

Лабораторная работа № 1

Задача: Минимизировать СДНФ методом Квайна - Мак-Класки.

Важно:

- Лабораторная работа выполняется на языке C++ в среде Microsoft Visual Studio с использованием объектно ориентированного подхода. Результатом выполненной работы является программа, выполняющая требуемую задачу.

- При выполнении лабораторных работ оцениваются следующие критерии:

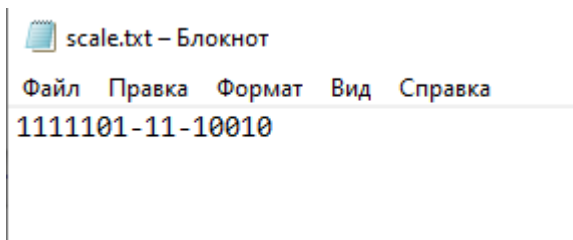
- написанная студентом программа и демонстрация её работы; оценивается корректность работы программы, соблюдение требований к программе;

- точность ответов на контрольные вопросы по теме лабораторной работы;

- самостоятельная работа при написании программ.

Требования к программе

Входные данные. Функция n переменных задана СДНФ, записанной в файле *scale.txt*, в виде шкалы длиной 2^n . Пример файла:



Выходные данные. Минимизированная МДНФ должна быть записана в файл *mdnf.txt* в виде перечня минимизированных импликант. Формат вывода:

- переменные импликанты, по которым была произведена склейка обозначаются символом «-»;
- переменные без инверсии обозначаются символом «1»;
- переменные с инверсией обозначаются символом «0».

Например, для МДНФ $z\bar{u} \vee \bar{x}\bar{u} \vee \bar{y}$ формат вывода следующий:

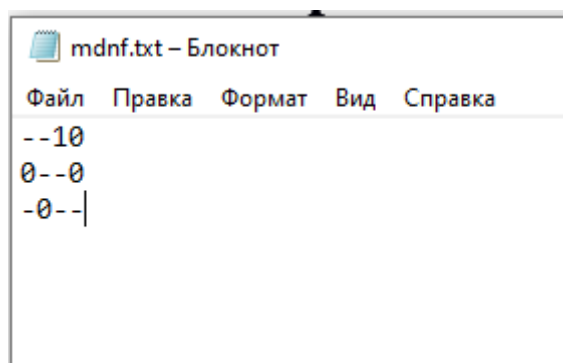
xuz

--10

0--0

-0--

Содержимое файла:



Реализация. Программа должна содержать следующие классы для представления импликант и булевых функций в виде СДНФ.

```
class Impl
{
private:
    int Num;
    int Ind;
    int P;
    bool Pw;
    bool Inf;

    static int Count(int); // Метод рассчитывает количество единиц в числе N
                           // Необходим для расчета поля индекса Ind

public:
    Impl(int); // Конструктор создает объект класса Impl из номера введенной импликанты
    ~Impl();
    static Impl Patch(Impl &, Impl &); // Метод проверяет условие теоремы Квайна – Мак-Класки для проверки возможности склеивания двух входных
                                         // импликант. Если склейка возможна, то метод возвращает результат склейки – новую импликанту,
                                         // в противном случае возвращает null. Метод изменяет значение поля Pw обеих импликант.
                                         // Для создания новой импликанты внутри метода вызывается конструктор Impl(int),
                                         // поле P новой импликанты заполняется непосредственным обращением к соответствующему полю внутри настоящего метода.
};
```

Рисунок 1 – Пример описания класса, представляющего импликанту.

```
class DNF // Disjunctive normal form
{
private:
    std::vector<Impl> Data;
public:
    DNF(std::string); // Конструктор создает объект класса DNF из строчной записи ДНФ
    ~DNF();

    void Minimize(); // Метод, минимизирующий ДНФ, хранящуюся в поле data методом Квайна – Мак-Класки
    void Print(std::ostream &); // Метод, выводящий буквенную запись ДНФ
    Impl& GetImpl(int); // Метод, выдающий импликанту из Data по порядковому номеру в vector

    // Остальные методы при необходимости добавить самостоятельно
};
```

Рисунок 2 – Пример описания класса, представляющего ДНФ.

Описание алгоритма

Представление ДНФ и СДНФ

В данной работе рассматривается алгоритм минимизации логической функции, представленной в совершенной дизъюнктивной нормальной форме (СДНФ). СДНФ представляет собой логическую сумму слагаемых, называемых импликантами. Каждая импликанта – это логическое произведение какой-либо комбинации логических переменных, являющимися аргументами логической функции. Например, СДНФ для функции трех переменных может иметь вид:

$$f(x, y, z) = \bar{x}y\bar{z} + xyz + x\bar{y}z. \quad (1)$$

Мы будем задавать СДНФ в виде массива, каждый элемент которого может принимать значения:

«1» - импликанта входит в СДНФ;

«0» - импликанта не входит в СДНФ;

«-» - неопределенное состояние.

Количество элементов данного массива равняется 2^n , где n - количество аргументов логической функции. В таблице показано, как интерпретировать буквенную запись СДНФ в СДНФ в виде массива на примере формулы (1).

Номер элемента массива	0	1	2	3	4	5	6	7
Номер элемента в двоичном коде	000	001	010	011	100	101	110	111
Импликанта в виде формулы	$\bar{x}\bar{y}\bar{z}$	$\bar{x}\bar{y}z$	$\bar{x}y\bar{z}$	$\bar{x}yz$	$x\bar{y}\bar{z}$	$x\bar{y}z$	$xy\bar{z}$	xyz
Значение элемента массива	0	0	1	0	0	1	0	1

Таблица 1 – Задание СДНФ в виде массива

Тогда СДНФ можно представить в виде номеров элементов массива: [2, 5, 7].

В общем случае логические функции, представленные в виде ДНФ могут содержать импликанты, в которые не обязательно входят все аргументы. Например,

$$f(x, y, z) = \bar{x}\bar{z} + \bar{y}. \quad (2)$$

В этом случае номеров элементов массива для задания ДНФ недостаточно. Необходимо для каждой импликанты хранить информацию о том, какие аргументы в нее не входят. Мы будем представлять импликанты в виде пар значений {номер, маска}. Маска – это число. Если представить его в

двоичном коде, то «1» будут стоять напротив тех аргументов, которые в импликанту не входят. Например, формулу 2 можно представить в виде двух пар $\{0, 2\}, \{1, 5\}$:

Номер элемента массива	0	1
Маска	2	5
Номер элемента в двоичном коде	000	001
Маска в двоичном коде	010	101
Импликанта СДНФ в виде формулы	$\bar{x}\bar{y}\bar{z}$	$\bar{x}\bar{y}z$
Импликанта ДНФ в виде формулы	$\bar{x}\bar{z}$	\bar{y}

Таблица 2 – Представление ДНФ в виде пар {номер, маска}

Так как СДНФ – это частный случай ДНФ, то ее тоже можно представить в таком виде. Для СДНФ маска для всех номеров равна нулю. Тогда формула 1 эквивалентна списку пар: $\{2, 0\}, \{5, 0\}, \{7, 0\}$. Далее в описании алгоритма будет использоваться такое представление.

Метод карт Карно

Известен графический метод минимизации СДНФ – метод карт Карно. Рассмотрим пример. Пусть задана логическая функция четырех аргументов:

$$1111101-11-10010.$$

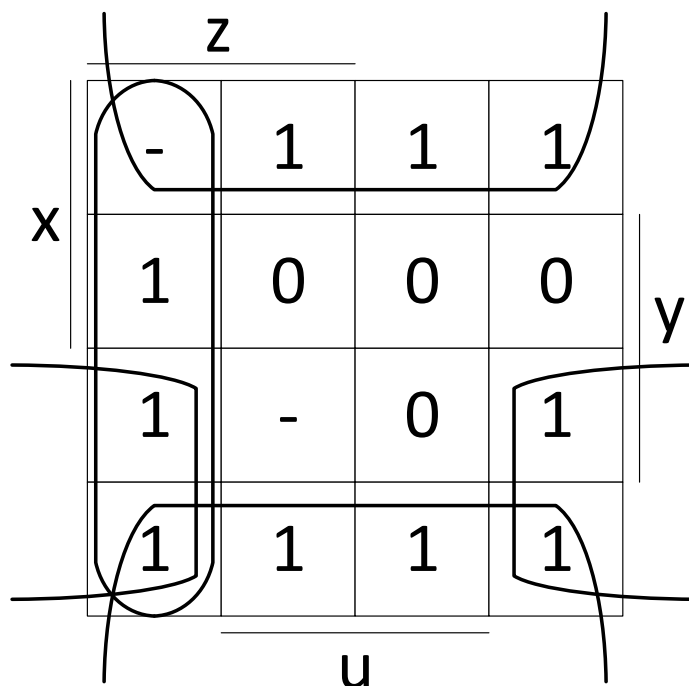


Рисунок 3 – Пример минимизации логической функции методом карт Карно

Минимизируем ее методом карт Карно (см. рисунок 3). Минимальная ДНФ (МДНФ) будет иметь вид:

$$f(x, y, z, u) = \bar{x}\bar{u} + z\bar{u} + \bar{y}, \quad (3)$$

или в виде списка пар {номер, маска}:

$$[\{0, 6\}, \{0, 11\}, \{2, 12\}].$$

Метод Квайна - Мак-Класки

Метод карт Карно – графический. Нам же предстоит написать программу для минимизации СДНФ. Одно из решений – реализация алгоритма Квайна-Макласки.

Алгоритм состоит из двух этапов:

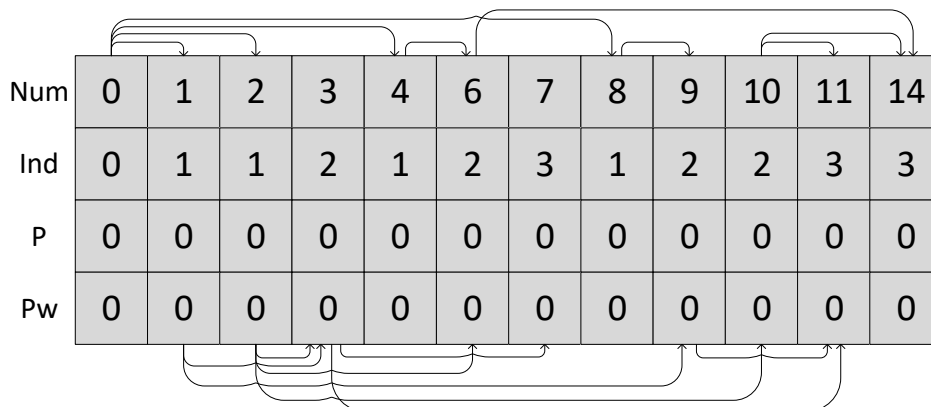
- 1) получение из СДНФ «тупиковой» ДНФ (ТДНФ);
- 2) получение МДНФ из ТДНФ.

Этап 1. Первый этап базируется на теореме Квайна.

Рассмотрим пример. В качестве исходных данных возьмем СДНФ, которую мы ранее минимизировали методом карт Карно.

Шаг 1. Сформируем список, в который для каждой импликанты выпишем четыре значения (два из них уже упоминалось ранее – {номер, маска}):

- номер (Num);
- индекс (Ind) – количество «1» в двоичном коде N;
- маска (P);
- индикатор склейки (Pw).



Num	0	1	2	3	4	6	7	8	9	10	11	14
Ind	0	1	1	2	1	2	3	1	2	2	3	3
P	0	0	0	0	0	0	0	0	0	0	0	0
Pw	0	0	0	0	0	0	0	0	0	0	0	0

Рисунок 4 – Исходная СДНФ в виде списка.

Стоит обратить внимание, что в данный список **включены** те импликанты, которые соответствуют неопределенному состоянию («-») в исходном массиве.

Шаг 2. Далее необходимо произвести всевозможные операции склейки импликант. Проверка возможности склейки каждой импликанты со всеми остальными, находящимися справа от нее в списке, осуществляется перебором. Интерпретируя суть теоремы Квайна, можно сформулировать условие возможности склейки пары импликант:

для возможности склейки пары импликант с номерами i и j необходимо выполнение следующих условий одновременно:

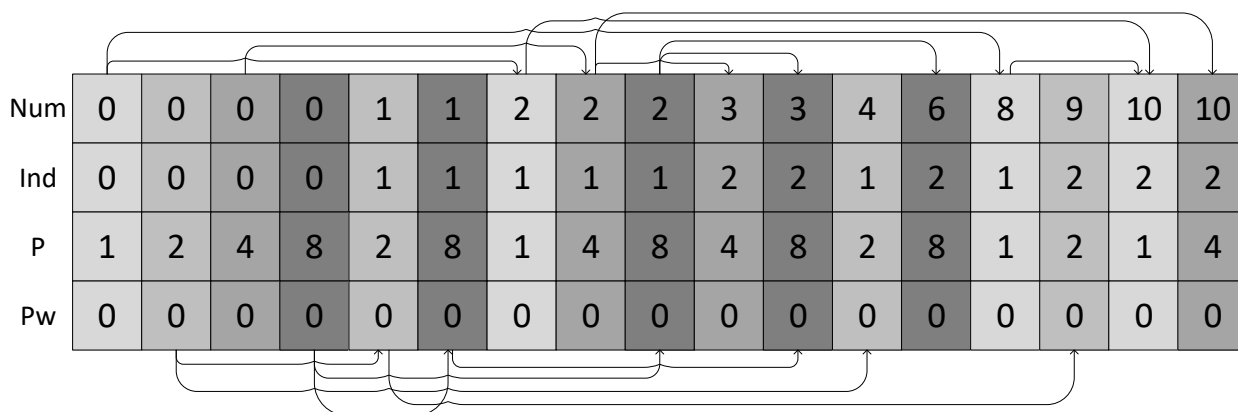
$$\begin{aligned} \text{Num}_i &< \text{Num}_j; \\ P_i &== P_j; \\ \text{Ind}_j - \text{Ind}_i &== 1; \\ \text{count}(\text{Num}_j - \text{Num}_i) &== 1. \end{aligned}$$

Последнее условие означает, что количество единиц в двоичном коде разности номеров должно быть равно 1. То есть номера должны отличаться на степень 2.

На рисунке 4 стрелками обозначены все возможные пары импликант, подлежащих склейке. В результате каждой склейки формируется новая импликанта, которая заносится в новый список (см. рисунок 6). В текущем списке необходимо изменить значение индикатора склейки с 0 на 1, если соответствующая импликанта хотя бы один раз участвовала в склейке (см. рисунок 5).

Num	0	1	2	3	4	6	7	8	9	10	11	14
Ind	0	1	1	2	1	2	3	1	2	2	3	3
P	0	0	0	0	0	0	0	0	0	0	0	0
Pw	1	1	1	1	1	1	1	1	1	1	1	1

Рисунок 5 – Текущий список (список-1) импликант после всевозможных склеек.



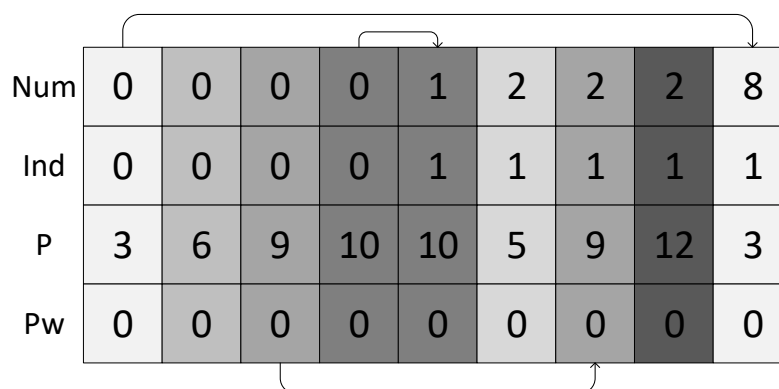
Num	0	0	0	0	1	1	2	2	2	3	3	4	6	8	9	10	10
Ind	0	0	0	0	1	1	1	1	1	2	2	1	2	1	2	2	2
P	1	2	4	8	2	8	1	4	8	4	8	2	8	1	2	1	4
Pw	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Рисунок 6 – Очередной список (список-2), сформированный в результате склеек импликант из предыдущего списка.

Шаг 2. С новым списком также выполняются всевозможные склейки с формированием следующего списка и обновления индикатора склейки в текущем списке. Эти действия выполняются до тех пор, пока в списке есть импликанты, подлежащие склейке. На рисунках 7-9 представлены списки, полученные на следующих шагах.

Num	0	0	0	0	1	1	2	2	2	3	3	4	6	8	9	10	10
Ind	0	0	0	0	1	1	1	1	1	2	2	1	2	1	2	2	2
P	1	2	4	8	2	8	1	4	8	4	8	2	8	1	2	1	4
Pw	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Рисунок 7 – Текущий список (список-2), после всевозможных склеек.



Num	0	0	0	0	1	2	2	2	8
Ind	0	0	0	0	1	1	1	1	1
P	3	6	9	10	10	5	9	12	3
Pw	0	0	0	0	0	0	0	0	0

Рисунок 8 – Очередной список (список-3), сформированный в результате склеек импликант из предыдущего списка.

Num	0	0	0	0	1	2	2	2	8
Ind	0	0	0	0	1	1	1	1	1
P	3	6	9	10	10	5	9	12	3
Pw	1	0	1	1	1	0	1	0	1

Рисунок 9 – Текущий список (список-3), после всевозможных склеек.

Num	0	0	2	2
Ind	0	0	1	1
P	6	11	5	12
Pw	0	0	0	0

Рисунок 10 – Итоговый список, соответствующий ТДНФ.

В итоговый список (рисунок 10) записываются все импликанты, которые на всех этапах не были подвергнуты операции склейки ($Pw = 0$). Импликанты, входящие в этот список составляют ТДНФ.

Этап 2. Для получения МДНФ из ТДНФ строится так называемая импликантная таблица (рисунок 11).

N		0	1	2	3	4	6	8	9	11	14	7	10
Нт.	0	+		+		+	+						
Рт.	6												
Нт.	0	+	+	+	+			+	+	+			+
Рт.	11												
Нт.	2			+	+		+					+	
Рт.	5												
Нт.	2			+			+				+		+
Рт.	12												

Рисунок 11 – Импликантная таблица.

Она формируется следующим образом. Строки – это импликанты из ТДНФ, столбцы – импликанты СДНФ. Поле таблицы отмечается знаком «+», если выполняется условие перекрытия:

$$N \& (\sim P_{T.}) == N_{T.}, \quad (5)$$

где N —номер импликанты СДНФ, $N_{T.}$ —номер импликанты ТДНФ, $P_{T.}$ —маска импликанты ТДНФ, «&» - побитовое «И», «~» - побитовое «НЕ».

При этом, на данном этапе импликанты СДНФ с неопределенным состоянием (отмечены красным) **не учитываются**. Согласно идее Мак-Класки, импликанту из ТДНФ можно исключить, если остальной набор импликант ТДНФ полностью перекрывает СДНФ, то есть в каждом столбце остается хотя бы один знак «+». МДНФ получается исключением из ТДНФ **максимально** возможного числа импликант, соблюдая условие перекрытия. В нашем примере исключить можно импликанту {2, 5} (отмечена темно-серым).

В конечном итоге мы получили МДНФ

$$[\{0, 6\}, \{0, 11\}, \{2, 12\}],$$

которая соответствует МДНФ, полученной при минимизации СДНФ методом карт Карно.

Контрольные вопросы

1. Что такое импликанта?
2. Что такое ДНФ, СДНФ, МДНФ? Привести примеры.
3. В чем суть метода карт Карно? Привести пример.
4. Как сформировать карту Карно для функции произвольного количества переменных? Привести пример.
5. Сформулировать теорему Квайна и изложить ее суть.
6. Алгоритм Квайна – Мак-Класки. Его назначение и основные этапы.
7. Как представить СДНФ в виде массива? Привести пример.
8. В чем отличия минимизации логических функций методом Квайна от минимизации алгоритмом Квайна – Мак-Класки?
9. В чем преимущество минимизации алгоритмом Квайна – Мак-Класки над минимизацией методом Квайна или картами Карно?
10. Что должно получиться в результате работы алгоритма Квайна – Мак-Класки, если входные данные представлены в виде массива, состоящего из нулей?
11. Что должно получиться в результате работы алгоритма Квайна – Мак-Класки, если входные данные представлены в виде массива, состоящего из единиц?
12. Другие вопросы по теме.