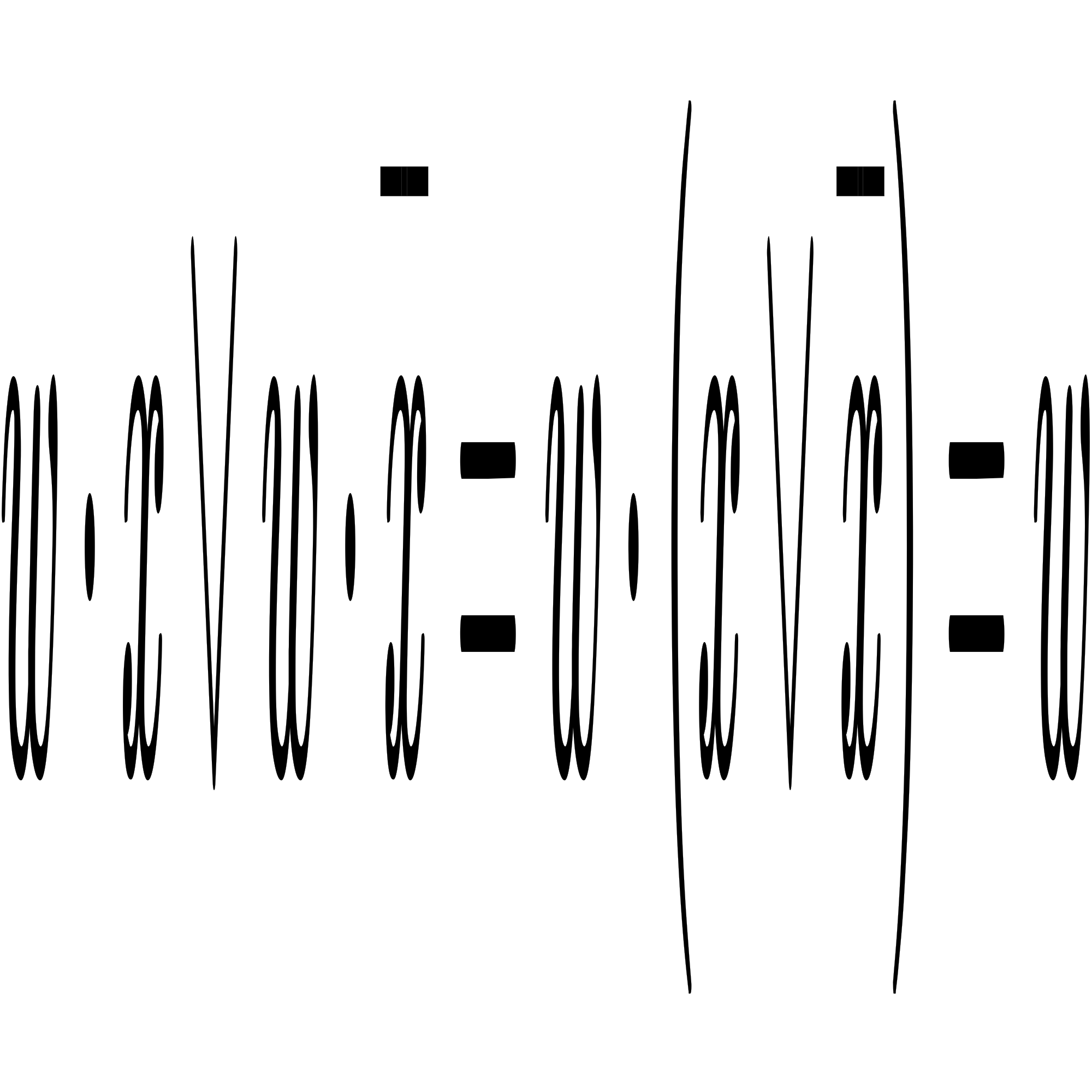
1. Изучить метод Квайна-Мак-Класки
2. Реализовать представление сднф в программе
3. Реализовать алгоритм получения таблицы импликантов
4. Протестировать приложение

# Этапы выполнения

**Теоретическая часть**

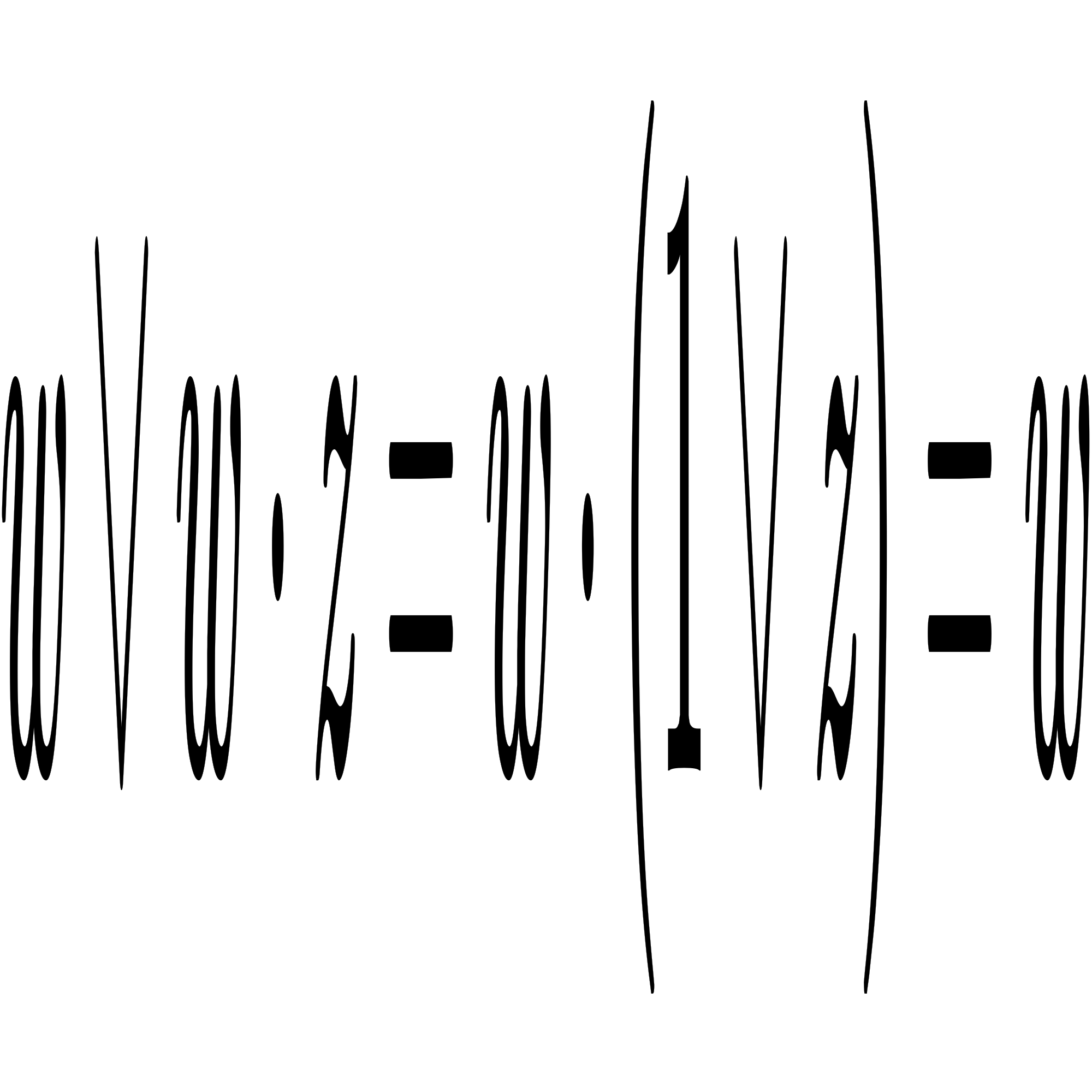
Метод минимизации функций Квайна-МакКласски подразумевает нахождение импликантов путем нахождения СДНФ, следующие импликанты получаются в результате их склеивания, предыдущих между собой, после производится поглощение, и так продолжается дальше, пока это возможно. После этого строится таблица(импликанты и конъюнкты), на которой выбирается минимальное покрытие. Данное минимальное покрытие будет являться МДНФ.

Операция склеивания сводится к нахождению пар членов. Выполняется операция таким образом:



Согласно этому выражению нужно найти такие пары, чтобы они отличались на одну переменную. Это правило выполняется на всех уровнях.

После происходит поглощение, которое основано на равенстве:



То есть те конъюнкты, из которых были образованы импликанты, грубо говоря, вычеркиваются из выражения.

Далее обе операции продолжают выполнятся, пока это возможно. Бывают случаи, когда сразу получается минимальная форма. Но как правило необходимо построение таблицы и дальнейшее нахождение минимального покрытия.

**Объяснение работы кода программы**

В данной версии программы покрытие находится вручную. Программа предоставляет лишь таблицу, для нахождения минимального покрытия.

Для реализации СДНФ использовалась структура вектор. Каждый из них состоит из 4 значений. Каждое значение отвечает за одну переменную, если значение равно -1, значит переменная не участвует в конъюнкции. Если же значение равно 1 или 0, значит что переменная имеется в конъюнкции в не инверсированном виде, иначе в инверсированном виде.

Для удобства используется объявление typedef:

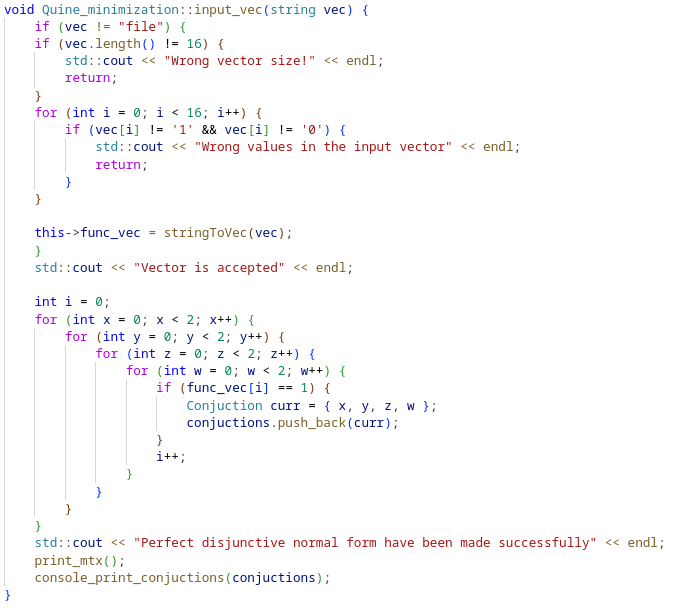


Также для удобства все необходимые функции упакованы в класс. Сам класс выглядит таким образом:



Выполнение программы начинается с введения векторов, которые преобразуются в СДНФ. Эту операцию выполняет функция input\_vec.

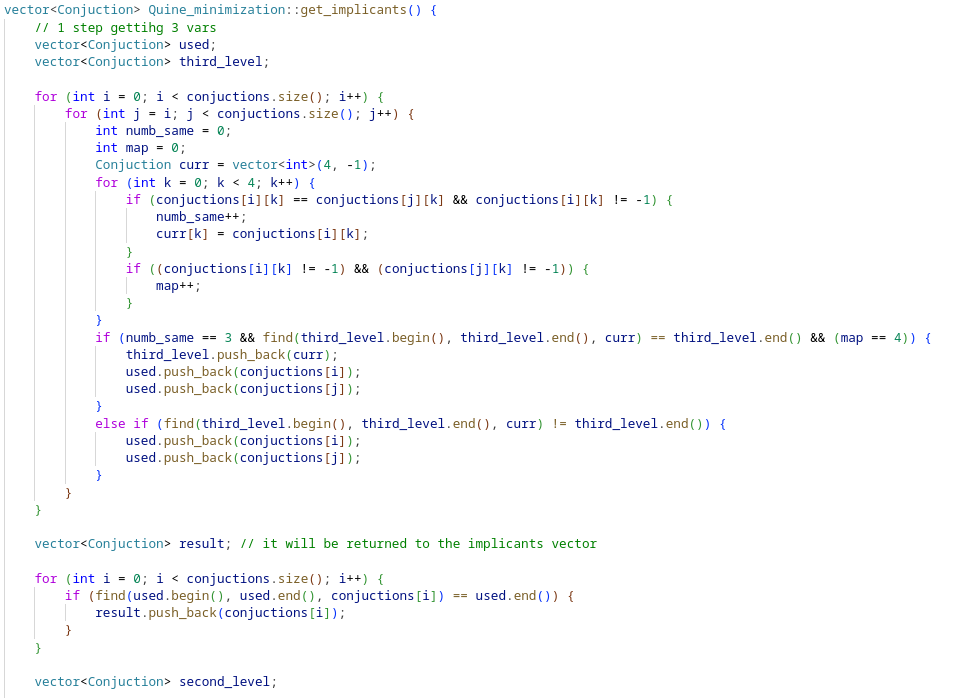
Выглядит функция следующим образом:



Сначала происходят проверки введенного вектора на его целостность, после из std::string он переводится в std::vector<int> с помощью функции stringToVec. Далее выводится сообщение об успешном обработке вектора. И в последнюю очередь преобразует вектор в СДНФ. После чего выводит таблицу истинности и СДНФ.

Далее происходит получение импликантов. Получение импликантов происходит с помощью функции get\_implicants.

Функция get\_implicants выполняет три шага по получению импликантов. На каждом из них происходит стадия склейки и поглощения. Часть функции выглядит следующим образом:

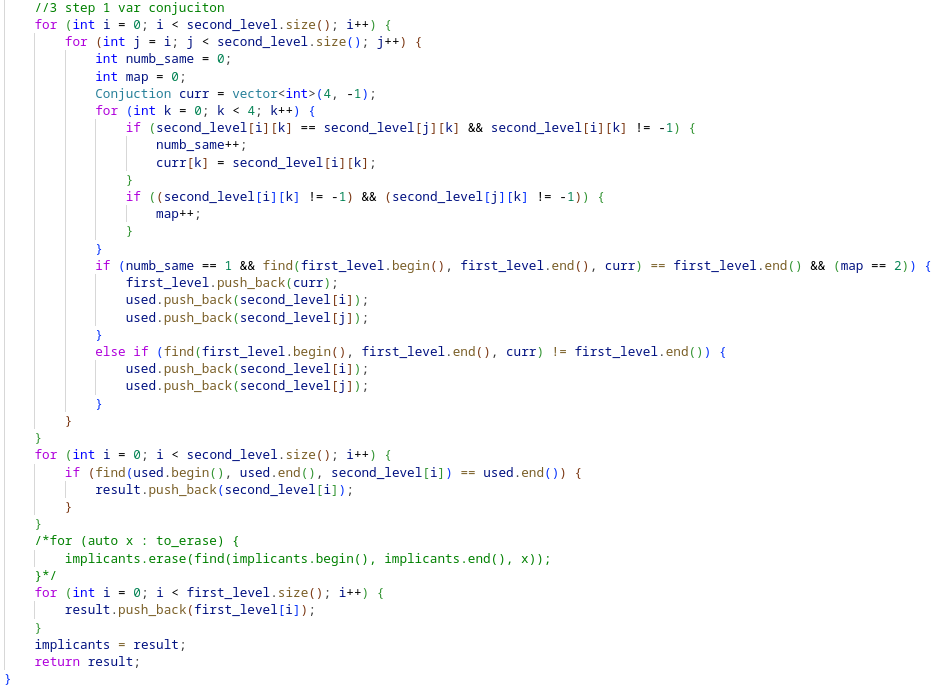


Сначала инициализируется два вектора used и third\_level. В вектор used записываются те конъюнкты, которые были использованы при склейке. Далее выполняется цикл, который обрабатывает каждый из конъюнктов в текущем векторе conjuctions. Происходит это с помощью numb\_same и map, и ряда проверок.

Цикл начинается с выбора двух разных конъюнктов, после создается вектор сurr, который выполняет функцию конъюнкта, в который запишется новый вектор в результате склейки. После программа пробегается по каждой из переменных и находит совпадающие, изменяет количество совпадающих переменных numb\_same и количество переменных в целом map.

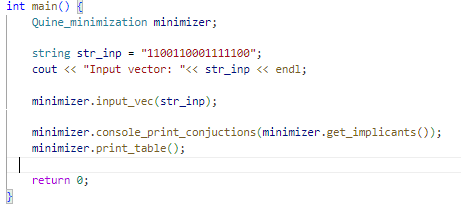
Далее программа выполняет проверку на наличие этого конъюнкта в векторе third\_level и добавляет конъюнкт curr в third\_level. А те конъюнкты, из которых был образован curr попадают в used. После создается вектор result, который будет возвращаться в конце функции. В этот вектор добавляются те вектора, которые не участвовали в склейках. После выполняются еще два таких же блока, но для другого количества переменных, поскольку их количество с каждым уровнем должно снижаться.

В конце функции результат записывается в поле класса implicants, для дальнейшего вывода с помощью таблицы. А также результат возвращается с помощью return, для вывода ДНФ через console\_print\_conjuctions.



После обработки СДНФ можно вывести таблицу с помощью print\_table.

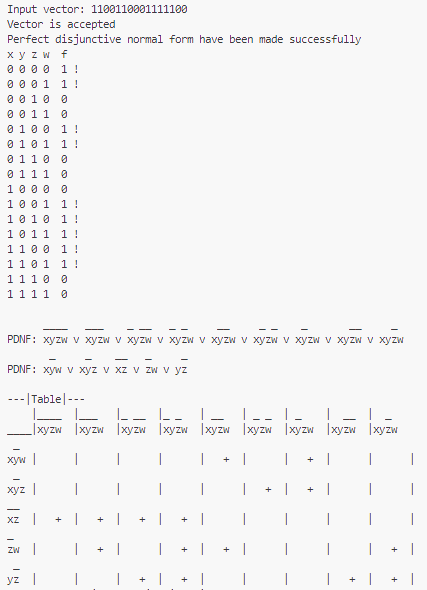
Для взаимодействия с классом можно использовать такой алгоритм:



**Тестирование программы**

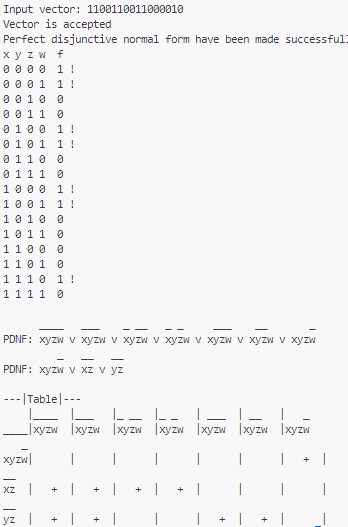
Для тестирования программы применяется тот алгоритм, который был показан чуть выше. Помимо этого было подобрано несколько векторов.

Результат первого тестирования:



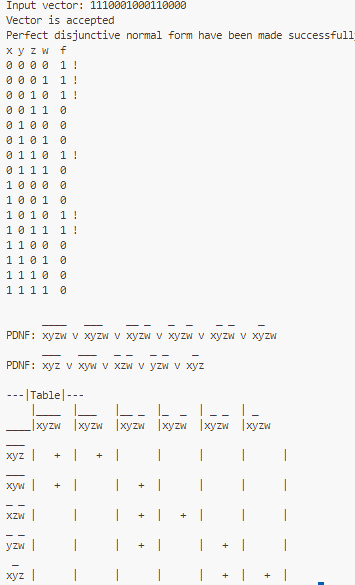
Минимальным покрытием для этого вектора будет МДНФ вида x-yw v x-yz v -x-z v y-z.

Результат второго тестирования:



Минимальным покрытием будет совпадать с ДНФ, который получился в результате склеек: xyz-w v -x-z v -y-z.

Результат третьего тестирования:



Минимальное покрытие для этого вектора будет: -x-y-z v -xz-w v x-yz.

# Заключение

В ходе работы были закреплены знания о минимизации функции с помощью метода Квайна-Мак Класски. Была реализована программа по получению таблицы импликантов. Также было проведено тестирование программы.

# Список используемой литературы

1. <https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%9A%D1%83%D0%B0%D0%B9%D0%BD%D0%B0>
2. <https://studfile.net/preview/958433/page:4/>