РЕФЕРАТ

Пояснювальна записка: ___ с., ___ рис., ___ табл., __ дод., ___ джерел.

Об'єкт розробки: алгоритми мінімізації булевих функцій.

Мета кваліфікаційної роботи: проектування та розробка навчального програмного додатку «Методи мінімізації булевих функцій» для обчислення задачі мінімізації, факторизації і декомпозиції булевих функції за допомогою метода Куайна-Мак-Класкі (Quine–McCluskey), основаного на кубічному представлені булевих функцій який є формалізованим та застосуванням карт Карно для вибору та візуалізації обраних мінтерм.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проведено аналіз предметної галузі, визначено актуальність завдання та призначення розробки, розроблена постановка завдання, задані вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі виконано аналіз існуючих рішень, обрано вибір платформи для розробки, виконано проектування і розробка програми, наведено опис алгоритму і структури функціонування програми, визначені вхідні і вихідні дані, наведені характеристики складу параметрів технічних засобів, описаний виклик та завантаження програми, описана робота програми.

В економічному розділі визначено трудомісткість розробленої інформаційної системи, проведений підрахунок вартості роботи по створенню програми та розраховано час на його створення.

Практичне значення роботи полягає в тому, що дана програма надає можливість зручного обчислення булевих функцій та їх застосування на заняттях таких предметів як: «Математична логіка», «Дискретна математика» та при конструюванні і спрощенні логічних схем електронних пристроїв комп'ютерів, калькуляторів, телефонних систем і ряду інших пристроїв, а також в таких областях, як розпізнавання образів, теорія кодування і криптографія.

Актуальність теми кваліфікаційної роботи визначається тим, що методи і поняття математичної логіки є основою, ядром інтелектуальних інформаційних систем, а засоби математичної логіки стали ефективним робочим інструментом для фахівців багатьох галузей науки і техніки. Тому розробки, що спрощують навчання студентів за рахунок сучасних технологій і роблять вивчення матеріалу більш цікавим та простішим.

Список ключових слів: БУЛЕВА ФУНКЦІЯ, МАТЕМАТИЧНА ЛОГІКА, МЕТОД КУАЙНА-МАК-КЛАСКІ, КАРТИ КАРНО, МІНТЕРМИ, НАВЧАЛЬНА ПРОГРАМА, ПРОГРАМУВАННЯ, ЗАСТОСУВАННЯ.

ABSTRACT

Explanatory note: ___ pp., ___ fig., ___ table, __ appendix, ___ sources.

Object of development: algorithms for minimizing Boolean functions.

The purpose of the qualification work: design and development of educational software "Methods of minimization of Boolean functions" to calculate the problem of minimization, factorization and decomposition of Boolean functions using the method of Quine-McCluskey (Quine-McCluskey), based on the cubic representation of Boolean functions which is formalized and using Carnot maps to select and visualize selected minterms.

The introduction considers the analysis and the current state of the problem, specifies the purpose of the qualification work and the scope of its application, provides a justification for the relevance of the topic and clarifies the problem.

In the first section the analysis of the subject area is carried out, the urgency of the task and purpose of development is defined, the statement of the task is developed, requirements to software realization, technologies and software are set.

The second section analyzes existing solutions, selects a platform for development, performs design and development of the program, describes the algorithm and structure of the program, determines the input and output data, provides characteristics of the parameters of hardware, describes the call and download of the program, describes the program .

The economic section determines the complexity of the developed information system, calculates the cost of work to create a program and calculates the time for its creation.

The practical significance of the work is that this program provides an opportunity to conveniently calculate Boolean functions and their application in classes such as: "Mathematical Logic", "Discrete Mathematics" and in the design and simplification of logic circuits of electronic devices of computers, calculators, telephones systems and a number of other devices, as well as in areas such as image recognition, coding theory and cryptography.

The relevance of the topic of qualification work is determined by the fact that the methods and concepts of mathematical logic are the basis, the core of intelligent information systems, and the means of mathematical logic have become an effective tool for professionals in many fields of science and technology. Therefore, developments that simplify the learning of students through modern technologies and make learning the material more interesting and easier.

List of key words: BOOL FUNCTION, MATHEMATICAL LOGIC, QUINE-MAC-CLASS METHOD, CARNO CARDS, MINTERMS, CURRICULUM, PROGRAMMING, APPLICATION.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ДДНФ – доскона́ла диз'юнкти́вна норма́льна форма;

ДКНФ – доскона́ла кон'юнкти́вна норма́льна форма;

ДНФ – диз'юнкти́вна норма́льна форма;

ЕУ – елементи управління;

КНФ – кон'юнкти́вна норма́льна форма;

ОС – операційна система;

GUI – Graphical User Interface графічний користувальницький інтерфейс;

CUI – Console User Interface, консольний користувальницький інтерфейс.

ЗМІСТ

ВСТУП

Математична логіка є сучасною формою формальної логіки що застосовує математичні методи для дослідження свого предмета. Інші її назви символічна логіка, теоретична логіка, логістика. У формальній логіці і, відповідно, в математичній логіці, зібрані результати законів структури правильних висновків. Висновок є таким розумовим процесом, в результаті якого з'являються нові відкриття на підставі вже наявних, які передбачаються правильними, без практичних досліджень.

У дійсності, нове відкриття, отримане в результаті виведення, так званий остаточний висновок у прихованій формі знаходиться в попередньо наявних знаннях, у так званих передумовах. Найпростіші закономірності висновків відкривалися людством емпіричним шляхом в ході суспільного виробництва наприклад, найпростіші співвідношення арифметики і геометрії.

Відкриття більш складних законів пов'язано з результатами науки формальної логіки. Перше велике узагальнення формальної логіки належить Аристотелю.

В даний час результати математичної логіки використовуються в усіх традиційних областях формальної логіки, відкриті абсолютно нові області. Математична логіка не претендує на відкриття законів мислення взагалі, або ще меншою мірою на аналіз філософських проблем, пов'язаних з людським мисленням. Ці питання більше відносяться до логіки в більш загальному сенсі слова і до філософії.

Головна мета застосування в логіці математичної символіки полягала в тому, щоб звести операції з логічними висновками до формальних дій над символами. При цьому початкові положення записуються формулами, які перетворяться за певними законами, а отримані результати тлумачаться у відповідних поняттях.

Булеві функції знаходять застосування в логіці, електротехніці, багатьох розділах інформатики.

Застосування булевих функцій використовується в конструюванні і спрощенні логічних схем електронних пристроях комп'ютерів, калькуляторів, телефонних систем і ряду інших пристроїв.

У другій половині двадцятого століття були відкриті ряд важливих застосувань теорії булевих функцій в таких областях, як розпізнавання образів, теорія кодування і криптографія.

Наприклад, в комп'ютерній графіці така булева операція як "складання по модулю два" застосовується при виведенні спрайту на картинку – повторне її застосування прибирає спрайт з картинки. Завдяки інволютивності ця ж операція знайшла застосування в криптографії як проста реалізація абсолютно стійкого шифру (шифру Вернама). "Складання по модулю два" також може використовуватися для обміну двох змінних, використовуючи алгоритм обміну за допомогою того, що виключає АБО.

Метою кваліфікаційної роботи бакалавра є проектування та розробка навчального програмного додатку «Методи мінімізації булевих функцій», складеної за допомогою мови програмування C# та технології WindowsForms.

Дана програма дозволить користувачу настільної операційної системи Windows обчислити задачі мінімізації, факторизації і декомпозиції булевих функції за допомогою метода Куайна-Мак-Класкі (Quine–McCluskey), основаного на кубічному представлені булевих функцій який є формалізованим та застосуванням карт Карно для вибору та візуалізації обраних мінтерм.

Дана програма корисна в навчальному процесі при обчисленні булевих функцій на заняттях таких предметів як: «Математична логіка», «Дискретна математика» та при конструюванні і спрощенні логічних схем електронних пристроїв комп'ютерів, калькуляторів, телефонних систем і ряду інших пристроїв, а також в таких областях, як розпізнавання образів, теорія кодування і криптографія.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1. Загальні відомості з предметної галузі

1.1.1. Основні термінологічні поняття

Математична логіка – це розділ математики, який вивчає мислення за допомогою числень, застосовуючи математичні методи та спеціальний апарат символів. Тобто, математична логіка по суті є формальною логікою, що використовує математичні методи.

Предметом вивчення ж математичної логіки є взагалі будь-яка математична теорія.

Сфера застосування математичної логіки дуже широка. З кожним роком зростає глибоке проникнення ідей та методів математичної логіки в інформатику, обчислювальну математику, лінгвістику, філософію. Потужним імпульсом для розвитку та розширення сфери застосування математичної логіки стала поява електронно-обчислювальних машин. Виявилося, що в рамках математичної логіки вже є готовий апарат для проектування обчислювальної техніки. Методи і поняття математичної логіки є основою, ядром інтелектуальних інформаційних систем. Засоби математичної логіки стали ефективним робочим інструментом для фахівців багатьох галузей науки і техніки.

Потрібно відмітити, що використання матлогіки дуже ефективне в інформатиці. Річ у тому, що математична логіка має відношення і до програмного забезпечення комп’ютера, і до його «заліза». Наприклад булева алгебра (алгебра логіки) використовуються як в програмуванні, так і в розробці апаратного забезпечення комп’ютера.

Булева множина - в математиці, множина з двома елементами, що інтерпретуються як «істина» та «хибна». Зазвичай позначається як {0,1}.

Найпоширенішою алгебраїчною структурою на булевій множині є булева

алгебра з двома елементами.

В інформатиці, змінна булевого типу - це змінна, що приймає значення з булевої множини.

Таблиця істинності – математична таблиця, що широко використовується у математичній логіці зокрема в алгебрі логіки, численні висловлень для обчислення значень булевих функцій.

Під «логічною функцією» (також логічною операцією) у цьому випадку розуміється функція, у котрої значення змінних (параметрів функції) і значення самої функції виражають логічну істинність.

Наприклад, в двозначній логіці вони можуть приймати значення «істина» або «хиба» (true або false), (1 або 0).

Семантика – це сукупність правил, які надають формулам значення істинності.

Нехай A та B - формули. Тоді значення істинності формул (¬A), (A∧B), (A∨B), (A→B) та (A↔B) так пов'язані зі значеннями істинності формул A та B:

Для булевої функціїї з n змінних (x1,..xn), елементарна кон'юнкція, в якій кожна з n змінних набуває значення одиниці лише на одному з кортежів своїх змінних називається мінтермом (конституентою одиниці). Отже, мінтерм це логічний вираз, який використовує лише операцію доповнення та операцію кон'юнкції. Кількість різних мінтермів дорівнює кількості кортежів змінних, тобто 2^n для n змінних. Наприклад, abc, ab'c і abc' - три з восьми мінтермів для булевої функції з восьми змінних(a,b і c). Читаються ці вирази як «a і b і c», «a і не b і c „ a і b і не c“ відповідно.

Кон’ю́нкція – логічна операція, за змістом максимально наближена до сполучника «і». Найбільш часто використовуються такі позначення кон’юнкції: A ⋀ B, A&B, A and B, A·B.

При цьому позначення A⋀B найбільш широко поширено в сучасній математиці та математичній логіці, де воно, конкурує зі знаком амперсанда &

Кон'юнкти́вна норма́льна фо́рма (КНФ) в булевій логіці - нормальна форма в якій булева формула має вид кон'юнкції декількох диз'юнктів (де

диз'юнктами називаються диз'юнкції декількох пропозиційних символів або їх заперечень). Кон'юнктивна нормальна форма широко використовується в автоматичному доведенні теорем, зокрема вона є основою для використання правила резолюції.

Правило: результат дорівнює 1, якщо всі вирази дорівнюють 1; у всіх інших випадках результат дорівнює 0.

Таблиця істинності наведена в табл. 1.1.

Таблиця 1.1

Таблиця істинності

A	B	A∧B
1	1	1
1	0	0
0	1	0
0	0	0

Диз’юнкція – операція, за змістом максимально наближена до сполучника «або».

Диз'юнкція (лат. disjunctio - розділення) (операція OR) - двомісна логічна операція, що має значення «істина», якщо хоча б один з операндів має значення «істина». Операція відображає вживання сполучника «або» в логічних висловлюваннях. Диз'юнкція є бінарною операцією, тобто, має два операнда. Запис може бути префіксним - знак операції стоїть перед операндами, інфіксним - знак операції стоїть між операндами або постфіксним - знак операції стоїть після операндів. При числі операндів більше 2-х префіксний і постфіксний записи економніші.

Позначається: в в програмуванні як | чи or. Найбільш часто використовуються такі позначення диз’юнкції: *A||B, A|B, A* ∨ *B, A+B, A* or *B.*

У техніці операцію диз'юнкції втілює логічний вентиль АБО.

Диз'юнкти́вна норма́льна фо́рма (ДНФ) в булевій логіці - нормальна

форма, в якій булева формула має вид диз'юнкції декількох кон'юнктів (де кон'юнктами називаються кон'юнкції декількох пропозиційних символів або їх заперечень).

Правило: результат дорівнює 1, якщо хоча б один із виразів дорівнює 1; у всіх інших випадках результат дорівнює 0.

Таблиця істинності наведена в табл. 1.2.

Таблиця 1.2

Таблиця істинності

A	B	A V B
1	1	1
1	0	1
0	1	1
0	0	0

Імплікація – операція, за змістом максимально наближена до сполучника «якщо … , то …». Найбільш часто використовуються такі позначення імплікації: $A \therefore B$, $A \models B$, $A \rightarrow B$, $A \Rightarrow B$.

Правило: результат дорівнює 0 лише тоді, коли посилка (А) дорівнює 1, а наслідок (В) дорівнює 0.

Таблиця істинності наведена в табл. 1.3.

Таблиця 1.3

Таблиця істинності

A	B	A ⇒ B
1	1	1
1	0	0
0	1	1
0	0	1

Еквіваленція – операція, за змістом максимально наближена до сполучника «тоді й тільки тоді». Найбільш часто використовуються такі позначення еквіваленції: $A \equiv B$, A ↔ B.

Правило: результат дорівнює 1 тоді і тільки тоді, коли обидва вирази мають однакове значення, в інших випадках результат дорівнює 0.

Таблиця істинності наведена в табл. 1.4.

Таблиця 1.4

Таблиця істинності

A	B	A ↔ B
1	1	1
1	0	0
0	1	0
0	0	1

Заперечення – операція, за змістом максимально наближена до частки «не». Найбільш часто використовуються такі позначення заперечення: *!A*, ¬*A*,~*A*.

Правило: результат дорівнює 1 тоді, коли вираз дорівнює 0 і навпаки, результат дорівнює 0, коли вираз дорівнює 1.

Таблиця істинності наведена в табл. 1.5.

Таблиця 1.5

Таблиця істинності

A	¬A
1	0
0	1

1.1.2. Синтаксис та семантика логічних виразів

У логіці висловлювань вираз А або складне висловлювання називають правильно побудованою формулою, або формулою. При вивченні формул розглядають їх два аспекти – синтаксис та семантику.

Синтаксис – це сукупність правил, які дозволяють будувати формули та розпізнавати правильні формули серед послідовностей символів. Формули у логіці висловлювань визначають за такими правилами:

1. Вираз є формулою.
2. Якщо А формула, то (¬А) - теж формула.
3. Якщо А та В - формули, то (А V В), (А⋀В), (А→В), (¬А) - формули.
4. Жодних інших формул, крім породжених застосуванням вказаних вище правил, немає.

Формули, так само як і операнди, позначають латинськими буквами з індексами або без них.

Вирази (А→), (А ∧), (А¬), (∨В) – не формули.

Наприклад, нам потрібно побудувати таблицю істинності для виразу

а) ¬А V В. Для цього спочатку побудуємо таблицю істинності для ¬А та додамо у цю таблицю значення В (табл. 1.6).

Таблиця 1.6

Таблиця істинності

А	В	¬А
1	1	0
0	1	1
1	0	0
0	0	1

Потім додамо в цю таблицю стовпець із значенням ⌐A V B , для цього виконаємо операцію диз'юнкції для ⌐A та B (табл. 1.7):

Таблиця 1.7

Таблиця істинності

A	B	⌐A	⌐A V B
1	1	0	1
0	1	1	1
1	0	0	0
0	0	1	1

Отриманий останній стовпець і перші дві колонки є таблицею істинності даного виразу (табл. 1.8):

Таблиця 1.8

Таблиця істинності даного виразу

A	B	⌐A V B
1	1	1
0	1	1
1	0	0
0	0	1

б) (A V B) ⇒(A ⋀ B). Як і раніше, будуємо таблицю істинності для A, B та A V B, також у цій же таблиці будуємо A ⋀ B та (A V B) ⇒(A ⋀ B) (табл. 1.9):

Таблиця 1.9

Таблиця істинності

A	B	A ⋀ B	A V B	(A V B) ⇒(A ⋀ B)
1	1	1	1	1
0	1	0	1	1
1	0	0	1	1
0	0	0	0	0

Кінцева таблиця істинності буде виглядати так (табл. 1.10):

Таблиця 1.10

Таблиця істинності

A	B	(A V B) ⇒(A ⋀ B)
1	1	1
0	1	1
1	0	1
0	0	0

в) (A⇒B) ⋀ (A V (⌐B)) Виконуємо аналогічні дії:

Будуємо таблиці A, B, ⌐ B, A⇒B, A V(⌐ B) та (A⇒B) ⋀ (A V(⌐ B)) (табл.. 1.11):

Таблиця 1.11

Таблиця істинності

A	B	⌐B	A⇒B	A V(⌐B)	(A⇒B) ⋀ (A V(⌐ B))
1	1	0	1	1	1
0	1	0	1	0	0
1	0	1	0	1	0
0	0	1	1	1	1

1.2. Призначення розробки та галузь застосування

Метою кваліфікаційної роботи є проектування та розробка навчального програмного додатку «Методи мінімізації булевих функцій», складеної за допомогою мови програмування C# та технології WindowsForms.

Дана програма дозволить користувачу настільної операційної системи Windows обчислити задачі мінімізації, факторизації і декомпозиції булевих функції за допомогою метода Куайна-Мак-Класкі (Quine–McCluskey), основаного на кубічному представлені булевих функцій який є формалізованим та застосуванням карт Карно для вибору та візуалізації обраних мінтерм.

Дана програма корисна в навчальному процесі при обчисленні булевих функцій на заняттях таких предметів як: «Математична логіка», «Дискретна математика» та при конструюванні і спрощенні логічних схем електронних пристроїв комп'ютерів, калькуляторів, телефонних систем і ряду інших пристроїв, а також в таких областях, як розпізнавання образів, теорія кодування і криптографія.

1.3. Підстава для розробки

Підставою для розробки кваліфікаційної роботи бакалавра на тему «Розробка навчального програмного додатку «Методи мінімізації булевих функцій» за допомогою мови програмування C#» є наказ по Національному технічному університету «Дніпровська політехніка» від __.__. 2021р. № ____-_.

1.4. Постановка завдання

Завданням кваліфікаційної роботи бакалавра є спроектувати та розробити навчальний програмний додаток «Методи мінімізації булевих функцій», який призначений для швидкої перевірки правильності обчислення булевих функцій

з декількох змінних у навчальних умовах за допомогою: методу Куайна-Мак-Класкі та візуалізації обраних мінтерм картами Карно.

Для виконання даного завдання необхідно виконати наступні етапи:

1. Дослідити, проаналізувати та вивчити матеріал з теми «Математична логіка» .
2. Структурувати інформацію.
3. Розробити структуру додатку.
4. Спроектувати інтерфейс системи.
5. Програмно реалізувати алгоритм роботи інформаційної системи.
6. Виконати тестування додатку.

Програмний додаток «Методи мінімізації булевих функцій» повинен мати інтуїтивно зрозумілий інтерфейс користувача (UI) для введення та обробки даних.

Вхідними даними програми повинні бути:

– вибір кількісті змінних булевої функції – від 2 до 6;
– очікуваний результат виконання функції у вигляді ряду сум імпікант.

Вихідними даними програми є:

– карти Карно;
– представлення рішення Куайна-Мак-Класкі;
– дійсна форма функції;
– скорочена результуюча мінімізована форма функції.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

До розробленої системи виставляються наступні функціональні вимоги:

1. На початку виконання мінімізації булевої функції (БФ) необхідно надати користувачу можливість вибору кількості змінних функції у визначеному діапазоні.
2. Для вибору мінтерм БФ, як одного з видів вхідних даних, потрібно

запровадити в інтерфейсі програми візуалізовані карти Карно, у вигляді активних інформативних інструментів візуалізації БФ.

3. Карти Карно повинні генеруватися за кількістю обраної кількості змінних БФ, містити заголовки рядків та стовбців, які мають нумерацію у двійковому коді. Кожна комірка карти Карно повинна мати інформативні ознаки:

– представлення її номеру за координати розташування у двійковій та десятковій системі числення;

– кольорове забарвлення – ознака вибору комірки як мінтерми БФ.

4. Необхідно забезпечити UI скасуванням обраної мінтерми повторним обранням її на карті та видалення її з первинної БФ.

5. Для запуску та спостереженням за обчислювальним процесом мінімізації БФ, UI забезпечити інструментом запуску у вигляді кнопки та полів для перегляду оброблених імплекант у двійковій системі числення.

1.5.2. Вимоги до інформаційної безпеки

Для надійної роботи інформаційної системи зі сторони операційної системи необхідно дотримуватися таких факторів:

- використання ліцензійного ПЗ;
- захист від зловмисних програм;
- архівація даних на сервері;
- встановлення блоків безперебійного живлення.

Надійність роботи програмного забезпечення залежить від надійності операційної системи, під управлінням якої вона буде функціонувати, а також надійності розроблюваного програмного забезпечення.

Для надійності роботи програмного забезпечення зі сторони розробленої ІС потрібна перевірка введених даних користувачем для виключення можливих зловмисних ситуацій.

1.5.3. Вимоги до складу та параметрів технічних засобів

Для користування розробленим додатком необхідно мати персональний комп’ютер з наступними технічними характеристиками.

Мінімальні характеристики для роботи програми [21]:

– процесор із тактовою частотою 2,6 ГГц;
– оперативна пам’ять 2 ГБ;
– вільне місце на жорсткому диску 2 ГБ;
– графічна плата 256 МБ;
– монітор з діагоналлю 17 дюйм;
– роздільна здатність 1440x900(16:10) пікс. .

Рекомендовані характеристики для роботи в програмі: комп’ютер має відповідати таким вимогам [15]:

– процесор 3,2 ГГц;
– оперативна пам’ять 4 ГБ;
– вільне місце на жорсткому диску 2 ГБ;
– графічна плата 512 МБ;
– монітор з діагоналлю 17 дюйм ;
– роздільна здатність 1440x900 (16:10) пикс.

1.5.4. Вимоги до інформаційної та програмної сумісності

Програмний додаток повинен працювати в ОС Windows7/8.1/10.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення системи

Розроблений навчальний програмний додаток «Методи мінімізації булевих функцій» для настільної ОС Windows дозволяє користувачеві ознайомитися з методом мінімізації буле-вих функцій Куайна-Мак-Класкі та формування карт Карно.

Інтерфейс програми надає користувачеві можливість працювати лише мишею. Отримані результати мінімізації функції користувач може копіювати у буфер обміну для подальшого використання.

2.2. Опис застосованих математичних методів

2.2.1. Карта Карно

Карта Карно (К-карта скорочено) - метод спрощення виразів булевої алгебри, зроблений Морісом Карно в 1953 як поліпшення Діаграм Вейча, винайдених Едвардом Вейчем в 1952. Карта Карно зменшує потребу в обширних обчисленнях, використовуючи перевагу людської можливості розпізнання шаблонів, дозволяє швидке розпізнавання і виключення потенційних станів гонитви (рис. 2.1).

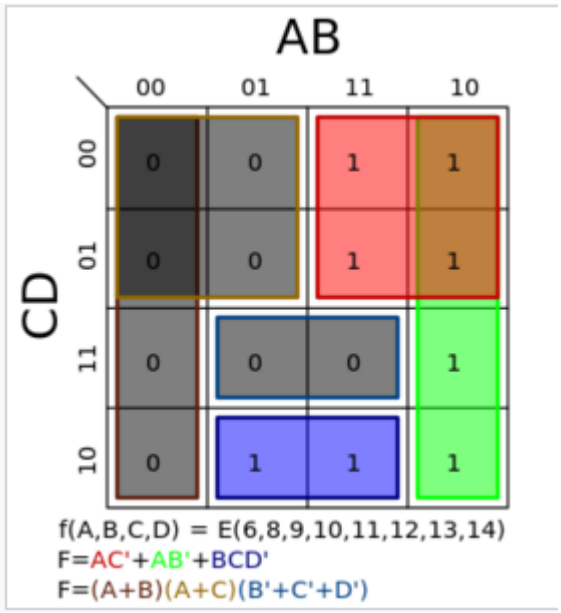

Рис. 2.1. Карта Карно

В карті Карно булеві змінні переносяться (зазвичай з таблиці істинності) і впорядковуються згідно з принципами кода Грея, в якому тільки одна змінна змінюється при переході між сусідніми квадратами. Коли таблиця згенерована, і у відповідні комірки записані вихідні значення, дані організовуються в найбільші можливі групи, що містять 2^n комірок (n=0,1,2,3...)[6]. Далі, працюючи з цими групами, отримують мінімізовану ДНФ.

Зазвичай, значні обчислення потрібні для отримання мінімального виду булевої функції, однак карта Карно зменшує потребу таких обчислень завдяки:

1. Використанню можливості людського розуму по розпізнаванню шаблонів для визначення які терми мають бути поєднані для отримання найпростішого виразу.
2. Дозволяє швидко визначити та видалити потенційні стани гонитв, які неминучі в булевих рівняннях.
3. Забезпечує найкращу допомогу в спрощенні до шістьох змінних, однак з більшою кількістю змінних стає складно розрізнити оптимальні шаблони.
4. Допомагає в навчанні про булеві функції та їх мінімізацію.

Карта Карно зазвичай становиться важкою для розпізнання при збільшені

кількості змінних. Загальне правило таке: карта Карно добре працює до чотирьох-п'яти змінних, і не має використовуватись з більше ніж шістьома змінними. Для виразів з більшою кількістю змінних може бути використаний Метод Куайна - Мак-Класкі. Сьогодні здебільшого для процесу мінімізації використовуються комп'ютери, для яких евристичний алгоритм еспресо став стандартною програмою мінімізації.

Алгоритм:

1. Кожна змінна має два значення: початкове та обернене. Змінні впорядковуються згідно з кодом Грея, тобто тільки одна змінна змінюється між двома суміжними комірками. У відповідні комірки записуємо вихідні значення функції.

2. Коли карта Карно заповнена, для отримання мінімізованої функції "1"-ці або "0"-лі групуються в найбільші можливі прямокутні групи, в яких кількість комірок в групі має дорівнювати степеню 2. Наприклад, група може складатися з 4 комірок в лінію, 2 в висоту і 4 в ширину, 2 на 2 і так далі. Байдужий стан (зазвичай позначений X) групується тільки тоді, коли група отримана з його використанням більша ніж без нього. Комірки можуть бути використані більше ніж один раз тільки якщо завдяки цьому утворюється менша кількість груп. Кожна "1" або "0" має бути задіяна як мінімум в одній групі.

Існує 16 варіантів комбінації вхідних змінних, таким чином карта Карно має 16 позицій, і організована у вигляді решітки 4*4.

Двійкові цифри в карті показують вихід функції для будь-якої комбінації на вході. Таким чином 0 вписаний в верхню ліву комірку решітки через те, що $f = 0$ коли A = 0, B = 0, C = 0, D = 0. Зауважте, що значення впорядковані згідно з кодом Грея, значить рівно одна змінна змінюється між двома суміжними комірками.

Після конструювання карти Карно наше завдання знайти мінімальні терми для використання в кінцевому виразі. Ці терми знаходяться шляхом окреслення груп 1-ць в карті. Група має бути прямокутною і мати площу, що

дорівнює ступеню двійки (тобто 1, 2, 4, 8…). Прямокутник має бути максимально великим і без 0-в. Оптимальне групування в матриці позначене зеленим, червоним і синім. Зауважте, що групи можуть перетинатися. Зона перетинання червоної і зеленої групи позначена коричневим.

Решітка тороїдально зв'язна, що означає, що прямокутні групи обгортатися навколо країв, таким чином AD правильний терм, хоч і не частина мінімального набору, він покриває мінтерми 8, 10, 12 і 14.

Можливо важче уявити такий терм BD, який обгортається навколо всіх чотирьох вершин, він покриває такі терми 0, 2, 8, 10.

Розв'язання:

Коли карта Карно побудована і отримані групи, розв'язок може бути отриманий шляхом виключення надлишкових змінних використавши аксіоми булевої алгебри.

Для червоної групи:

1. Змінна A має одне й те саме значення (1) в кожній комірці групи, значить вона має бути включена в терм червоної групи.
2. Змінна B змінює своє значення, отже має бути виключена.
3. C завжди 0. Через те, що C 0, вона має бути обернена (записана із символом інверсії, C перед включенням.
4. D змінюється.
5. Таким чином перший терм виглядає AC .

Для зеленої групи ми бачимо, що A та B зберігають одне значення, а D змінюється. B - 0 і має бути обернене перед включенням. Отже другий терм AB.

Аналогічно, синя група дає BCD.

Розв'язки усіх груп об'єднуються в: AC+AB+BCD.

2.2.2. Метод Куайна

Метод Куайна - спосіб мінімізації функцій алгебри логіки. Представляє функції у вигляді ДНФ або КНФ з мінімальною кількістю членів і з мінімальним набором змінних.

Метод Куайна має чітко сформульований алгоритм здійснення окремих операцій і через це може бути використаний для реалізації на ЕОМ. Для проведення мінімізації цим методом вимагається досить багато часу через необхідність попарного порівняння одного з одним членів логічної функції. У випадку мінімізації за методом Куайна припускається, що початкова функція задана в ДДНФ або ДКНФ.

Досконáлою диз'юнктúвною нормáльною фóрмою (ДДНФ) булевої функції називається диз'юнкція тих конституент одиниці, які перетворюються в одиницю на тих самих наборах змінних, що й задана функція. ДДНФ повинна задовольняти наступним умовам:

– в ній немає однакових доданків;

– жоден із доданків не містить двох однакових співмножників;

– жоден із доданків не містить змінну разом із її запереченням;

– в кожному окремому доданку є як співмножник або змінна xi, або її заперечення для будь-якого i = 1, 2, …, n.

Для будь-якої функції булевої алгебри існує своя ДДНФ, причому тільки одна.

Сам метод проходить в два етапи:

1. Перехід від канонічної форми (ДДНФ або ДКНФ) до скороченої форми.

2. Перехід від скороченої до мінімальної форми.

Досконалою кон’юнктивною нормальною формою (ДКНФ) булевої функції називається кон’юнкція тих конституент нуля, які перетворюються в нуль на тих самих наборах змінних, що й задана функція. Також по аналогії з ДДНФ, будь-яка булева функція має одну ДКНФ (кількість її членів дорівнює

кількості нульових значень функції) і декілька КНФ. Можна навести такі властивості ДКНФ, що виділяють її з усіх КНФ:

– в ній немає однакових співмножників;

– жоден із співмножників не містить двох однакових доданків;

– жоден із співмножників не містить якої-небудь змінної разом з її запереченням;

– в кожному окремому співмножнику є як складова або змінна xi, або її заперечення для будь-якого i=1,2,…,n.

2.3. Опис використаних технологій та мов програмування

Програмний додаток написаний за допомогою методів візуального програмування на мові C# і скомпільований з розширенням exe. Для стабільної роботи програми слід використовувати Windows 7 та Windows 8.1.

2.3.1. Опис середовища розробки

Windows 7 дозволяє оптимальним чином виконувати різні вимоги користувачів. Підприємства можуть дозволяти своїм співробітникам збільшувати свою продуктивність, працюючи з офісу, з будинку, в дорозі або з філій. У новій системі поліпшена безпека і керованість, що знижує ризик втрати даних на комп'ютерах і зовнішніх жорстких дисках. Управління настільними системами спрощене, що полегшує розгортання Windows 7 і забезпечує безперебійну роботу. Windows 7 містить багато нових і вдосконалених можливостей [16].

Windows 7 дозволяє кінцевим користувачам працювати незалежно від їх місцезнаходження або місцезнаходження даних. Це прискорює їх роботу і скорочує час простою, оскільки Windows 7 покращує продуктивність і надійність. Користувачам не доведеться шукати інформацію в різних місцях, оскільки однією операцією пошуку можна проглянути увесь сайт SharePoint в

інтрамережі організації, а також файли на комп'ютерах користувачів. DirectAccess дозволяє мобільним користувачам просто і безпечно звертатися до корпоративних ресурсів, знаходячись поза офісом. Користувачі у філіях з повільними мережевими підключеннями можуть також підвищити свою продуктивність за допомогою засобу BranchCache в Windows 7, яке кешує часто використовувані файли і веб-сторінки.

Windows 7 ґрунтований на інфраструктурі безпеки, що забезпечує підвищену гнучкість при захисті комп'ютерів і даних. Окрім захисту внутрішніх жорстких дисків комп'ютера система шифрування дисків BitLocker тепер підтримує шифрування зовнішніх USB-накопичувачів і жорстких дисків і надає ключі відновлення для звернення до даних при необхідності. У випадку якщо необхідно забезпечити максимальну безпеку на підприємстві, можуть бути використати нові засоби блокування додатків для виконання на комп'ютерах кінцевих користувачів тільки дозволених програм, що також дозволяє зменшити ризик використання шкідливих програм.

Windows 7 полегшує управління і розгортання для настільних, портативних комп'ютерів і віртуальних середовищ. Поліпшені засоби управління образами і їх розгортання дозволяють додавати, видаляти і складати звіти по драйверах, мовних пакетах і оновленнях, а також розгортати образи системи на комп'ютери користувачів з меншим використанням пропускної спроможності мережі. Нові можливості автоматизації і сценаріїв на базі Windows PowerShell 2.0 знижують витрати на управління комп'ютерами і усунення неполадок. При використанні віртуалізації клієнтів Windows 7 полегшує управління образами віртуальних машин і надає більшу кількість функцій при роботі через видалене підключення.

Оновлюваний щорічно пакет Microsoft Desktop Optimization Pack доповнює систему новими можливостями для роботи у рамках підприємств. Завдяки поєднанню Windows 7 і Microsoft Desktop Optimization Pack підприємства можуть оптимізувати інфраструктуру комп'ютерів і досягти гнучкості, необхідної для вирішення своїх бізнес-завдань.

З появою ОС Windows 8 почалось створення нових пристроїв до значного зростання числа додатків, а також важливих поліпшень операційної системи і додатків. Були зібрані відомості про використання продукту і отримано безліч відгуків. Випущено сотні оновлень продукту і додатків.

Windows 8.1 стає початком, що дозволить з'явитися новому поколінню комп'ютерів, планшетів і різних промислових пристроїв, які повинні і розширюватимуть взаємодію з користувачем як для споживчого, так і для комерційного використання. Windows 8.1 не лише враховує відгуки користувачів, але і додає нові функції і можливості, що розширюють сенсорну взаємодію і потенціал мобільної роботи на комп'ютері. Windows 8.1 містить поліпшення таких важливих компонентів, як персоналізація, пошук, вбудовані застосування, взаємодія з Магазином Windows і робота в хмарі. Також Windows 8.1 включає виграшні можливості для бізнесу в таких аспектах, як управління і забез-печення безпеки [16].

Підключення до хмари. У Windows 8.1 файлів можна зберігати безпосередньо в SkyDrive, щоб завжди мати до них доступ (рис. 2.2).

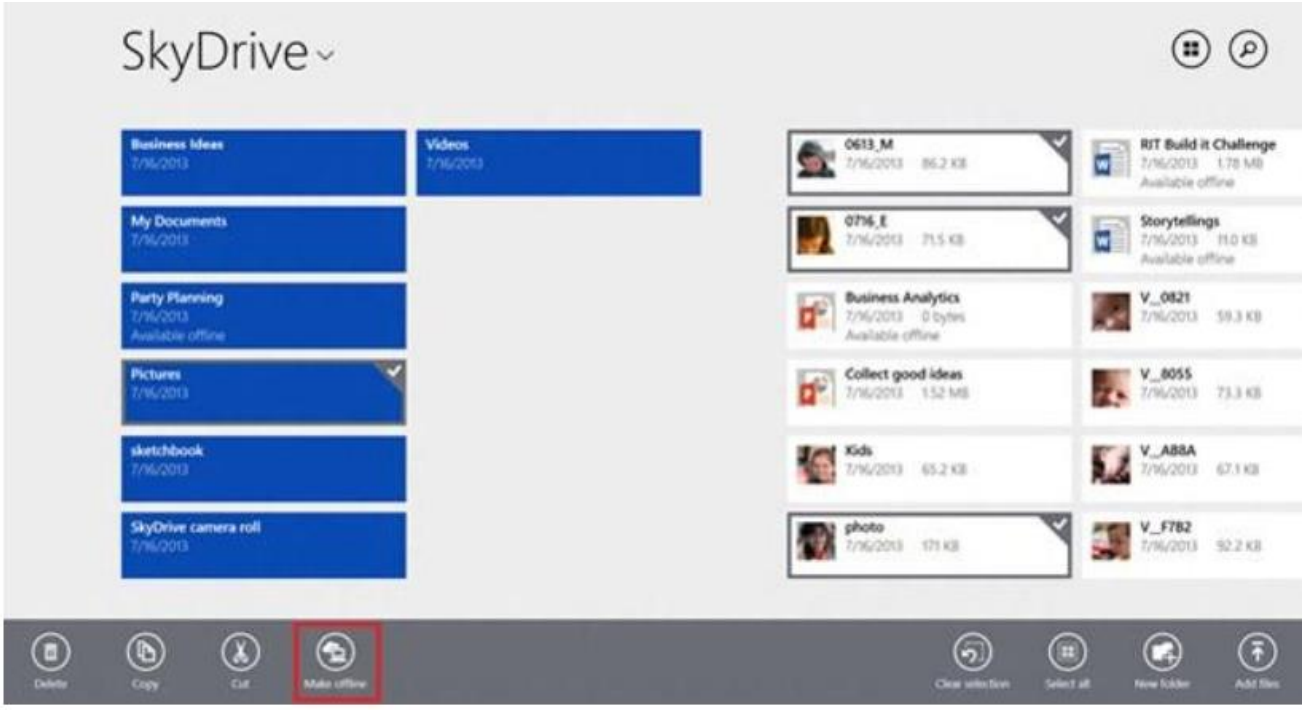

Рис. 2.2. Додаток SkyDrive

Новий додаток SkyDrive надає доступ до файлів, які збережені на пристрої або в хмарі, навіть при роботі в автономному режимі.

Параметри комп'ютера. Оновлені "Параметри комп'ютера" в Windows 8.1 пропонують доступ до усіх налаштувань без необхідності відкривати панель управління на робочому столі. За допомогою опції "Параметри комп'ютера" можна виконувати такі дії, як зміна дозволу екрану, налаштування параметрів живлення, перегляд типу і моделі пристрою, зміна ключа продукту, запуск оновлення Windows і приєднання до домена. Можна управляти звідси додатком

Framework Microsoft .NET надає:

– стійке загальномовне середовище виконання CLR (Common Language Runtime), яка входить до складу цієї платформи;

– засоби розробки застосувань на будь-якій з багатьох мов програмування, підтримуваних платформою .NET;

– відкриту модель програмування величезну бібліотеку класів .NET Framework. Ці класи містять багаторазово використовувані і код. Вони доступні в будь-якій мові програмування, підтримуваній платформою .NET;

– підтримку мережевої інфраструктури, побудованої на верхньому шарі стандартів Internet, внаслідок чого забезпечується високий рівень взаємодії між застосуваннями;

– підтримку нового промислового стандарту, а саме технології Web-служб. Технологія Web-служб надає новий механізм створення розподілених застосувань. По суті, вона є поширенням технології створення застосувань на базі компонентів і на сферу Internet;

– модель безпеки, яку програмісти можуть легко використовувати у своїх застосуваннях;

– потужні інструментальні засоби розробки.

Платформа .NET Framework складається із загальномовного середовища виконання (середовища CLR) і бібліотеки класів .NET Framework. Основою платформи .NET Framework є середовище CLR. Середовище виконання можна вважати агентом, який управляє кодом під час виконання і надає основні

служби, такі як управління пам'яттю, управління потоками і видалена взаємодія. При цьому середовищем накладаються умови строгої типізації і інші види перевірки точності коду, забезпечуюча безпека і надійність. Фактично основним завданням середовища виконання є управління кодом. Код, який звертається до середовища виконання, називають керованим кодом, а код, який не звертається до середовища виконання, називають некерованим кодом. Бібліотека класів є комплексною об'єктно-орієнтованою колекцією повторно використовуваних типів, які застосовуються для розробки застосувань, – починаючи із звичайних застосувань, що запускаються з командного рядка, і застосувань з графічним інтерфейсом (GUI) і закінчуючи застосуваннями, що використовують останні технологічні можливості ASP.NET, такі як веб-форми і веб-служби XML.

Платформа .NET Framework може розміщуватися некерованими компонентами, які завантажують середовище CLR у власні процеси і запускають виконання керованого коду, створюючи таким чином програмне середовище, що дозволяє використовувати засоби як керованого, так і некерованого виконання. Платформа .NET Framework не лише надає декілька базових середовищ виконання, але також підтримує розробку базових середовищ виконання незалежними виробниками[18].

На рис. 2.3 демонструється взаємозв'язок середовища CLR і бібліотеки класів з призначеними для користувача застосуваннями і усією системою. Також показано, як керований код працює в межах ширшої архітектури.

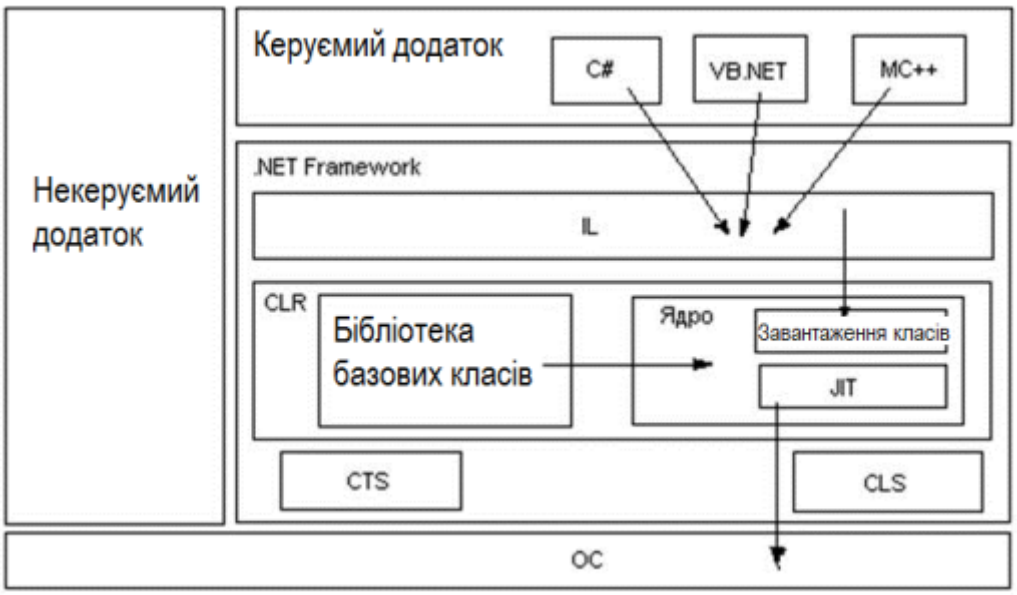

Рис. 2.3. Взаємозв'язок середовища CLR і бібліотеки класів .Net

Існує ряд засобів, які підтримуються .NET, але не підтримуються C#, і, можливо (наприклад, деякі випадки перевантаження операцій). Проте оскільки мова C# призначена для застосування на платформі .NET, важливо мати уявлення про .NET Framework.

Центральною частиною каркаса .NET є його загальномовне виконуюче середовище, відоме як Common Language Runtime (CLR) або .NET runtime. Код, що виконується під управлінням CLR, часто називають керованим кодом. З точки зору програмування під терміном виконуюче середовище може розумітися колекція зовнішніх служб, які вимагаються для виконання скомпільованої одиниці програмного коду.

У складі .NET пропонується ще одне виконуюче середовище. Головна відмінність між виконуючим середовищем .NET і згаданими вище середовищами, полягає в тому, що виконуюче середовище .NET забезпечує єдиний чітко певний рівень виконання, який здатні використовувати усі сумісні з .NET мови і платформи.

Проте перш ніж код зможе виконуватися CLR, будь-який початковий текст (на C# або іншій мові) має бути скомпільований. Компіляція в .NET складається з двох кроків:

– перший крок – компіляція початкового коду в Microsoft Intermediate Language (IL);

– другий крок – компіляція IL в специфічний для платформи код за допомогою CLR[19].

2.3.2. Опис мови програмування

Мова C# і пов'язане з ним середовище .NET Framework можна без перебільшення назвати найзначнішою з пропонованих нині технологій для розробників. Середовище .NET є таким середовищем, яке було створене для того, щоб в ній можна було розробляти практично будь-яке застосування для запуску в Windows, а C# є мовою програмування, яка була спеціально створена для використання в .NET Framework. Наприклад, із застосуванням C# і .NET Framework можна створювати динамічні веб-сторінки, застосування Windows Presentation Foundation, веб-служби XML, компоненти для розподілених застосувань, компоненти для доступу до баз даних, класичні настільні застосування Windows і навіть клієнтські застосування нового інтелектуального типу, що мають можливості для роботи в оперативному і автономному режимах.

C# є мовою, призначеною не лише для написання застосувань, здатних працювати в Інтернеті і в мережі. Він надає засоби для кодування практично будь-якого типу програмного забезпечення або ком-понентів для платформи Windows. Мову C# і середовище .NET привели до революційних змін в способі написання розробниками програм і зробили програмування застосувань для Windows набагато простішим, ніж коли-небудь[14].

C# – це відносно нова мова програмування, яка характеризується двома наступними перевагами:

1. C# спроектований і розроблений спеціально для застосування з Microsoft .NET Framework (розвиненою платформою розробки, розгортання і виконання розподілених застосувань).

2. C# – мова, заснована на сучасній об'єктно-орієнтованій методології проектування, при розробці якого фахівці з Microsoft спиралися на досвід створення подібних мов, побудованих відповідно до запропонованих близько 20 років тому об'єктно-орієнтованими принципами.

Треба підкреслити те важлива обставина, що C# – це повноцінна мова програмування. Хоча він і призначений для генерації коду, що виконується в середовищі .NET, сам по собі він не є частиною .NET. Існує ряд засобів, які підтримуються .NET, але не підтримуються C#, і, можливо, вас здивує, що є також засоби, підтримувані C# і не підтримувані .NET (наприклад, деякі випадки перевантаження операцій). Проте оскільки мова C# призначена для застосування на платформі .NET, вам, як розробникові, важливо мати уявлення про .NET Framework, якщо ви хочете ефективно розробляти застосування на C#.

Поточною є версія C# 4.0. Ця версія міцно спирається на три попередні основні версії C#, доповнюючи їх цілим рядом нових засобів. Ймовірно, найважливішими серед них є іменовані і необов'язкові аргументи. Зокрема, іменовані аргументи дозволяють зв'язувати аргумент з параметром по імені. А необов'язкові аргументи дають можливість вказувати для параметра використовуваний за умовчанням аргумент. Ще одним важливим новим засобом є тип dynamic, вживаний для оголошення об'єктів, які перевіряються на відповідність типів під час виконання, а не компіляції. Крім того, коваріантність і контраваріантність параметрів типу підтримується завдяки новому застосуванню ключових слів in і out. Тим, хто користується моделлю COM взагалі і прикладними інтерфейсами Office Automation API зокрема, істотно спрощений доступ до цих засобів. В цілому, нові засоби, впроваджені у версії C# 4.0, сприяють подальшій раціоналізації програмування і підвищують практичність самої мови C#.

Абсолютно усім .NET – програмістам важливо уміти працювати на мові, що віддається перевага ними, з об’єктно-орієнтованими типами з CTS: класи, інтерфейси, делегати[11].

У кожному сумісному з .NET мові підтримується, як мінімум, поняття типу класу (class type), яке грає центральну роль в об'єктно-орієнтованому програмуванні (ООП). Кожен клас може включати будь-яку кількість членів (таких як конструктори, властивості, методи і події) і точок даних (полів). У C# класи оголошуються за допомогою ключового слова class:

```
class mySum
{
public int Sum (int x, int y)
{ return x+y; }
}
```

Інтерфейси є не більше ніж просто іменованою колекцією визначень абстрактних членів, які можуть підтримуватися (тобто реалізуватися) в цьому класі або структурі. У C# типи інтерфейсів визначаються за допомогою ключового слова interface :

```
public interface ICommandSource
{       void CommandParameter();}
```

Інтерфейси реалізуються в класах або структурах унікальним чином, вони дозволяють діставати доступ до додаткових функціональних можливостей за рахунок додавання просто посилання на них в поліморфній формі.

Делегати (delegate) являються .NET – еквівалентом безпечних відносно типів покажчиків функцій в стилі C. Головна відмінність поля-гає в тому, що делегат в .NET є класом, який наслідує від System.MulticastDelegate, а не просто покажчик на якусь конкретну адресу в пам'яті. У C# делегати оголошуються за допомогою ключового слова delegate.

Делегати дуже зручні, коли вимагається забезпечити одну суть можливістю перенаправляти виклик іншої суті і утворювати основу для архітектури обробки подій .NET. Делегати мають внутрішню підтримку для групової адресації (тобто пересилки запиту відразу безлічі одержувачів) і асинхронного виклику методів (тобто виклику методів у вторинному потоці).

2.3.3. Формування графічного інтерфейсу

Windows Forms – це технологія інтелектуальних клієнтів для платформи .NET Framework, набір керованих бібліотек, що спрощують виконання поширених завдань, таких як читання і запис у файловій системі. При використанні середовища розробки, як MS Visual Studio, можна створювати застосування інтелектуальних клієнтів Windows Forms, які відображують інформацію, запрошують введення від користувачів і взаємодіяти з видаленими комп'ютерами по мережі[18].

Windows Forms включає широкий набір елементів управління, які можна додавати на форми: текстові поля, кнопки, списки, що розкриваються, перемикачі і навіть веб-сторінки. Якщо існуючий елемент управління не задовольняє потребам, в Windows Forms можна створити призначені для користувача елементи управління за допомогою класу UserControl.

До складу Windows Forms входять багатофункціональні елементи призначеного для користувача інтерфейсу, що дозволяють відтворювати можливості таких складних застосувань, як Microsoft Office. Використовуючи елементи управління ToolStrip і MenuStrip, можна створювати панелі інструментів і меню, що містять текст і малюнки, підміню і інші елементи управління, такі як текстові поля і поля із списками.

Формування графічного інтерфейсу застосувань Windows Будь-яке застосування завжди має набір засобів для взаємодії з користувачем – інтерфейс користувача. За допомогою інтерфейсу застосування користувач може описати вирішувану задачу, виконати її рішення, зберегти отримані результати на зовнішньому пристрої. Раніше в посібнику при розгляді прикладів використовувався консольний інтерфейс (Console User Interface – CUI).

Застосування, що використовують графічний інтерфейс називаються Windows застосуваннями (чи Windows Forms Application) і для взаємодії з користувачами вони використовують клас Form. Оскільки в платформі .Net усе

засновано на об'єктно-орієнтованому підході, то і графічний інтерфейс реалізований у вигляді набору спеціальних класів FCL з простору імен System.Windows.Forms[15].

Загальна схема застосувань, що розробляються для ОС Windows наведена на рис. 2.4.

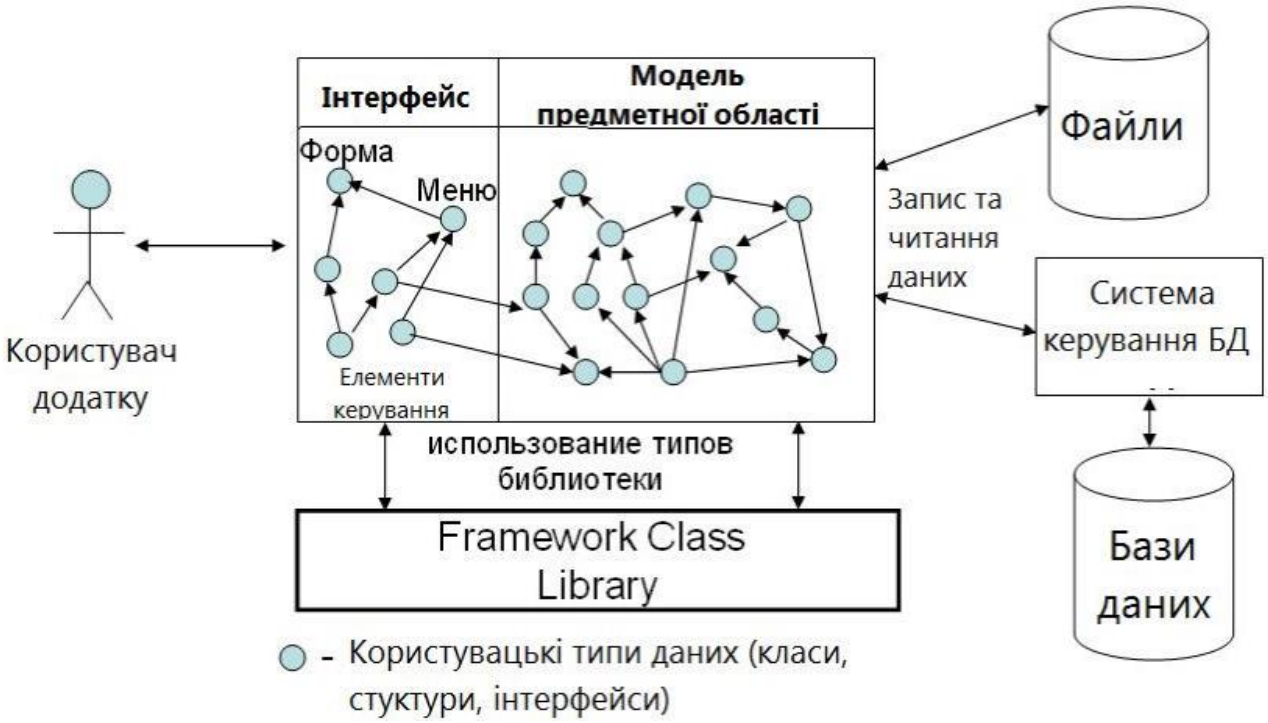

Рис. 2.4. Структура типового застосування Windows

У відповідність з цією схемою програма складається з класів (типів), які умовно розділені на класи, що реалізовують взаємодію застосування з користувачем (інтерфейс користувача), і класи, що моделюють предметну область вирішуваної задачі. Окрім власних класів застосування використовує класи бібліотеки платформи FCL. Результати роботи застосування зазвичай зберігаються на зовнішніх пристроях, у файлах або базах даних.

Графічний інтерфейс користувача реалізує і підтримує операційна система. Основним поняттям GUI є "вікно" (window). На екрані вікна є деякими ділянками (зазвичай прямокутні) в яких виконується введення і виведення інформації і де можуть розміщуватися інші вікна (дочірні), за допомогою яких

виконується управління програмою.

Створенням і управлінням вікнами займається сама операційна система, яка першою отримує інформацію від усіх пристроїв комп'ютера (натиснення клавіші, рух "миші", вступ даних по мережі). Про усіх цих події ОС розсилає повідомлення застосувань, вікна яких є активними у нинішній момент.

Для цього в усіх Windows застосуваннях, що працюють з вікнами, операційна система, створює спеціальні колекції типу FIFO (черги повідомлень), в які вона записує усі повідомлення про події, що сталися в комп'ютері, і пов'язаних з вікнами цього застосування. Застосування постійно стежить за повідомленнями, що поступають в цю чергу, і виконує їх обробку.

Вікно застосування включає наступні основні частини:

– рядок заголовка вікна, в якій розміщуються такі стандартні елементи управління вікном, як: системне меню, що з'являється при клацанні на іконці застосування в лівій частині рядка; три системні кнопки для згортання вікна, розкриття вікна на увесь екран і закриття вікна;

– рядок меню застосування – набір команд (пунктів меню), при виборі яких застосування виконуватиме різні завдання; рядок меню може розташовуватися в різних частинах вікна, найчастіше знаходиться в його верхній частині;

– інструментальні смуги (рядки з наборами спеціальних дочірніх вікон – елементів управління), за допомогою яких також можна викликати виконання різних завдань застосування; інструментальних смуг в інтерфейсі застосування може бути багато і вони можуть розміщуватися в різних частинах вікна;

– різні елементи графічного інтерфейсу і виконувати малювання;

– рядок стану, в який виводяться повідомлення що інформують користувача про хід роботи застосування[11].

Застосування зазвичай використовує велику кількість вікон. Усі вікна можна розділити на групи:

– основні вікна, які використовуються для представлення усього застосування; вони включають основні елементи інтерфейсу і ініціюють

створення інших вікон;

– діалогові вікна, які призначені для отримання інформації і запуску на виконання різних допоміжних завдань застосування;

– елементи управління (control), це дочірні вікна, які використовуються для виконання елементарних операцій по відображенню інформації (наприклад, текстові вікна – TextBox, вікна із списками рядків – ListBox, вікна із зображеннями PictureBox) або для отримання деяких команд користувача (наприклад, натиснення кнопок "миші", кнопок Button, пунктів меню).

Платформа .Net надає набір об'єктно-орієнтованих засобів для зручної і простої реалізації усіх частин графічного інтерфейсу.

У платформі .Net для реалізації графічного інтерфейсу використовуються різні технології, такі як: Windows Forms і Windows Presentation Foundation (WPF). Розглянемо тільки технологія Windows Forms. Ця технологія включає безліч типів (класи, структури, перерахування, делегати), які об'єднані в два основні простори імен System.Windows.Forms (для реалізації елементів інтерфейсу) і System.Drawing (для малювання в клієнтській області)[14].

Основними елементами графічного інтерфейсу є спеціальні класи, звані елементами управління (ЕУ), які мають дві особливості:

1. Реалізують роботу з різними типами вікон ОС Windows.
2. Підтримують роботу в двох режимах:
   – режим проектування (design mode), в якому з ними працює середовище розробки;
   – режим виконання (run mode), в якому виконується взаємодія користувачів з ЕУ.

Базовим класом для усіх елементів управління є клас Control, що реалізовує саму базову функціональність. Клас Control задає важливі властивості, методи і події, успадковані усіма його нащадками. Усі класи елементів управління є спадкоємцями класу Control (рис. 2.5). Базовий клас Control містить досить великий інтерфейс (79 властивостей, 56 методів, 67

подій), який доступний в усіх похідних класах.

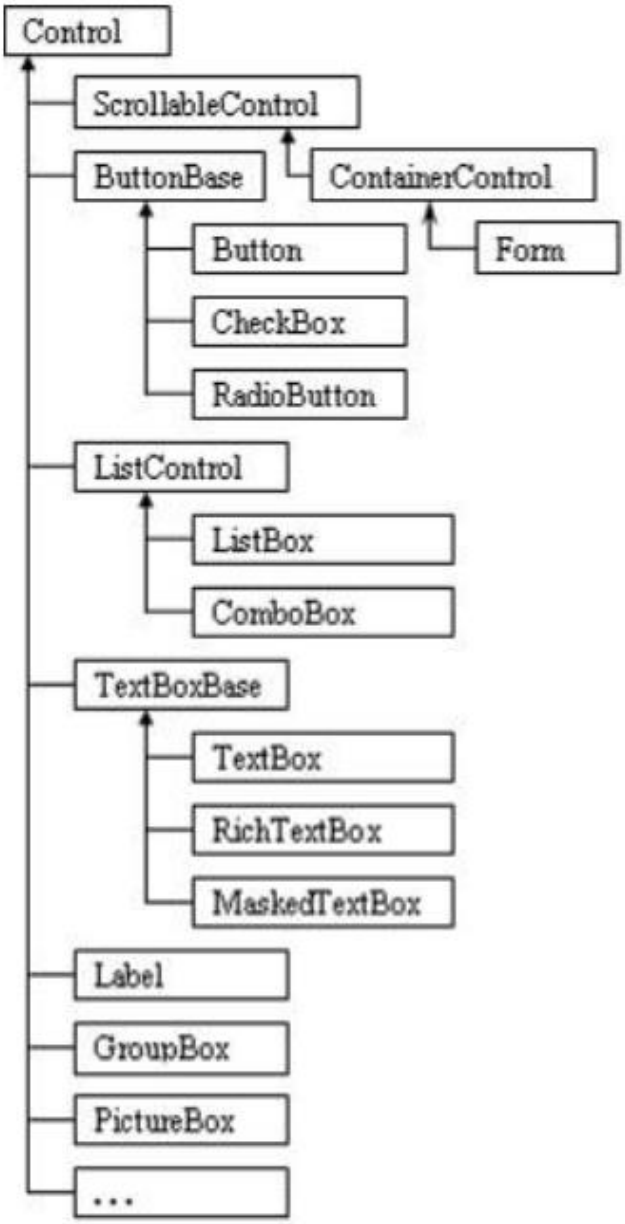

Рис. 2.5. Ієрархія класів елементів управління

У бібліотеці класів FCL усі події описуються за допомогою одного типу делегата, який має фіксовану сигнатуру з двома параметрами і що не повертає значення:

public delegate void <ім'я_делегата>(object sender, <тип_параметрів> args);

Перший параметр делегата задає посилання на об'єкт, який ініціює подію. Другий параметр args задає посилання на параметри, пов'язані з виниклою подією, передавані обробникові. Тип цього параметра повинен задаватися класом опису параметрів події EventArgs, який міститься в .Net Framework або похідним від нього класом (наприклад, PaintEventArgs, MouseEventArgs і тому подібне). Якщо обробникові ніяких додаткових параметрів не передаються, то слід просто вказати клас EventArgs, передаючи null як фактичного параметра при включенні події[11].

Взаємодія користувача краще всього описується у вигляді різних подій, які ініціюють ЕУ (на основі повідомлень ОС про дії користувача) і на які відповідає (обробляє) застосування. Найбільш часто використовуваними подіями є наступні:

– Click – клацання лівої кнопки "миші" в зоні вікна;

– DoubleClick – два клацання лівої кнопки "миші" з інтервалом менше деякого заданого значення;

– KeyDown – натиснення клавіші клавіатури;

– KeyPress – натиснення і відпуск клавіші, в результаті яких в програму передається деякий символ;

– Validating – перевірки введених даних;

– Paint – необхідно перемальовувати клієнтську область.

Події від пристрою "миша", такі, як Click, DoubleClick, MouseDown, MouseUp, MouseEnter, MouseLeave і MouseHover, пов'язані з різними діями користувачів над областю ЕУ.

Для подій Click і DoubleClick передається параметр типу EventArgs, а для подій MouseDown і MouseUp передається параметр типу MouseEventArgs, який містить таку корисну інформацію (властивості класу), як поточні координати курсора в клієнтській області, опис натиснутої кнопки, кількість натиснень кнопки, кількість клацань при обертанні колеса "миші".

Події клавіатури працюють аналогічно: кількість передаваної інформації залежить від типу оброблюваної події. Наприклад, для події KeyPress в метод

обробки події передається параметр KeyPressEventArgs, який містить властивість KeyChar, – значення типу char, яке представляє символ натиснутої клавіші.

Властивість Handled використовується для визначення того, чи було оброблено цю подію. Якщо властивості Handled задано значення true, то ця подія не передаватиметься ОС для стандартної обробки. Події KeyDown або KeyUp більше підходять для обробки, якщо вимагається отримати більше інформації про натиснуту клавішу, оскільки вони отримують параметр KeyEventArgs. Параметр KeyEventArgs включає властивості про те, які клавіші Ctrl, Alt або Shift були натиснуті. Властивість KeyCode повертає значення перерахування Keys, яке вказує на віртуальний код натиснутої клавіші. На відміну від KeyPressEventArgs.

KeyChar властивість KeyCode передає віртуальний код будь-якої натиснутої клавіші клавіатури, а не алфавітно-цифровий символ клавіші.

Властивість KeyData повертає значення перерахування Keys, а також стан додаткових клавіш. Наприклад, чи була натиснута клавіша Shift або Ctrl. Властивість KeyValue містить ціле значення перерахування Keys. Властивість Modifiers містить значення Keys, які відповідають кодам додатково натиснутих клавіш. Якщо було натиснуто декі-лька клавіш, то вони об'єднуються за допомогою операції OR. Події, пов'язані з клавішею ініціюються в наступному порядку:

– KeyDown;
– KeyPress;
– KeyUp.

Події Validating, Validated, Enter, Leave, GotFocus і LostFocus мають відношення до отримання ЕУ фокусу введення (коли ЕУ стає активним) або втрати фокусу. Це відбувається, коли користувач натискає клавішу Tab для переходу до потрібного ЕУ або вибирає цей елемент за допомогою "миші". Здається, що події Enter, Leave, GotFocus і LostFocus дуже схожі по виконуваній роботі. Події GotFocus і LostFocus є подіями нижчого рівня, які пов'язані з

WM_SETFOCUS і WM_KILLFOCUS повідомленнями ОС. Зазвичай краще використовувати події Enter і Leave. Події Validating і Validated виникають при перевірці значення в ЕУ. Вони отримують параметр типу CancelEventArgs. З його допомогою можна перервати наступні події, якщо задати властивості Cancel значення true. Якщо розробник задає власний код перевірки введених значень і перевірка виявилася не успішною, то можна задати властивості Cancel значення true і ЕУ не втрачатиме фокус введення (не виконуватиметься перехід до наступного ЕУ форми). Подія Validating виникає в ході перевірки, а подія Validated виникає після виконання перевірки. Ці події виникають в на-ступному порядку:

1. Enter;
2. GotFocus;
3. Leave;
4. Validating;
5. Validated;
6. LostFocus[15].

2.3.4. Побудова застосунку в середовищі MS Visual Studio 2019

Для побудови застосувань C# можна вибирати відповідний варіант з безлічі IDE – середовищ; одні IDE – середовища пропонуються компанією Microsoft, а інші поступають від незалежних постачальників. Нині численні IDE – середовища виробництва Microsoft є безкоштовними.

Для побудові програмного забезпечення .NET для ОС Windows (7, 8.x або 10), можна скористатися варіантами Visual Studio 2019: Community, Professional, Enterprise. Значніша різниця пов'язана з моделлю ліцензування. Редакція Community ліцензується для використання з проектами з відкритим кодом, в учбових установах і на малих підприємствах.

Переваги середовища MS Visual Studio 2019 :

– розробка для WPF, WinForms, ASP.NET, універсальної платформи

Windows, Win32, Android, iOS і багатьох інших платформ в єдиному інтерфейсі IDE з усіма потрібними засобами;

– прискорення розробки а саме завантаження проектів і підвищена продуктивність: можна працювати на мовах, яким віддаються перевага, таких як C і C++, C#, Visual Basic, F#, Python і багато інших;

– підвищення ефективності: редагування коду, навігація і відладка спрощені завдяки таким функціям, як " Change and continue" в XAML, поліпшеному IntelliSense, вдосконаленому рефакторингу кода, динамічному аналізу коду і можливості відкрити будь-яку теку, навіть якщо у немає проектів або рішень;

– сучасна веб-розробка за допомогою C#, Visual Basic, F#, C++, Python, Node.js и HTML/JavaScript, ASP.NET і багатьох інших повнофункціональних засобів програмування;

– створення застосувань і ігор для Windows за допомогою нових потужних функцій платформи, таких як аналіз і відладка призначеного для користувача інтерфейсу, емулятори мобільних пристроїв з Windows 10 і інструменти графіки DirectX;

– – динамічна екосистема – доступ до тисяч розширень, яких немає в Express, у тому числі GitHub, ReSharper і Visual Assist. І також Web Essentials[20].

2.4. Опис структури системи та алгоритмів її функціонування

2.4.1. Опис використаних алгоритмів

Відповідно постановки задачі та опису функціональних можливостей був спроектований та розроблений програмний додаток мінімізації мулевих функцій методом Куайна- Мак-Класкі та візуалізації обраних мінтерм - карт Карно.

2.4.1.1. Реалізація метода Куайна - Мак-Класкі

Метод Куайна - Мак-Класкі (метод простих імплікант) - табличний метод мінімізації булевих функцій розроблений Уілардом Куайном і Едвардом Мак-Класкі. Функціонально ідентичний карті Карно, але таблична форма робить його ефективнішим для використання в комп'ютерних алгоритмах.

Алгоритм Куайна-Мак-Класкі ґрунтується на теоремі Куайна. Скорочена ДНФ БФ може бути побудована з її досконалою ДНФ шляхом застосування операцій простого склеювання і поглинання. Схема роботи метода наведена на рис. 2.6 .

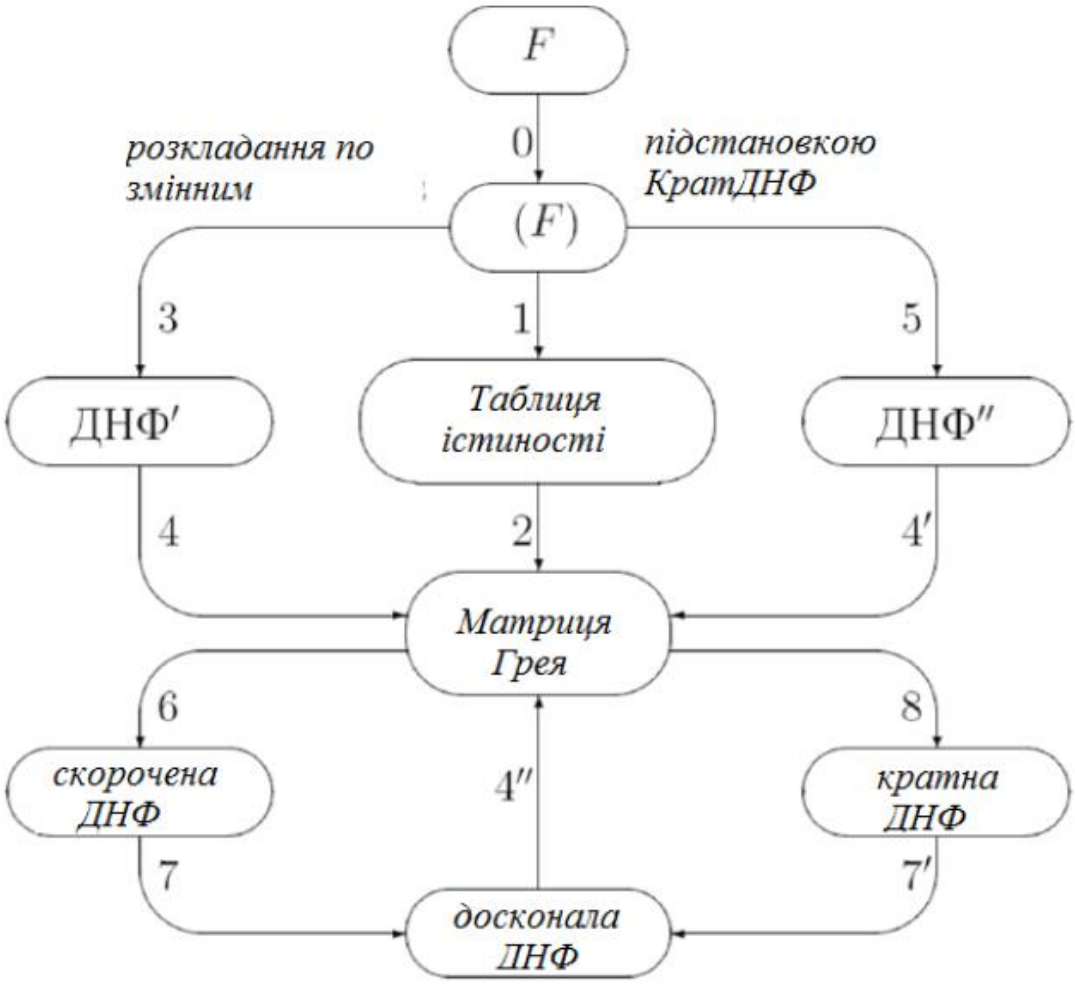

Рис. 2.6. Схема виконання мінімізації методом Куайна-Мак-Класкі

Практичне використання маніпуляцій алгебри, діаграм Венна або

відображень Карно для мінімізації булевих виразів обмежено проблемами з декількома змінними. Процедура Куайна-Ма-Класки для мінімізації може бути реалізована у вигляді комп'ютерного алгоритму та її застосовують до виразів з будь-яким числом змінних[22].

Приклад: заданий логічний вираз F (p, q, r, s, t) = m (0,1,3, 4,6,11,14,15,16,18,24,27,28,31) звести до мінімуму, використовуючи процедуру Куайна-Мак-Класкі. В даному прикладі є 5 змінних p, q, r, s і t, які дають нам 32 можливих мінтерми (0 – 31).

Алгоритм рішення:

Крок 1. Розподілити мінтерми на групи, ґрунтуючись на кількості одиниць в їх двійкових представленнях (рис. 2.7).

0	00000	no 1's
1	00001	one 1's
4	00100	
16	10000	
3	00011	two 1's
6	00110	
18	10010	
24	11000	
11	01011	three 1's
14	01110	
28	11100	
15	01111	four 1's
27	11011	
31	11111	five 1's

Рис. 2.7. Розподіл мінтерм на групи

Крок 2. Порівняти сусідні групи і замінити пари термів, які мають один біт (2.8). Ці терми називаються суміжними. Можна замінити будь-які дві

суміжні терми одним термом, який містить тире в місці невідповідного біта. Обов'язково відмітити мінтерми, які використовуються для створення загальних термів, щоб їх можна було видалити із списку.

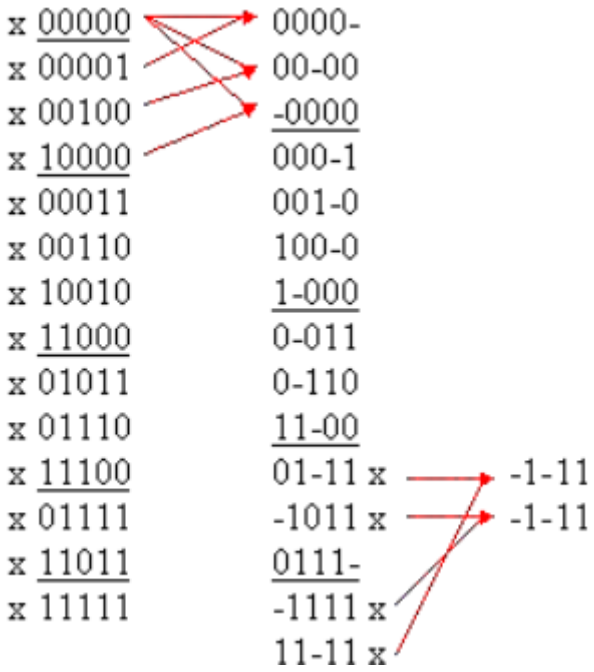

Рис. 2.8. Визначення суміжних мінтерм

Крок 3. Повторити пошук, поки не будуть знайдені нові суміжні терми. Два терми, які вже містять тире, вважаються суміжними, тільки якщо усі позиції тире співпадають, і усі літеральні позиції, окрім однієї, містять однакове значення.

Крок 4. Видалити дублікати і перерахувати усі унікальні терми, що залишилися. Вказати, які з первинних мінтерм охоплюються термінами, що залишилися. Їх називають імплікантами, оскільки вони мають на увазі існування мінтермів в початковому виразі (рис. 2.9).

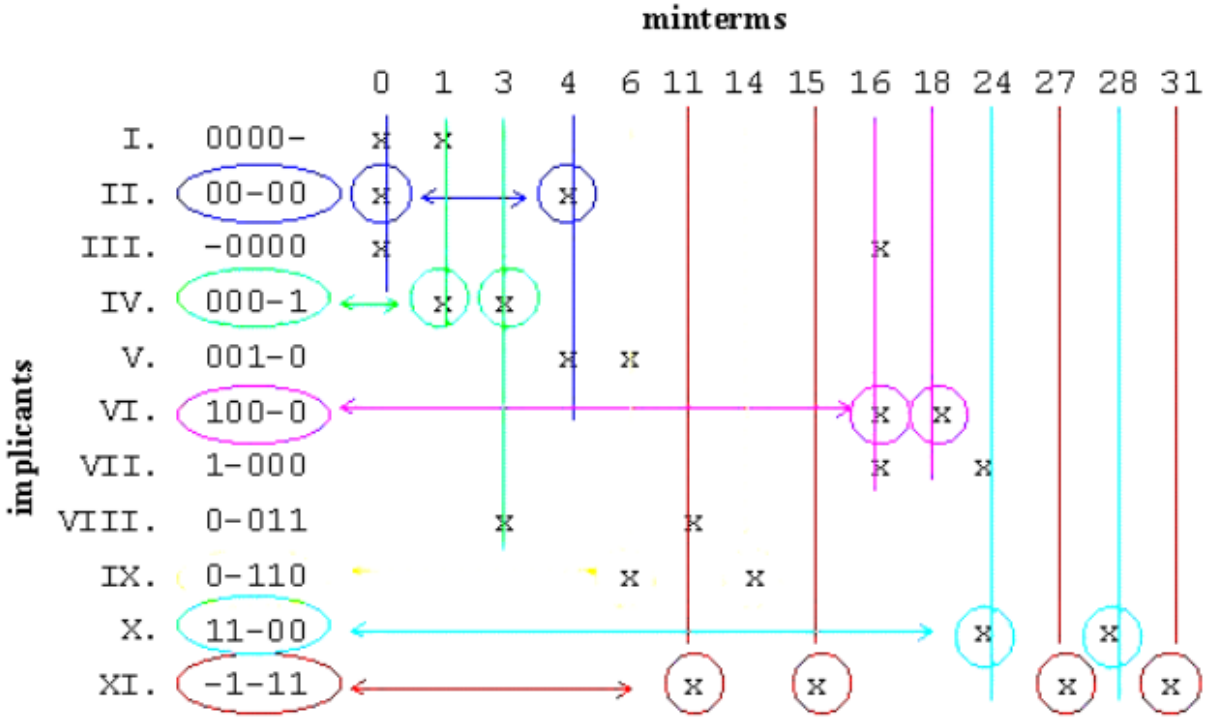

Рис. 2.9. Визначення імплікант

Знаходимо мінімальне покриття усіх терм (конституєнт одиниці) простими імплікантами. При цьому шукається така мінімальна сукупність простих імплікант, які спільно покривають усі терми початкової функції. Факт покриття відзначається в клітині матриці символом * (зірочка) у разі, коли імпліканта покриває відповідну конституєнту (є її власною частиною). З усіх простих імплікант вибираються спочатку тільки такі, які тільки одні покривають терми (в колонці матриці тільки один символ покриття), потім проводиться перебір[22].

Крок 5. Первинні імпліканти VI, IX, X і XI наведені на рис. 2.9, є єдиними, які охоплюють мінтерми (minterms) 18, 14, 28 і 31 відповідно. Ці імпліканти мають бути включені у список обраних. Їх називають основними первинними імплікантами.

Крок 6. Далі обрати мінімальну кількість імплікант, що залишилися, щоб охопити усі мінтерми.

Крок 7. Використати обраний набір простих імплікант для генерації

спрощеного логічного виразу (рис. 2.10).

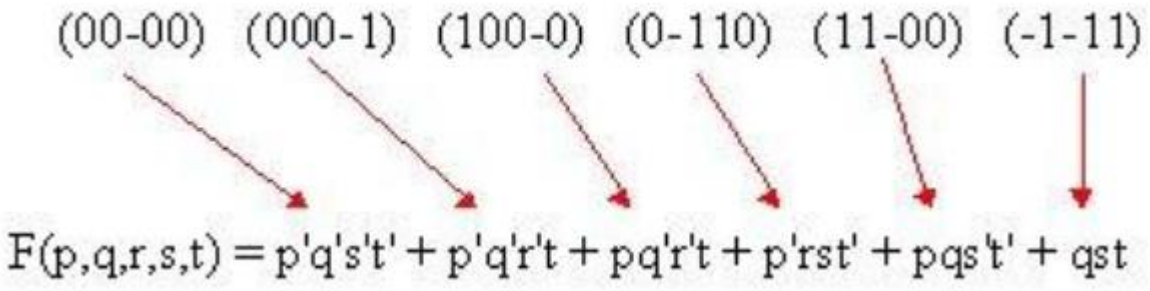

Рис. 2.10. Обраний список простих імплікант

У своєму первинному виді F() складався з 14 п'ятизначних мінтермів.

2.4.1.2. Мінімізація методом мінімізуючих карт Карно

Карти Карно – це графічне представлення таблиці істинності. Карти Карно налічують стільки клітинок, скільки рядків є в таблиці істинності.

Основу мінімізації за допомогою карт Карно складають такі положення:

1. Дві одиниці, які знаходяться в сусідніх клітинках карти можуть бути замінені однією кон’юнкцією, яка містить на одну змінну менше.

2. Якщо сусідніми є дві пари одиниць, то така група змінюється на кон’юнкцію, яка містить на дві змінних менше, відповідно якщо сусідніми є 2n одиниць, то така група може бути замінена кон’юнкцією, яка містить на n змінних менше.

3. Сусідніми є клітинки розміщенні поряд по горизонталі і вертикалі, а також клітинки, які знаходяться на протилежних границях карти Карно.

4. Поєднувати можна тільки 2n одиниць за принципом квадрату, прямокутнику або тору.

Метод мінімізуючих карт, також як і метод Куайна-Мак-Класкі, відноситься до точних методів. Класичний метод Куайна-Мак-Класкі при великому числі змінних вимагає значних тимчасових витрат, оскільки

вимагається визначити усі первинні імпліканти функції. Існує група методів мінімізації, що дозволяють знаходити точне рішення задачі, уникаючи отримання усіх первинних імплікант[7].

Метод мінімізуючих карт найбільш зручний метод мінімізації БФ при числі змінних – 6. Мінімізуюча карта – це розгортка на площині n – мірного куба (карта Карно). Клітини карти відповідають вершинам куба, координати клітин співпадають з координатами вершин. Карта Карно заповнюється так: якщо на якомусь наборі змінних БФ має значення 1, у відповідній клітині записується 1. Значення 0 зазвичай на картах не проставляється[7]. Приклад карти Карно для п'яти змінних (n=5) наведений на рис. 2.11.

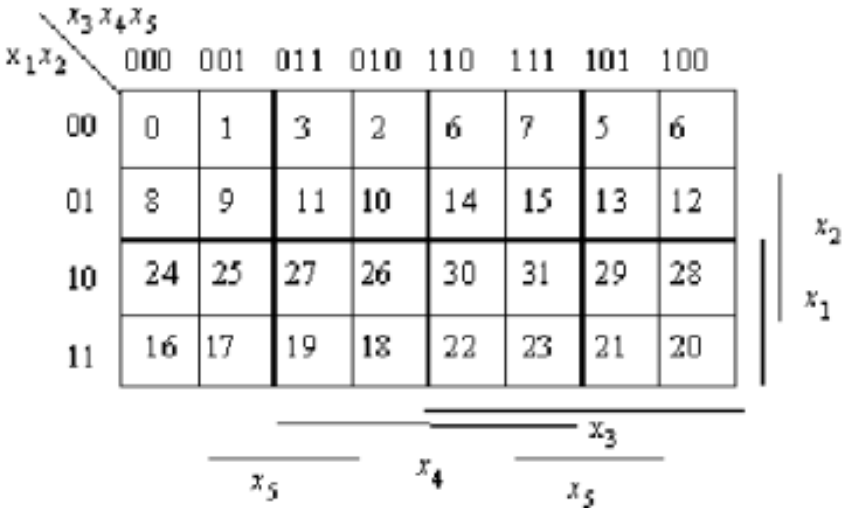

Рис. 2.11. Карта Карно для функції п'яти змінних

У разі шести змінних знадобиться вже чотири карти Карно для чотирьох змінних. Ця карта складається з 64 клітин (рис. 2.12).

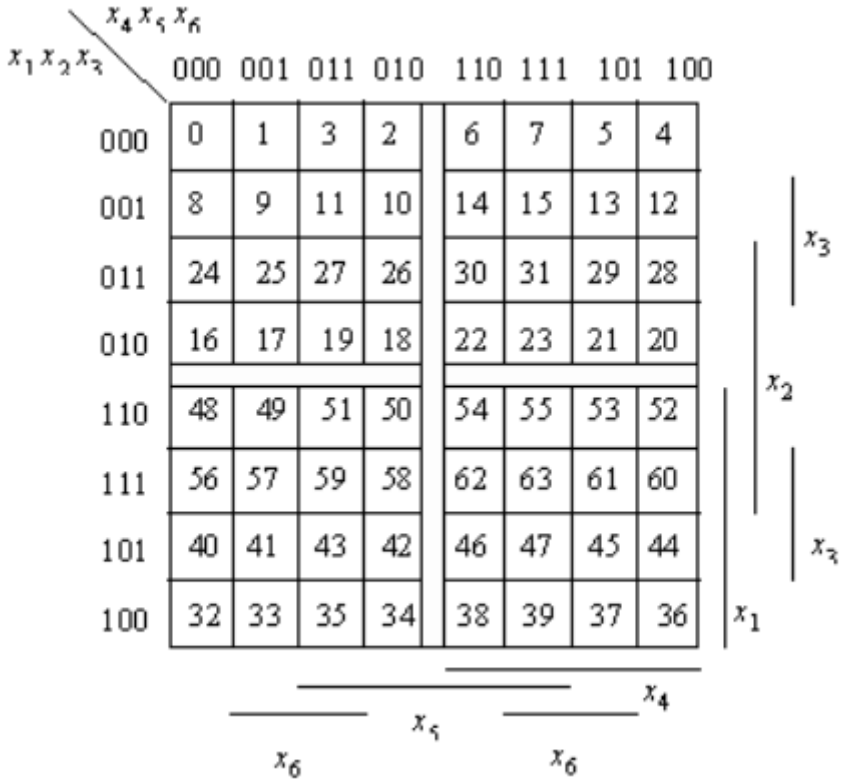

Рис. 2.12. Карта Карно для функції шести змінних

2.4.2. Структура програмного додатку

Внутрішня структура програмного проекту розбивається на декілька блоків (рис. 2.13):

– блок даних;

– блок представлення даних;

– блок обробки даних;

– інтерфейс.

Рис. 2.13. Внутрішня структура програмного проекту

Використання такого виду блоків і взаємозв'язків між ними пояснюється тим, що при реалізації застосування не повинно бути сильного зв'язку між бізнес-логікою застосування, тобто структурованим описом предметної області застосування, і бізнес-фасадом застосування, тобто інтерфейсом, що відображує предметну область на форму, видиму користувачем.

У блоці даних програмний додаток отримує з візуальних компонентів форми вхідні дані. Створення окремого модуля для отримання даних дозволяє позбавитися залежності логіки додатка від джерела даних. Це дозволяє використовувати в ролі сховища даних будь-які інші джерела, такі як файли. Крім того, змінивши сховище даних, змін зажадає тільки сам блок даних.

У блоці обробки даних знаходиться уся бізнес-логіка застосування. У цьому блоці здійснюється реалізація усіх алгоритмів вирішуваних завдань. Здійснюється підготовка внутрішніх даних для надання їх по запиту користувача. Так само за допомогою цього блоку здійснюється взаємодія між різними режимами роботи застосування. Для зручності подальшого

розширення програмного додатку, кожен етап алгоритму повинен записуватися у вигляді відповідної йому функції, що дозволить при додаванні в застосування нового алгоритму використовувати вже наявні функції і писати тільки необхідні частини додатку.

Блок представлення даних забезпечує коректне представлення даних на формі. Цей блок відповідає за прив'язку даних і оновлення цих елементів управління при роботі з даними користувача.

Інтерфейс програмного додатку призначений для взаємодії користувача і бізнес-логіки цього застосування. Інтерфейс є елементами візуального представлення даних, забезпечуючих зручну і інтуїтивно зрозумілу роботу з даними.

Основним завданням блоку обробки даних є забезпечення коректності даних, які використовуються додатком у будь-який момент часу.

У цьому блоці забезпечується уся бізнес-логіка застосування, що забезпечує обробку даних.

У програмному додатку обробляються два види даних :

– внутрішні дані;

– дані, що виводяться.

Частина внутрішніх даних приймається з сховища даних і використовується в роботі алгоритмів для контролю і відстеження інформації про нові об'єкти. Інша частина цих даних створюється в результаті роботи застосування.

Дані, що виводяться призначені для передачі на форму і відображення їх на ній. Такі дані можуть представлятися у вигляді списків мінтерм, мінімізованої функції або у вигляді спливаючих підказок. Дані такого виду відповідають поточному стану рішення задачі. Вони відображують результати на запити користувача в застосуванні, якщо вимагається, мають бути збережені в застосуванні після запиту користувача.

Також в модулі обробки даних забезпечується відповідність між внутрішніми даними, що виводяться, в результаті запиту користувача.

У цьому модулі реалізуються контроль коректності даних, що вводяться, і алгоритми, оброблювальні запити користувача.

Основним завданням модуля представлення даних є завдання коректного відображення і своєчасного оновлення даних на формі. Це дозволяє відображувати актуальний стан додатку у нинішній момент часу.

2.4.3. Опис фізичної структури програми

Відповідно функціональним можливостям та математичного опису алгоритму методу мінімізації, створюємо інтерфейс програмного додатку у інтегрованому середовищі розробки MS Visual Studio 2019 мовою C# за допомогою інтерфейсу програмування WindowsForms.

UI складається візуальних активних компонентів, явно розташованих на формі (QMC) та компонентів динамічно створених та розташованих на формі під час роботи програми (рис. 2.14). До активних, явно розташованих компонентів можна віднести:

– ComboBox (lstNrVar) – вибір розміру змінної (мінтерми) БФ;

– Label (lblFunction) – вивід результату мінімізації БФ;

– Button (btbProcess) – запуск процесу мінімізації БФ;

– StatusStrip (statusStrip1) – обрані мінтерми у десятковій системі числення, відсортовані за зростанням.

До динамічних компонентів відносяться:

– Label (Label[] pic) – масив компонентів вибору мінтерм на карті Карно;

– Label ([] picrow) – масив заголовку рядків карти Карно;

– Label ([] piccol) – масив заголовку стовбців карти Карно;

– CheckedListBox ([] lstTerms) – масив відображення обраних мінтерм та етапів обчислення за методом Куайна-Мак-Класки у двійковій системи числення.

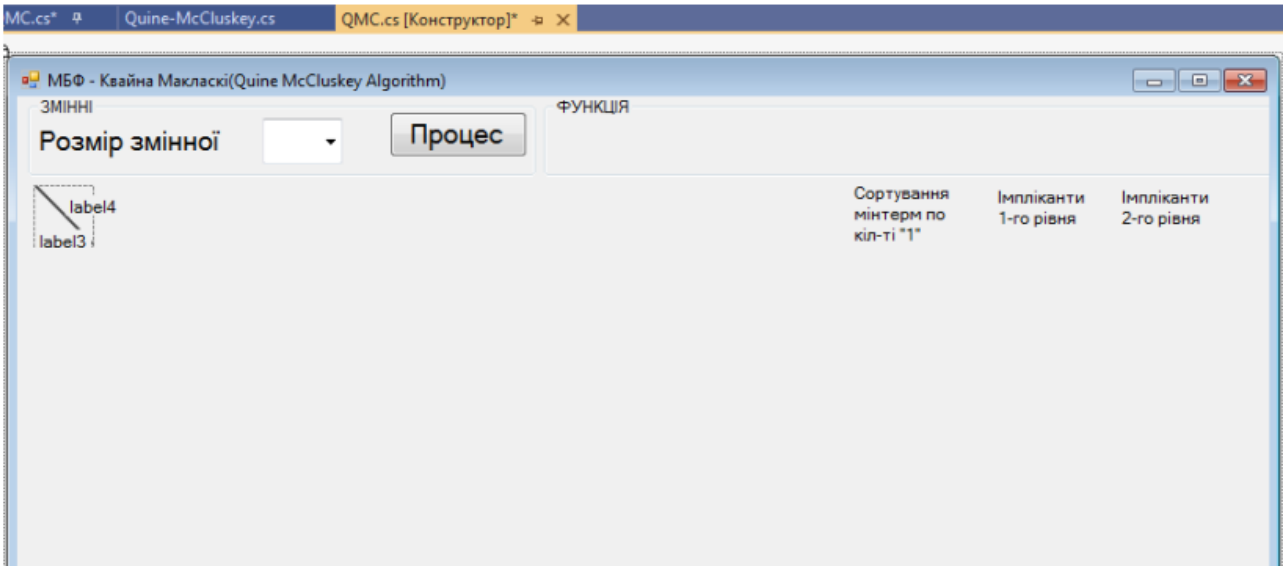

Рис. 2.14. Головна форма програми

Код програми фактично вирішує задачу мінімізації. Програмний код додатку поділяється на дві частини:

– безпосередня обробка подій компонентів форми, файл – QMC.cs;

– відокремлений модуль розташований у просторі імен QM – файлу Quine-McCluskey.cs. Цей клас можна і треба використовувати повторно.

Файл QMC.cs – містить визначення конструктору головної форми програми, подій компонентів форми та додаткових методів обробки:

– private void Init() – ініціалізація динамічних компонентів керування форми;

– private void lstNrVar_SelectedIndexChanged(object sender, EventArgs e) – вибір розміру змінної (мінтерми) БФ;

– private void FunctionDesigner(int NrVar) – ініціалізація динамічних візуальних компонентів форми;

– public void casuta_noua(int i, int j) – заповнення комірки карти Карно;

– public void hedercol(int k) – заповнення заголовків стовбців карти Карно;

– public void hederrow(int k) – заповнення заголовків рядків карти Карно;

– private void Pic_MouseMove(object sender, EventArgs e) - зміна меж

комірки карти Карно під час руху мишею по формі над коміркою;

– private void Pic_Clicked(object sender, EventArgs e) – включення/виключення комірки карти Карно до/з списку мінтерм БФ;

– private void ShowTerms() – відображення обраних мінтерм БФ у рядку стана в десятковій системі числення та у полі ChekedListBox – двійковій системі числення;

– private void btbProcess_Click(object sender, EventArgs e) – запуск алгоритму мінімізації БФ методом Куайна-Мак-Класкі.

Простір імен QM містить клас:

– qmTools – містить методи обробки інтерфейсу програми при формуванні результату роботи алгоритму:

1. public static bool eInAcoperire(string el, string[] colectie) – визначення включення мінтерм на поверхні.
2. public static bool eInTermen(string alaMic, string alaMare) – визначення включення мінтентерми до первинної БФ.
3. public static string binaryToString(String term) – перетворення мінтерми у двійкову систему числення у літеральну.
4. public static string litera(char c, int poz) – перетворення літеральної частини мінтерми у двійкове значення.
5. public static String binary(int i, int NrVar, int NrCols, int NrLins) – перетворення мінтермів до її двійкового представлення за позиціям комірок карт Карно.
6. public static String binstr(int i, int NrVar) – перетворення десяткового числа у двійкове в залежності від розміру мінтерми.
7. public static void scoateTermen(string ss, string[] eta) – видалення поверхні з масиву поверхонь.
8. public static void adaugaTermen(string ss, string[] eta) – додання поверхні до масиву поверхонь.

– qmAlg – містить методи виконання алгоритму Куайна-Мак-Класкі:

1. public void ProcessInit(int qr) – застосувати обробку мінтерми у

первиній формі.

2. private int reshape() – застосувати обробку визначеної інмпілканти.

3. public void eliminaDuplicati(string[] array) – знаходження дублікатів імплікант на поверхнях.

4. public void FindFunction(string[][] etap,int qr) – отримання функції мінімізації в кінцевому вигляді – ДДНФ.

2.4.4. Реалізація класу Quine_McCluskey

Відповідно до теорії метод Мак-Класки складається з двох основних етапів:

1. Знаходження всіх простих терм ЛФ, використовуючи правило (закони) склеювання:

– (A & B) ∨ (A & !B) ≡ A;

– (A ∨ B) & (A ∨ !B) ≡ A;

де & - операція логічного «І»;

∨ - операція логічного «АБО»;

! - операція логічного «Ні».

2. Мінімізація кількості простих терм отриманої безлічі, як завдання знаходження оптимального покриття безлічі підмножини різної довжини.

Клас Quine_McCluskey є власне реалізацією цього алгоритму, якому допомагають інші класи і інтерфейси: Dnf, TreeNodeBase, TreeNodeMiddle, TreeNodeEnd, TreeFuncTerm. Для запуску оптимізації потрібно викликати один з перевантажених методів Start.

У пункті 1 відбувається багаторазовий попарний перебір терм з метою виявлення можливості їх склеювання. Два терма склеюються, якщо вони відрізняються один від одного значеннями тільки в одній з позицій: у одного варто '0', а в іншого - '1'. Склеюються вихідні (батьківські) терми виключаються

з подальшої роботи, а замість їх двох далі на наступній ітерації п. 1 алгоритму розглядається нова дочірня терма, у якій на позиції різняться змінних батьківських терм стоїть знак '*'. Таким чином, двійковий алфавіт вхідних вихідних терм всередині алгоритму розширюється до потрійного: '0', '1', '*'. Слід зазначити, що в даній реалізації замість масивів символів використовуються байтові масиви, в яких символів '0', '1' і '*' відповідають значення байт 0, 1 і 2.

Пункт 1 в класі Quine_McCluskey реалізований в процедурі LogicFuncMinimize двома принципово різними способами. Перший спосіб, заснований на 64 бітних цілочисельних хеш-кодах і пошуку в .NET словнику Dictionary <UInt64, ...>, працює при кількості вхідних змінних булевою функції менше або дорівнює 40, а другий, заснований на пошуку в троїчному дереві, включається при більшому їх кількості. Таке роздвоєння обумовлено тим, що перший спосіб працює трохи швидше, ніж другий, тому його використання приоритетнее, але він працює коректно тільки при кількості вхідних змінних до 40, а при більшому їх кількості виходить за 64 біт розрядну сітку цілочисельного хеш-коду, використовуваного для роботи алгоритму. Дійсно, т. К. В термах всередині алгоритму застосовується тризначна логіка, то при кількості вхідних змінних рівному 41 максимальне значення хеш-коду 341 вже перевищує максимальне значення (264-1), яке можна записати в 64 бітну змінну.

Другий спосіб роботи п. 1, який працює при кількості змінних у вхідних термах більше 40, заснований на пошуковому троїчному дереві терм TreeFuncTerm. Проміжні вузли дерева реалізовані за допомогою класу TreeNodeMiddle. Кожен з них може посилатися максимум на три наступних вузла в залежності від того чи були в дерево додані відповідні терми. Всі кінцеві вузли є екземплярами класу TreeNodeEnd, глибина до яких від кореня у них всіх однакова і дорівнює кількості вхідних змінних. Кожен кінцевий вузол також має посилання на інший вузол TreeNodeEnd, який був доданий до нього, реалізуючи тим самим односпрямований пов'язаний список. Такого роду

список використовується для швидкого перебору всіх кінцевий вузлів пошукового дерева в процесі їх склеювання.

Якщо в п. 1 для терма чи не знаходить іншого відрізненого від нього тільки в одній позиції терма, тобто терм ні з ким не склеюється, то він вважається одним з результатів роботи п. 1 алгоритму, виключається з подальшої роботи в ньому і надходить на вхід пункту 2 роботи алгоритму, реалізований в процедурі ReduceRedundancyTerms.

Відкидання надлишкових терм в ReduceRedundancyTerms здійснюється за допомогою алгоритму наближеного рішення задачі про покриття безлічі підмножин змінної довжини. Покриття, близьке до найкоротшому дає алгоритм перетворення таблиці покриттів (ТП), заснований на методі "мінімальний стовпець-максимальна рядок".

Логіка його роботи полягає в наступному:

Вихідна таблиця вважається поточної перетворюється ТП, безліч рядків покриттів - порожньо;

У поточній таблиці виділяється стовпець з найменшим числом одиниць. Серед рядків, що містять одиниці в цьому стовпці, виділяється одна з найбільшим числом одиниць. Цей рядок включається в покриття, поточна таблиця скорочується викреслюванням усіх стовпців, в яких обрана рядок має одиницю;

Якщо в таблиці не викреслені стовпці, то виконується п. 2, інакше - покриття побудовано.

Примітка: При підрахунку числа одиниць в рядку враховуються одиниці в невичеркнутих стовбцях.

Для перевірки роботи алгоритму використовуються дві тестові функції. Перша процедура дозволяє для набору вхідних змінних кількістю N сформувати сукупність терм кількістю 2^N, обчислити випадкове значення ЛФ для кожного терма, провести мінімізацію на основі тих термінів, для яких відповідне їм значення функції дорівнювало TRUE, і вивантажити їх у разі потреби в файл. Після цього проводиться перевірка правильності роботи

мінімізованої форми функції шляхом обчислення її значення для кожного терма і порівняння з вихідним.

Приклад рішення алгоритму:

```
public static void TestQuineMcCluskeyRandom(int iVariableAmount, string sFileName = "")
{
 int iTotalCombines = 1 << iVariableAmount;
 ICollection<KeyValuePair<byte[], bool>> FuncResult = new List<KeyValuePair<byte[], bool>>(iTotalCombines);
 if (!String.IsNullOrEmpty(sFileName)) File.Delete(sFileName);
 Random rnd = new Random();
 //int iLogicFuncMask = 0;
 //while (iLogicFuncMask == 0) iLogicFuncMask = rnd.Next(iTotalCombines);
 for (int iCounter = 0; iCounter < iTotalCombines; iCounter++)
 {
  int iCurValue = iCounter;
  byte[] pTerm = new byte[iVariableAmount];
  for (int i = 0; i < iVariableAmount; i++)
  {
   pTerm[i] = (byte)(iCurValue % 2);
   iCurValue /= 2;
  }
  bool bFuncValue = (rnd.Next(2) != 0);
  //bool bFuncValue = (iCounter & iLogicFuncMask) > 0;
  if (bFuncValue && !String.IsNullOrEmpty(sFileName))
  {
   File.AppendAllText(sFileName, pTerm.ToString() + Environment.NewLine);
  }
```

```
            FuncResult.Add(new KeyValuePair<byte[], bool>(pTerm, bFuncValue));
        }
        //Позитивні терми
        IEnumerable<byte[]> pTrueTerms = FuncResult.Where(p =>
p.Value).Select(p => p.Key);
        //Запуск в работу
        DateTime DtStart = DateTime.Now;
        Console.WriteLine("Старт - " + DtStart.ToLongTimeString());
        Quine_McCluskey Logic = new Quine_McCluskey();
        Logic.Start(pTrueTerms);
        DateTime DtEnd = DateTime.Now;
        TimeSpan Elapsed = DtEnd - DtStart;
        Console.WriteLine("Завершение - " + DtEnd.ToLongTimeString());
        Console.WriteLine("Длительность - " +
String.Format("{0:00}:{1:00}:{2:00}",
          Elapsed.Hours, Elapsed.Minutes, Elapsed.Seconds));
        //Сравнение результатов
        int iErrorsCounter = 0;
        foreach (KeyValuePair<byte[], bool> kvp in FuncResult)
        {
          if (Logic.Result.Calculate(kvp.Key) != kvp.Value) iErrorsCounter++;
        }
        Console.WriteLine("Кількість вхідних змінних = " + iVariableAmount);
        Console.WriteLine("Кількість вхідних термів = " + iTotalCombines);
        Console.WriteLine("Кількість диз’юнктивних термів = " +
Logic.Result.Terms.Count);
        Console.WriteLine("Кількість помилок = " + iErrorsCounter);
        Console.ReadLine();
      }
```

2.5. Обґрунтування та організація вхідних та вихідних даних програми

Вхідними даними програми є:

– вибір кількісті змінних булевої функції – від 2 до 6;

– очікуваний результат виконання функції у вигляді ряду сум імплікант.

Вихідними даними програми є:

– карти Карно;

– представлення рішення Куайна-Мак-Класкі;

– дійсна форма функції;

– скорочена результуюча мінімізована форма функції.

У програмному додатку обробляються два види даних :

– внутрішні дані;

– дані, що виводяться.

Створення окремого модуля для отримання даних дозволяє позбавитися залежності логіки додатка від джерела даних. Це дозволяє використовувати в ролі сховища даних будь-які інші джерела, такі як файли. Крім того, змінивши сховище даних, змін зажадає тільки сам блок даних.

Частина внутрішніх даних приймається з сховища даних і використовується в роботі алгоритмів для контролю і відстеження інформації про нові об'єкти. Інша частина цих даних створюється в результаті роботи додатку.

Дані, що виводяться призначені для передачі на форму і відображення їх на ній. Такі дані можуть представлятися у вигляді списків мінтерм, мінімізованої функції або у вигляді спливаючих підказок. Дані такого виду відповідають поточному стану рішення задачі. Вони відображують результати на запити користувача в застосуванні, якщо вимагається, мають бути збережені в застосуванні після запиту користувача.

2.6. Опис роботи розробленої системи

2.6.1. Використані технічні засоби

Для розробки даного додатку було використано ПК з наступними характеристиками:

1. Процесор Intel® Core™2 Duo CRU @ 3.06GHz
2. Оперативна пам’ять: 512 Мб.
3. Операційна система: Windows 7.
4. Роздільна здатність монітору: 1280x800.
5. Наявність засобів вводу (клавіатура, мишка).
6. Вільне місце на жорсткому диску: 10 Мб.

2.6.2. Використані програмні засоби

Програмний додаток написаний за допомогою методів візуального програмування на мові C# у інтегрованому середовищі розробки MS Visual Studio 2019 і скомпільований з розширенням exe. Для стабільної роботи програми слід використовувати Windows 7 та Windows 8.1.

2.6.3. Виклик та завантаження програми

Для установки програмного додатку потрібно папку Bool_Function скопіювати на потрібний комп’ютер і відкрити файл Bool_Function.exe

Вхід в систему виконується після завантаження додатку. При цьому відкривається головне вікно системи.

2.6.4. Опис інтерфейсу користувача

Під час розробки програми велика увага була приділена графічному інтерфейсу UI, з ним користувачеві повинно бути легко працювати за допомогою миші.

Інтерфейс розподілений на 4 частини:

– верхня частина інтерфейсу дозволяє користувачу обрати розмір змінної логічної функції (кількість бітів), від 2 до 6;

– середня частина – найбільша. Ліва частина призначена для графічного представлення вхідних даних у вигляді карт Карно, а права – для складання таблиці істинності з мінтерм у двійковому представлені та етапів визначення імплікант;

– нижня частина вікна програми містить рядок стану, який відображає список обраних мінтерм у десятковій системі числення.

Під час запуску програми, UI має вигляд за визначеним розміром змінної булевої функції (БФ), що дорівнює 2 (рис. 2.15).

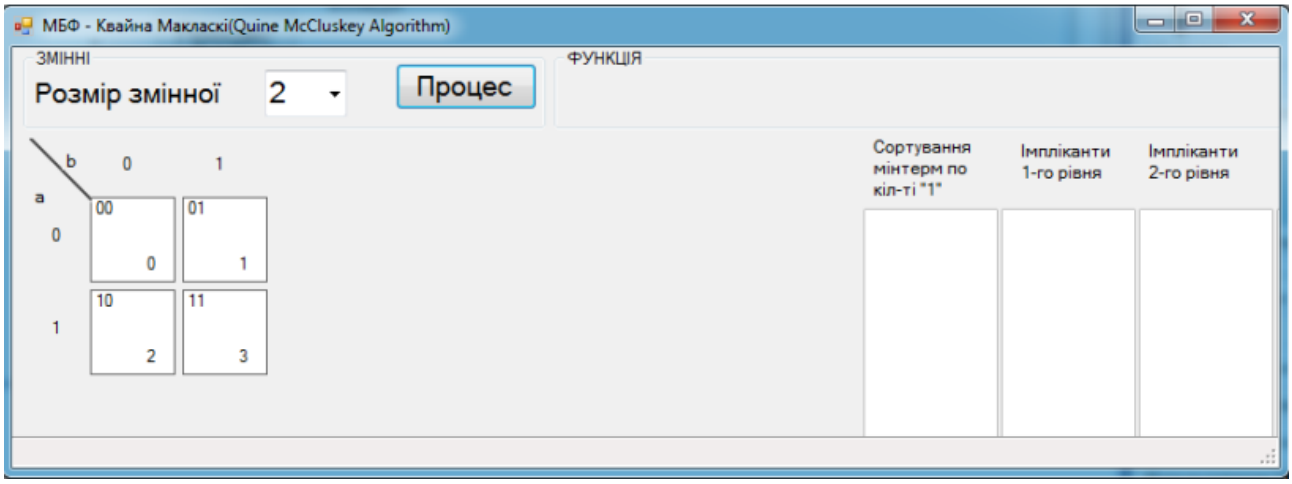

Рис. 2.15. Головна форма програмного додатку

Для формування первинного виразу БФ, користувачеві насамперед необхідно обрати зі списку розмір змінної БФ. На відповідь вибору користувача

буде сформована карта Карно, де користувач може обрати необхідні мінтерми для первинного виразу БФ.

Під час вибору мінтерм на карті Карно користувач може відмінити свій вибір відносно окремої комірки повторним щигликом миші по комірці. Список обраних мінтерм відображається у двох місцях вікна програми:

– рядок стану – у десятковій системі числення;

– список «Сортування мінтерм по кількості «1»» – у двійковій системі числення.

По завершенню формування виразу БФ (таблиці істинності) користувач може натиснути на кнопку «Процес».

Окрім згорнутої функції (у верхній частині вікна) яку можна копіювати, програма надає користувачеві серію списків: «Імпліканти 1-го рівня» та «Імпліканти 2-го рівня», які точно показують, як виконувався кожен крок алгоритму, і статус після кожного кроку (мінтерми з прапорцем або неперевірений стан).

Даний програмний додаток був протестований на відсутність помилок при різних наборах вхідних даних: розмірах мінтерм та їх значень, на рис. 2.16-2.23 представлені результати мінімізації БФ різних розмірів значення змінної.

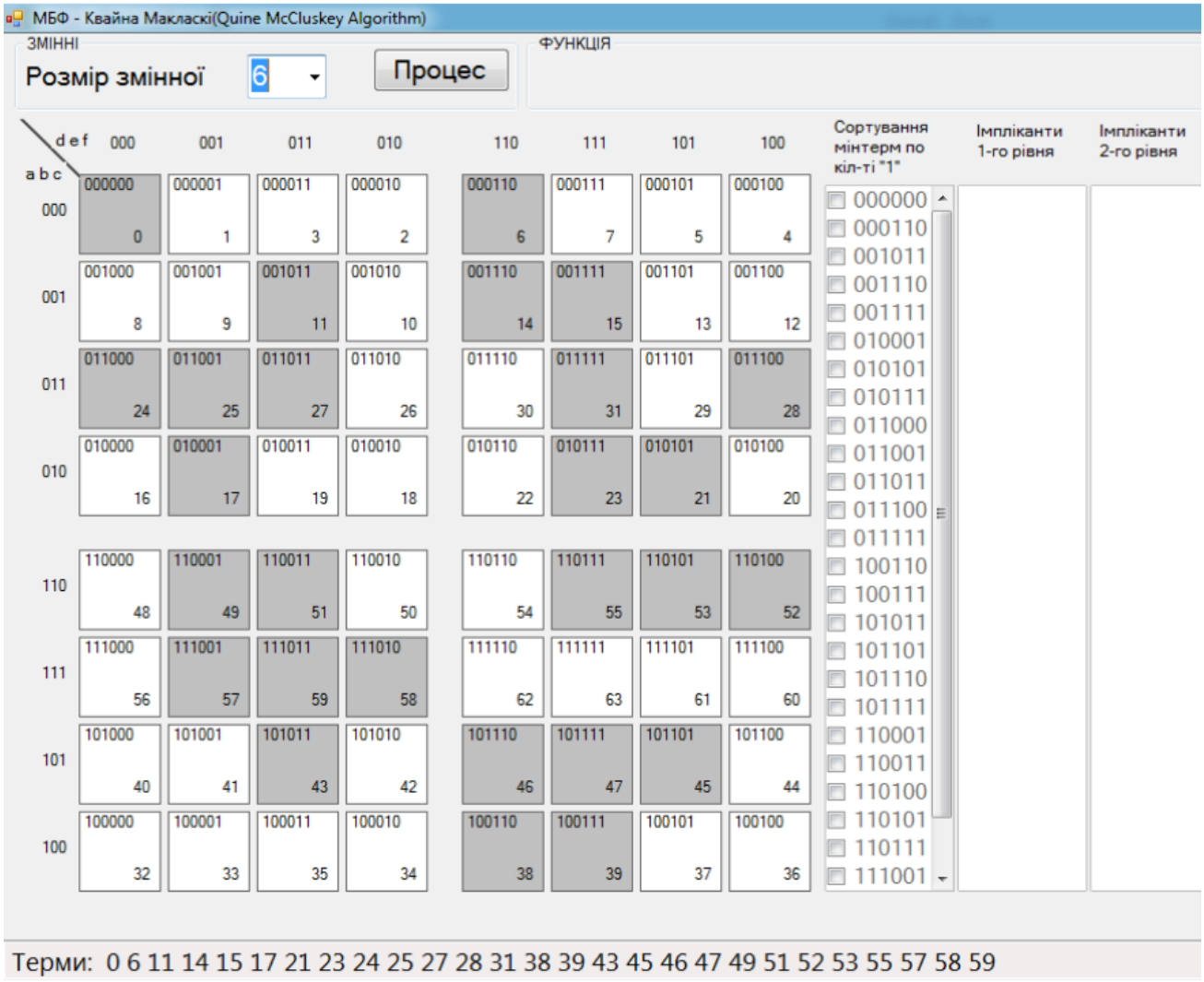

Рис. 2.16. Результат роботи програми з розміром значення змінної – 6: вибір змінної БФ;

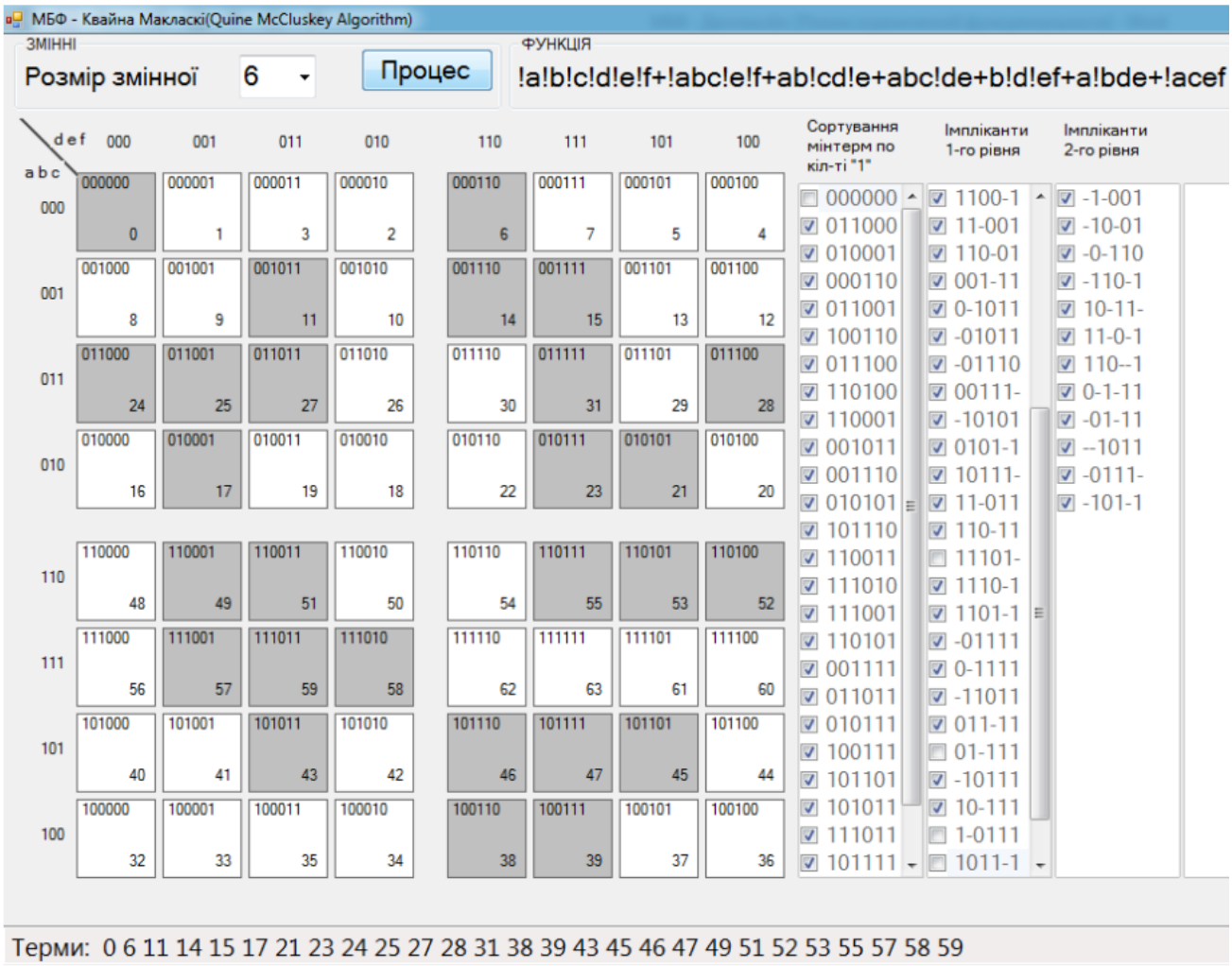

Рис. 2.17. Результат роботи програми з розміром значення змінної – 6: після мінімізації БФ.

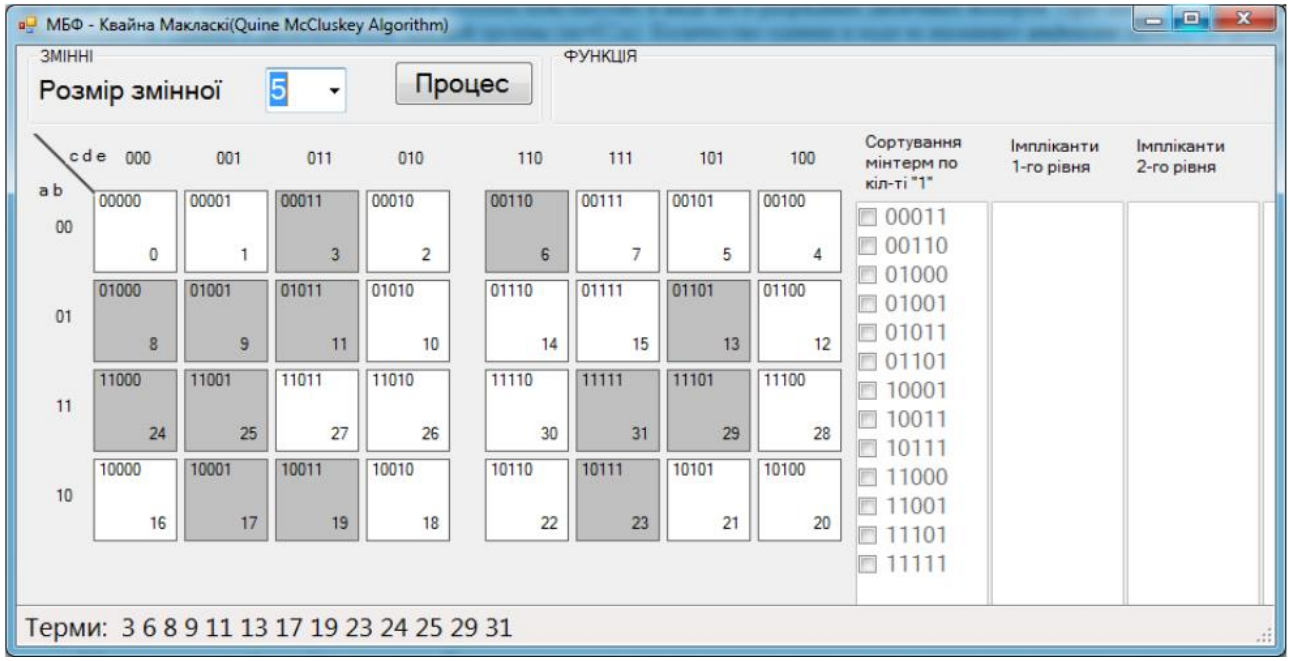

Рис. 2.18. Результат роботи програми з розміром значення змінної – 5: вибір змінної БФ

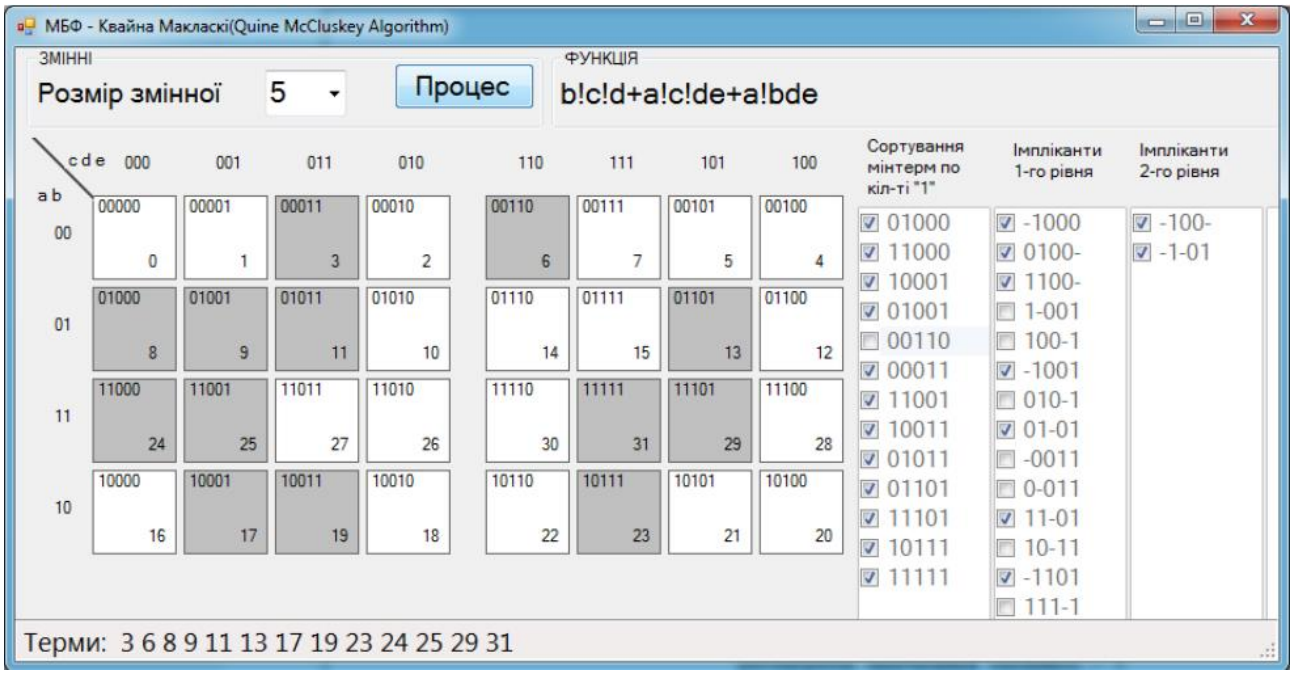

Рис. 2.19. Результат роботи програми з розміром значення змінної – 5: після мінімізації БФ.

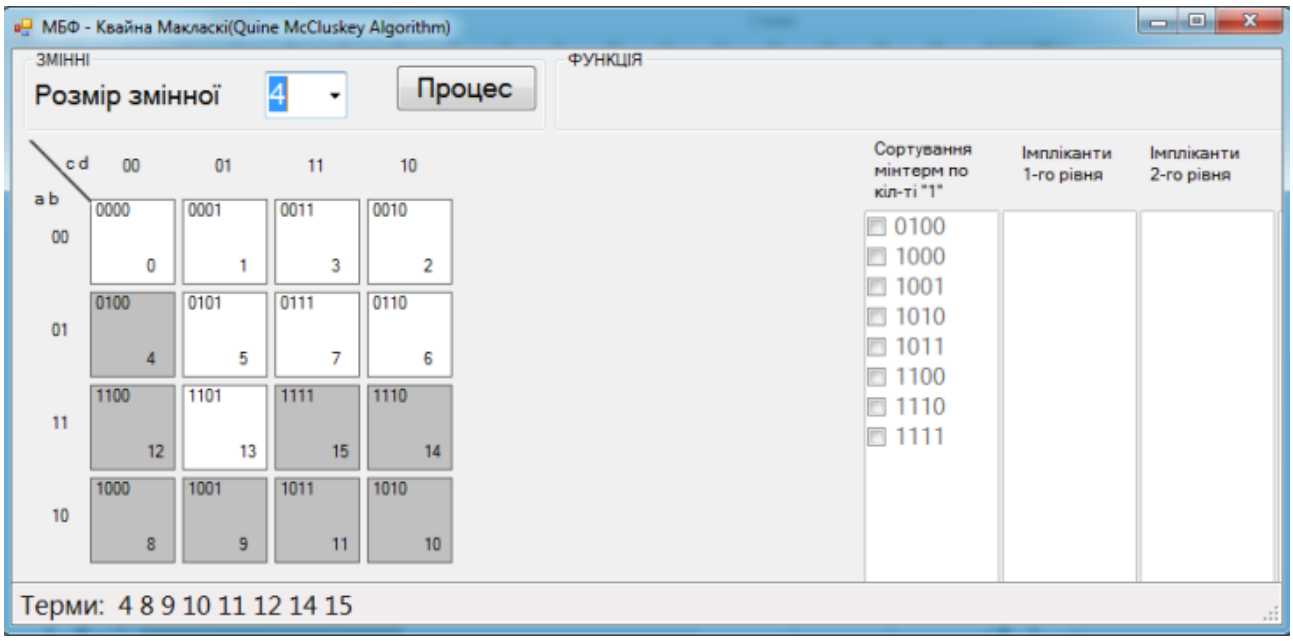

Рис. 2.20. Результат роботи програми з розміром значення змінної – 3: вибір змінної БФ

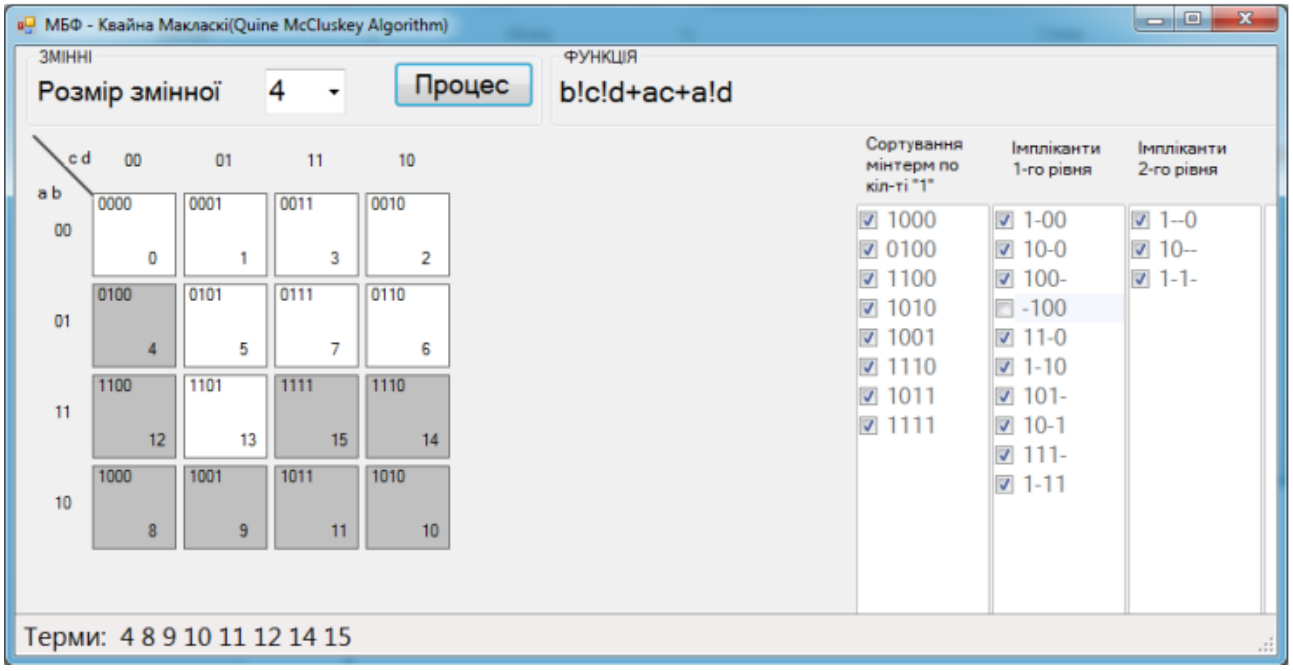

Рис. 2.21. Результат роботи програми з розміром значення змінної – 3: після мінімізації БФ.

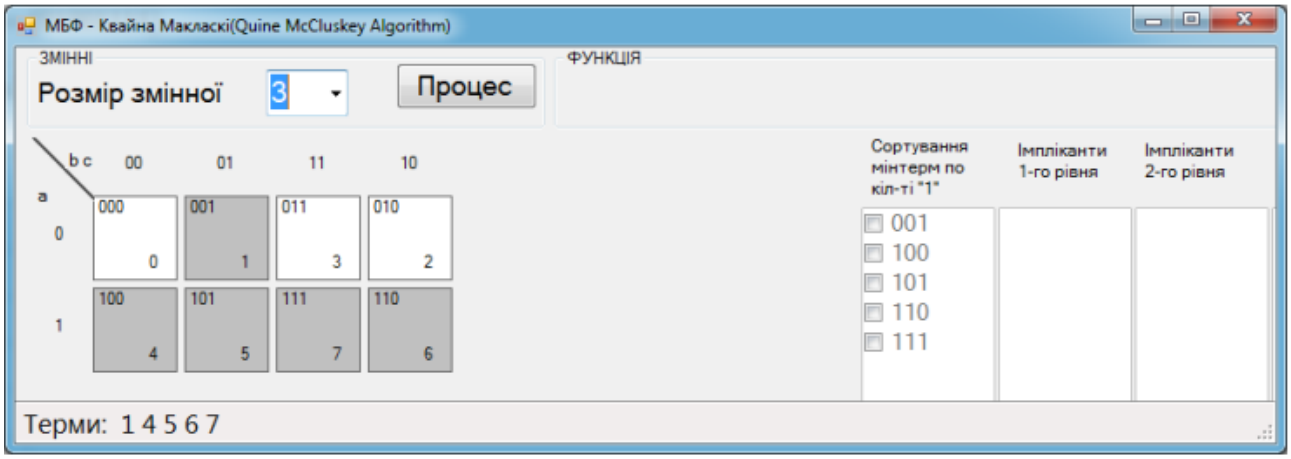

Рис. 2.22. Вибір мінтерм БФ з розміром значення змінної – 2

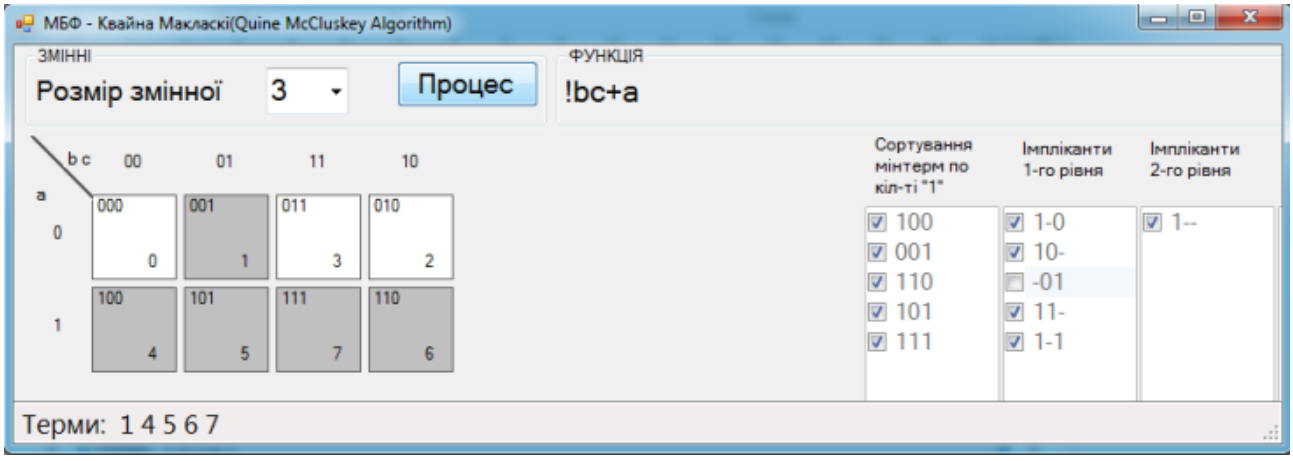

Рис. 2.23. Результат роботи програми з розміром значення змінної – 2

Інструкція користувача програмним додатком додається у файлі MBF_help.pdf.

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Вхідні дані:

- передбачуване число операторів – 800;
- коефіцієнт складності програми – 1,5;
- коефіцієнт корекції програми в ході її розробки – 0,05;
- годинна заробітна плата програміста, грн / год – 40.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_u + t_a + t_n + t_{отл} + t_д, \text{ людино-годин,} \quad (3.1)$$

де:

to - витрати праці на підготовку й опис поставленої задачі (приймається 50);

ти - витрати праці на дослідження алгоритму рішення задачі;

та - витрати праці на розробку блок-схеми алгоритму;

тп - витрати праці на програмування по готовій блок-схемі;

тотл - витрати праці на налагодження програми на ЕОМ;

тд - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \times C \times (1+ p) \text{ , людино-годин,} \quad (3.2)$$

де q - передбачуване число операторів;

С - коефіцієнт складності програми;

р - коефіцієнт кореляції програми в ході її розробки.

$$Q = 800 \cdot 1{,}5 \cdot (1 + 0{,}05) = 1260 \text{ людино-годин.} \quad (3.3)$$

Витрати праці на вивчення опису задачі ti визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{QB}{(75...85)K} \text{ , людино-годин,} \quad (3.3)$$

де В - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі; В=1.2 ... 1.5;

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності.

Витрати праці на розробку алгоритму рішення задачі:

$$t_u = \frac{1260 \cdot 1{,}2}{80 \cdot 0{,}8} = 23 \text{, людино-годин.} \quad (3.4)$$

Витрати на складання програми по готовій блок-схемі:

$$t\alpha = \frac{Q}{(20...25)K} \quad \text{людино-годин.} \quad (3.5)$$

$$t_a = \frac{1260}{20 \cdot 0{,}8} = 78 \text{ людино-годин.} \quad (3.6)$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25)K} \quad \text{людино-годин.} \tag{3.7}$$

$$t_n = \frac{1260}{25 \cdot 0{,}8} = 63 \text{ людино-годин.} \tag{3.8}$$

Витрати праці на налагодження програми на ЕОМ:

$$t_{отл} = \frac{Q}{(4...5)K} \quad \text{людино-годин.} \tag{3.9}$$

$$t_{отл} = \frac{1260}{4 \cdot 0{,}8} = 393 \text{ людино-годин.} \tag{3.10}$$

за умови комплексного налагодження завдання:

$$t^{к}_{отл} = 1{,}2 \cdot t_{отл}; \tag{3.11}$$

$$t_{отл} = 393 * 1{,}2 = 472{,}5 \tag{3.12}$$

Витрати праці на підготовку документації:

$$t_д = t_{др} + t_{до}, \text{ людино-годин,} \tag{3.13}$$

де tдр - трудомісткість підготовки матеріалів і рукопису.

$$t_{др} = \frac{Q}{(15...20)K}, \text{ людино-годин.} \tag{3.14}$$

$$t_{др} = \frac{1260}{15 \cdot 0{,}8} = 105 \text{ людино-годин,} \tag{3.15}$$

tдо - трудомісткість редагування, печатки й оформлення документації

$$t_{до} = 0{,}75 \cdot t_{др}, \text{ людино-годин.} \quad (3.16)$$

$$tdo = 0{,}75 \cdot 105 = 78 \quad (3.17)$$

$$t_{д} = 105 + 78 = 183 \text{ людино-годин.} \quad (3.18)$$

Отримуємо трудомісткість розробки програмного забезпечення:

$$t = 50 + 23 + 78 + 63 + 472 + 183 = 871 \text{ людино-годин.} \quad (3.19)$$

3.2. Витрати на створення програмного забезпечення

Витрати на створення ПЗ Кпо включають витрати на заробітну плату виконавця програми Зз/п і витрат машинного часу, необхідного на налагодження програми на ЕОМ:

$$К_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн,} \quad (3.20)$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР}, \text{ грн,} \quad (3.20)$$

де:

t - загальна трудомісткість, людино-годин;

Спр - середня годинна заробітна плата програміста, грн/година

$$З_{ЗП} = 871 \cdot 40 = 34865 \text{ грн.} \quad (3.21)$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} \times С_{М}, \text{ грн,} \quad (3.22)$$

де:

тотл - трудомісткість налагодження програми на ЕОМ, год.

Смч - вартість машино-години ЕОМ, 7 грн/год.

Визначені в такий спосіб витрати на створення програмного забезпечення є частиною одноразових капітальних витрат на створення АСУП.

$$З_{МВ} = 472 \times 7 = 3307 \text{ грн.} \quad (3.23)$$

$$Кпо = 34865 + 3307 = 38172 \text{ грн.} \quad (3.24)$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p} \text{ міс.} \quad (3.25)$$

де:

Bk - число виконавців;

Fp - місячний фонд робочого часу (при 40 годинному робочому тижні Fp=176 годин).

$$T = \frac{871}{1 \cdot 176} = 4{,}9 \text{ міс.} \quad (3.26)$$

Визначено трудомісткість розробленої інформаційної системи (871 люд-год), проведений підрахунок вартості роботи по створенню програми (38172 грн.) та розраховано час на його створення (4,9 міс).

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було розроблено та спроектовано навчальний програмний додаток для настільної ОС Windows, який дозволяє користувачеві ознайомитися з методом мінімізації булевих функцій методом Куайна-Мак-Класкі та формуванням карт Карно.

Інтерфейс програми надає користувачеві можливість працювати з даними та отримати результати мінімізації функції, які може скопіювати у буфер обміну для подальшого використання.

Для розробки програмного додатку були використані: мова програмування - C#, інтерфейс програмування – WindowsForms, середовище розробки – MS Vis-ual Studio 2019.

Під час розробки програми велика увага була приділена графічному інтерфейсу користувача, з ним користувачеві повинно бути легко працювати за допомогою миші.

Інтерфейс розподілений на 4 частини:

– верхня частина інтерфейсу дозволяє користувачу обрати розмір змінної логічної функції (кількість бітів), від 2 до 6;

– середня частина – найбільша. Ліва частина призначена для графічного представлення вхідних даних у вигляді карт Карно, а права – для складання таблиці істинності з мінтерм у двійковому представлені та етапів визначення імплікант;

– нижня частина вікна програми містить рядок стану, який відображає список обраних мінтерм у десятковій системі числення.

Програма надає користувачеві серію списків: «Імпліканти 1-го рівня» та «Імпліканти 2-го рівня», які точно показують, як виконувався кожен крок алгоритму, і статус після кожного кроку (мінтерми з прапорцем або неперевірений стан).

Дана програма надає можливість зручного обчислення булевих функцій та їх застосування на заняттях таких предметів як: «Математична логіка», «Дискретна математика» та при конструюванні і спрощенні логічних схем електронних пристроїв комп'ютерів, калькуляторів, телефонних систем і ряду інших пристроїв, а також в таких областях, як розпізнавання образів, теорія кодування і криптографія.

В економічному розділі були проведені обчислення трудомісткості та витрат для розробки програмного забезпечення. Для створення даної інформаційної системи знадобиться 4,9 місяця, трудомісткість розробки ПЗ – 871 людино-годин, а витрати на її створення програмного забезпечення – 38172 грн.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання : (ГОСТ 7.1-2003, IDT) : ДСТУ ГОСТ 7.1:2006. – Чинний з 2007–07–01. – К. : Держспоживстандарт України, 2007. – 47 с. – (Система стандартів з інформації, бібліотечної та видавничої справи) (Національний стандарт України).

2. Бусигін Б.С., Коротенко Г.М., Коротенко Л.М. Прикладна інформатика. Підручник для студентів комп'ютерних спеціальностей. – Дніпропетровськ: Видавництво НГУ, 2004. – 559 с. URL: http://www.programmer.dp.ua/book-ua-k01.php. дата звернення: 15.05.2021.

3. Бусыгин Б.С., Дивизинюк М.М., Коротенко Г.М., Коротенко Л.М. Введение в современную информатику. Учебник. – Севастополь: Издательство СНУЯЭиП, 2005. – 644 с. / URL: http://www.programmer.dp.ua/book-ru-k02.php. дата звернення: 15.01.2018.

4. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення : ДСТУ 3008-95. – Чинний від 1996–01–01. – К. : Держстандарт України, 1996. – 39 с.

5. ДСТУ 2394-94 Інформація та документація. Комплектування фонду, бібліографічний опис, аналіз документів. Терміни та визначення. – Чинний від 01.01.1995. - Київ : Держстандарт України, 1994. – 88 с.

6. Карти Карно - Правила спрощення. URL: https://uk.wikipedia.org/wiki/Карта_Карно. дата звернення: 15.05.2021.

7. Кочубей О.О., Сопільник О.В. Прикладна теорія цифрових автоматів. Логічні основи, - Видавництво Дніпропетровського університету, 2009. - 264 с.

8. Методичні вказівки з виконання економічного розділу в дипломних проектах студентів спеціальності “Комп’ютерні системи ” / Укладачі О.Г. Вагонова, Нікітіна О.Б. Н.Н. Романюк – Дніпропетровськ: Національний гірничий університет. – 2013. – 23с.

9. Методичні рекомендації до виконання кваліфікаційних робіт бакалаврів напряму підготовки 122 «Комп'ютерні науки» галузі знань 12 Інформаційні технології/, Л.М. Коротенко , О.С. Шевцова; Нац. гірн. ун-т. – Д : ДВНЗ НГУ, 2019. – 65 с.

10. Методичні рекомендації щодо написання, оформлення та представлення учнівських науково-дослідницьких робіт учнів – членів Малої академії наук України / Г.Г. Півняк, Л.М. Коротенко, І.М. Удовик, Є.М. Головня – Д.: ДВНЗ «Національний гірничий університет», 2017. – 24 с.

11. Нейгел К., Ивьен Б., Глинн Дж., Уотсон К. C# 4.0 и платформа .NET 4 для профессионалов, -М.: ООО "И.Д. Вильямс", 2011. – 1440 с.

12. Скорочення слів в українській мові у бібліографічному описі. Загальні правила та вимоги : ДСТУ 3582-97. – Чинний від 1998–07–01. – К. : Держстандарт України, 1998. – 24 с. – (Державний стандарт України).

13. Офіційний сайт середовища розробки Visual Studio Code URL: https://code.visualstudio.com/docs дата звернення: 15.05.2021.

14. Троелсен Э., Дж., Филипп. Язык программирования C# 7 и платформы .NET и .NET Core, 8-е изд.: – СПб:"Диалектика", 2018 – 1328 с.

15. Прайс Марк Дж. C# 7 и .NET Core. Кросс-платформенная разработка для профессионалов. 3-е изд. – СПб.: Питер, 2018. – 640 с.

16. Статті про операційну систему Windows 7 URL: microsoft.com. дата звернення: 15.05.2021.

17. Статті про операційну систему Windows 8.1 URL: microsoft.com. дата звернення: 15.05.2021.

18. Internet – сайт framework URL: https://docs.microsoft.com/ru-ru/dotnet/framework/ get-started/overview. дата звернення: 5.03.2021.

19. Internet – сайт professorweb URL: https://professorweb.ru/my/csharp/charp_theory/ level1/1_4.php. дата звернення: 15.05.2021.

20. Internet – сайт visual-studio-2019-community URL: https://info-comp.ru/programmirovanie/739-install-visual-studio-2019-community.html. дата звернення: 15.03.2021.

21. Internet – сайт computers-notebooks URL:: https://rozetka.com.ua/computers-notebooks/c80253/. дата звернення: 5.04.2021.

22. Intellectual Technologies on Transport. Issue № 2, 2018, В.А.Табанина «Выбор оптимального метода минимизации при разработке программы поиска минимальной дизъюнктивной нормальной формы». дата звернення: 1.04.2021.

ДОДАТОК А

КОД ПРОГРАМИ

```
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace Kval
{
   static class Program
   {
      /// <summary>
      /// Главная точка входа для приложения.
      /// </summary>
      [STAThread]
      static void Main()
      {
         Application.EnableVisualStyles();
         Application.SetCompatibleTextRenderingDefault(false);
         Application.Run(new Form_start());
      }   }}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace Kval_
{
   public partial class Elektroschemes : Form
   {
      public Elektroschemes()
      {
         InitializeComponent();
      }
      Graphics gr;
      bool a=false, b=false,res=false;
      double x, y,lengtha=20;
       void   draw_osn(float x,float y,double col)
      {
         float width = pictureBox1.Width / (float)col, height = pictureBox1.Height / (float)col;
         gr = pictureBox1.CreateGraphics();
         float fl_lengtha = (float)lengtha;
         float fl_x = (float)x; float fl_y = (float)y;
         gr.DrawLine(Pens.Black, fl_x + fl_lengtha, fl_y + (height / 3), fl_x + width / 4, fl_y +
(height / 3));
```

```
            gr.DrawLine(Pens.Black, fl_x + fl_lengtha, fl_y + ((2 * height) / 3), fl_x + width / 4, fl_y +
((2 * height) / 3));
            gr.FillRectangle(Brushes.White, fl_x, fl_y + (height / 3) - fl_lengtha, 2 * fl_lengtha, 2 *
fl_lengtha);
            gr.FillRectangle(Brushes.White, fl_x, fl_y + ((2 * height) / 3) - fl_lengtha, 2 * fl_lengtha, 2
* fl_lengtha);
            gr.DrawRectangle(Pens.Black, fl_x, fl_y + (height / 3) - fl_lengtha, 2 * fl_lengtha, 2 *
fl_lengtha);
            gr.DrawRectangle(Pens.Black, fl_x, fl_y + ((2 * height) / 3) - fl_lengtha, 2 * fl_lengtha, 2 *
fl_lengtha);
            gr.DrawRectangle(Pens.Black, fl_x + (width / 4), fl_y + fl_lengtha, width / 2, height - 2 *
fl_lengtha);
            gr.DrawLine(Pens.Black, fl_x + (width * 3) / 4, fl_y + height / 2, fl_x + width - 30, fl_y +
height / 2);
            gr.FillRectangle(Brushes.White, fl_x + width - 60, fl_y - 30 + height / 2, 60, 60);
            gr.DrawRectangle(Pens.Black, fl_x + width - 60, fl_y - 30 + height / 2, 60, 60);
        }
        void draw_not(float x, float y, double col)
        {
            float width = pictureBox1.Width / (float)col, height = pictureBox1.Height / (float)col;
            gr = pictureBox1.CreateGraphics();
            float fl_lengtha = (float)lengtha;
            float fl_x = (float)x; float fl_y = (float)y;
 gr.FillEllipse(Brushes.White, fl_x + width / 4 - (fl_lengtha / 2), fl_y + (height / 3) - (fl_lengtha / 2),
fl_lengtha, fl_lengtha);
gr.DrawEllipse(Pens.Black, fl_x + width / 4 - (fl_lengtha / 2), fl_y + (height / 3) - (fl_lengtha / 2),
fl_lengtha, fl_lengtha);
  gr.FillEllipse(Brushes.White, fl_x + width / 4 - (fl_lengtha / 2), fl_y + (2 * height / 3) - (fl_lengtha
/ 2), fl_lengtha, fl_lengtha);
  gr.DrawEllipse(Pens.Black, fl_x + width / 4 - (fl_lengtha / 2), fl_y + (2 * height / 3) - (fl_lengtha /
2), fl_lengtha, fl_lengtha);
        }
        void draw_conun(double x,double y, double col)
        {
            float width = pictureBox1.Width / (float)col, height = pictureBox1.Height / (float)col;
            gr = pictureBox1.CreateGraphics();
            float fl_lengtha = (float)lengtha;
            float fl_x = (float)x; float fl_y = (float)y;
            draw_osn(fl_x, fl_y, col);
            gr.DrawString("&", new Font("Arial", 50), Brushes.Black, fl_x - 30 + width / 2, fl_y - 50 +
height / 2);
        }
        void draw_not_conun(double x, double y, double col)
        {
            float width = pictureBox1.Width / (float)col, height = pictureBox1.Height / (float)col;
            gr = pictureBox1.CreateGraphics();
            float fl_lengtha = (float)lengtha;
            float fl_x = (float)x; float fl_y = (float)y;
            draw_osn(fl_x, fl_y, col);
            draw_not(fl_x, fl_y, col);
            gr.DrawString("&", new Font("Arial", 50), Brushes.Black, fl_x - 30 + width / 2, fl_y - 50 +
height / 2);
```

```
        }
        void draw_not_disun(double x, double y, double col)
        {
            float width = pictureBox1.Width / (float)col, height = pictureBox1.Height / (float)col;
            gr = pictureBox1.CreateGraphics();
            float fl_lengtha = (float)lengtha;
            float fl_x = (float)x; float fl_y = (float)y;
            draw_osn(fl_x, fl_y, col);
            draw_not(fl_x, fl_y, col);
            gr.DrawString("||", new Font("Arial", 50), Brushes.Black, fl_x - 30 + width / 2, fl_y - 50 +
height / 2);
        }
        void draw_disun(double x, double y,double col)
        {
            float width = pictureBox1.Width / (float)col, height = pictureBox1.Height / (float)col;
            gr = pictureBox1.CreateGraphics();
            float fl_lengtha = (float)lengtha;
            float fl_x = (float)x; float fl_y = (float)y;
            draw_osn(fl_x,fl_y,col);
            gr.DrawString("A", new Font("Arial", 16), Brushes.Black, fl_x, fl_y + (height / 3) -
fl_lengtha - 20);
            gr.DrawString("B", new Font("Arial", 16), Brushes.Black, fl_x, fl_y + ((2 * height) / 3) -
fl_lengtha - 20);
            gr.DrawString("||", new Font("Arial", 50), Brushes.Black, fl_x - 30 + width / 2, fl_y - 50 +
height / 2);
        }
        void draw_connected_disun(double x, double y, double col)
        {
            float width = pictureBox1.Width / (float)col, height = pictureBox1.Height / (float)col;
            gr = pictureBox1.CreateGraphics();
            float fl_lengtha = (float)lengtha;
            float fl_x = (float)x; float fl_y = (float)y;
    gr.DrawLine(Pens.Black, fl_x - width/10, fl_y + (height / 3), fl_x + width / 4, fl_y + (height / 3));
 gr.DrawLine(Pens.Black, fl_x - width/10, fl_y + ((2 * height) / 3), fl_x + width / 4, fl_y + ((2 *
height) / 3));
      gr.DrawLine(Pens.Black, fl_x - width / 10, fl_y + ((2 * height) / 3), fl_x - width / 10, fl_y + ((3
* height) / 3));
     gr.DrawLine(Pens.Black, fl_x - width / 10, fl_y + (( height) / 3), fl_x - width / 10, fl_y);
            gr.DrawRectangle(Pens.Black, fl_x + (width / 4), fl_y + fl_lengtha, width / 2, height - 2 *
fl_lengtha);
            gr.DrawLine(Pens.Black, fl_x + (width * 3) / 4, fl_y + height / 2, fl_x + width - 30, fl_y +
height / 2);
            gr.FillRectangle(Brushes.White, fl_x + width - 60, fl_y - 30 + height / 2, 60, 60);
            gr.DrawRectangle(Pens.Black, fl_x + width - 60, fl_y - 30 + height / 2, 60, 60);
            gr.DrawString("||", new Font("Arial", 50), Brushes.Black, fl_x - 50 + width / 2, fl_y - 50 +
height / 2);
        }
        void draw_connected_conun(double x, double y, double col)
        {
            float width = pictureBox1.Width / (float)col, height = pictureBox1.Height / (float)col;
            gr = pictureBox1.CreateGraphics();
            float fl_lengtha = (float)lengtha;
```

```
        float fl_x = (float)x; float fl_y = (float)y;
        gr.DrawLine(Pens.Black, fl_x - width / 10, fl_y + (height / 3), fl_x + width / 4, fl_y +
(height / 3));
        gr.DrawLine(Pens.Black, fl_x - width / 10, fl_y + ((2 * height) / 3), fl_x + width / 4, fl_y +
((2 * height) / 3));
        gr.DrawLine(Pens.Black, fl_x - width / 10, fl_y + ((2 * height) / 3), fl_x - width / 10, fl_y +
((3 * height) / 3));
        gr.DrawLine(Pens.Black, fl_x - width / 10, fl_y + ((height) / 3), fl_x - width / 10, fl_y);
        gr.DrawRectangle(Pens.Black, fl_x + (width / 4), fl_y + fl_lengtha, width / 2, height - 2 *
fl_lengtha);
        gr.DrawLine(Pens.Black, fl_x + (width * 3) / 4, fl_y + height / 2, fl_x + width - 30, fl_y +
height / 2);
        gr.FillRectangle(Brushes.White, fl_x + width - 60, fl_y - 30 + height / 2, 60, 60);
        gr.DrawRectangle(Pens.Black, fl_x + width - 60, fl_y - 30 + height / 2, 60, 60);
        gr.DrawString("&", new Font("Arial", 50), Brushes.Black, fl_x - 50 + width / 2, fl_y - 50 +
height / 2);
    }
    }
    private void Elektroschemes_FormClosing(object sender, FormClosingEventArgs e)
    {
        if (e.CloseReason==CloseReason.UserClosing)
        {
            e.Cancel = true;
            Hide();
        }       }
    string func = "";
    private void button1_Click(object sender, EventArgs e)
    {
        a = false; b = false;
        gr = pictureBox1.CreateGraphics();
        gr.Clear(Color.White);
        x = 0;double col = 1;
        y = 0;
        float fl_x = (float)x; float fl_y = (float)y; float fl_lengtha = (float)lengtha;
        float width = pictureBox1.Width / (float)col, height = pictureBox1.Height / (float)col;
        gr.DrawString("A", new Font("Arial", 16), Brushes.Black, fl_x, fl_y + (height / 3) -
fl_lengtha - 20);
        gr.DrawString("B", new Font("Arial", 16), Brushes.Black, fl_x, fl_y + ((2 * height) / 3) -
fl_lengtha - 20);
        label1.Text="A"+ Encoding.UTF8.GetString(Encoding.Convert(Encoding.Unicode,
Encoding.UTF8, Encoding.Unicode.GetBytes("\u2227")))+"B";
        func = "conun";
        draw_conun(x, y, col);
    }
    private void pictureBox1_MouseClick(object sender, MouseEventArgs e)
    {
        gr = pictureBox1.CreateGraphics();
        float col;
        float width, height;
        float fl_x = (float)x; float fl_y = (float)y; float fl_lengtha = (float)lengtha;
            if (func=="conun")
        {
```

```
            //a
            col = 1;
            width = pictureBox1.Width / col; height = pictureBox1.Height / col;
  if ((e.X>=x&&e.Y>= y + (height / 3) - fl_lengtha) &&(e.X<=x+2* fl_lengtha && e.Y<= y +
(height / 3) + 2*fl_lengtha) &&a==false)
          {
            gr.FillRectangle(Brushes.Orange, fl_x, fl_y + (height / 3) - fl_lengtha, 2* fl_lengtha, 2*
fl_lengtha);
            gr.DrawRectangle(Pens.Black, fl_x, fl_y + (height / 3) - fl_lengtha, 2 * fl_lengtha, 2 *
fl_lengtha);
            gr.FillRectangle(Brushes.White, fl_x, fl_y + (height / 3) - fl_lengtha - 16, 50, 16);
            gr.DrawString("A=1", new Font("Arial", 16), Brushes.Black, fl_x, fl_y + (height / 3) -
fl_lengtha - 20);
            a = true;
          }
          else if ((e.X >= x && e.Y >= y + (height / 3) - fl_lengtha) && (e.X <= x + 2* fl_lengtha
&& e.Y <= y + (height / 3) + fl_lengtha) && a == true)
          {
            gr.FillRectangle(Brushes.White, fl_x, fl_y + (height / 3) - fl_lengtha, 2* fl_lengtha, 2*
fl_lengtha);
            gr.DrawRectangle(Pens.Black, fl_x, fl_y + (height / 3) - fl_lengtha, 2* fl_lengtha, 2*
fl_lengtha);
            gr.FillRectangle(Brushes.White, fl_x, fl_y + (height / 3) - fl_lengtha -16, 50, 16);
            gr.DrawString("A=0", new Font("Arial", 16), Brushes.Black, fl_x, fl_y + (height / 3) -
fl_lengtha - 20);
            a = false;
          }
          //b

          if ((e.X >= x && e.Y >= y + 2*(height / 3) - fl_lengtha) && (e.X <= x + 2 * fl_lengtha &&
e.Y
            //b
            if ((e.X >= x && e.Y >= y + 2 * (height / 3) - fl_lengtha) && (e.X <= x + 2 * fl_lengtha
&& e.Y <= y + (2 * (height / 3)) + 2 * fl_lengtha) && b == false)
            {
              gr.FillRectangle(Brushes.Orange, fl_x, fl_y + ((2 * height) / 3) - fl_lengtha, 2 *
fl_lengtha, 2 * fl_lengtha);
              gr.DrawRectangle(Pens.Black, fl_x, fl_y + ((2 * height) / 3) - fl_lengtha, 2 *
fl_lengtha, 2 * fl_lengtha);
        gr.FillRectangle(Brushes.Orange, fl_x, fl_y + height+((2 * height) / 3) - fl_lengtha, 2 *
fl_lengtha, 2 * fl_lengtha);
          gr.DrawRectangle(Pens.Black, fl_x, fl_y + height+((2 * height) / 3) - fl_lengtha, 2 *
fl_lengtha, 2 * fl_lengtha);
              gr.FillRectangle(Brushes.White, fl_x, fl_y + (2 * height / 3) - fl_lengtha - 16, 50, 16);
             gr.DrawString("B=1", new Font("Arial", 16), Brushes.Black, fl_x, fl_y + ((2 * height) /
3) - fl_lengtha - 20);
              b = true;
            }
            else if ((e.X >= x && e.Y >= y + ((2 * height) / 3) - fl_lengtha) && (e.X <= x + 2 *
fl_lengtha && e.Y <= y + ((2 * height) / 3) + fl_lengtha) && b == true)
            {
```

```
                gr.FillRectangle(Brushes.White, fl_x, fl_y + ((2 * height) / 3) - fl_lengtha, 2 *
fl_lengtha, 2 * fl_lengtha);
                gr.DrawRectangle(Pens.Black, fl_x, fl_y + ((2 * height) / 3) - fl_lengtha, 2 *
fl_lengtha, 2 * fl_lengtha);
                gr.FillRectangle(Brushes.White, fl_x, fl_y + height+((2 * height) / 3) - fl_lengtha, 2 *
fl_lengtha, 2 * fl_lengtha);
                gr.DrawRectangle(Pens.Black, fl_x, fl_y + height +((2 * height) / 3) - fl_lengtha, 2 *
fl_lengtha, 2 * fl_lengtha);
                gr.FillRectangle(Brushes.White, fl_x, fl_y + (2 * height / 3) - fl_lengtha - 16, 50, 16);
            gr.DrawString("B=0", new Font("Arial", 16), Brushes.Black, fl_x, fl_y + ((2 * height) / 3) -
fl_lengtha - 20);
                b = false;
            }
            if (b == true)
            {
                notb = false;
            }
            else
            {
                notb = true;
            }
            //nota
            if (nota == true)
            {
                gr.FillEllipse(Brushes.Orange, fl_x + width / 4 - (fl_lengtha / 2), fl_y + height+(height
/ 3) - (fl_lengtha / 2), fl_lengtha, fl_lengtha);
                gr.DrawEllipse(Pens.Black, fl_x + width / 4 - (fl_lengtha / 2), fl_y + height+(height / 3)
- (fl_lengtha / 2), fl_lengtha, fl_lengtha);
            }
            else
            {
                gr.FillEllipse(Brushes.White, fl_x + width / 4 - (fl_lengtha / 2), fl_y + height + (height
/ 3) - (fl_lengtha / 2), fl_lengtha, fl_lengtha);
                gr.DrawEllipse(Pens.Black, fl_x + width / 4 - (fl_lengtha / 2), fl_y + height + (height /
3) - (fl_lengtha / 2), fl_lengtha, fl_lengtha);
            }
            //notb
            if (notb == true)
            {
                gr.FillEllipse(Brushes.Orange, fl_x + width / 4 - (fl_lengtha / 2), fl_y +
height+(2*height / 3) - (fl_lengtha / 2), fl_lengtha, fl_lengtha);
                gr.DrawEllipse(Pens.Black, fl_x + width / 4 - (fl_lengtha / 2), fl_y + height+(2*height /
3) - (fl_lengtha / 2), fl_lengtha, fl_lengtha);
            }
            else
            {
                gr.FillEllipse(Brushes.White, fl_x + width / 4 - (fl_lengtha / 2), fl_y + height + (2 *
height / 3) - (fl_lengtha / 2), fl_lengtha, fl_lengtha);
                gr.DrawEllipse(Pens.Black, fl_x + width / 4 - (fl_lengtha / 2), fl_y + height + (2 *
height / 3) - (fl_lengtha / 2), fl_lengtha, fl_lengtha);
            }
            //res
```

```
            bool res1=false,res2=false;
                  if (a && b == true)
            {
               res1 = true;
            }
            else
            {
               res1 = false;
            }
            if (nota && notb == true)
            {
               res2 = true;
            }
            else
            {
               res2 = false;
            }
            if (res1 == true)
            {
               gr.FillRectangle(Brushes.Green, fl_x + width - 60, fl_y - 30 + height / 2, 60, 60);
               gr.DrawRectangle(Pens.Black, fl_x + width - 60, fl_y - 30 + height / 2, 60, 60);
               gr.FillRectangle(Brushes.White, fl_x + width - 40, fl_y - 70 + height / 2, 20, 30);
              gr.DrawString("1", new Font("Arial", 20), Brushes.Green, fl_x + width - 40, fl_y - 70 +
height / 2);
            }
            else
            {
               gr.FillRectangle(Brushes.White, fl_x + width - 60, fl_y - 30 + height / 2, 60, 60);
               gr.DrawRectangle(Pens.Black, fl_x + width - 60, fl_y - 30 + height / 2, 60, 60);
               gr.FillRectangle(Brushes.White, fl_x + width - 40, fl_y - 70 + height / 2, 20, 30);
               gr.DrawString("0", new Font("Arial", 20), Brushes.Black, fl_x + width - 40, fl_y - 70 +
height / 2);
            }
            if (res2 == true)
            {
               gr.FillRectangle(Brushes.Green, fl_x + width - 60, fl_y - 30 +height+ height / 2, 60,
60);
               gr.DrawRectangle(Pens.Black, fl_x + width - 60, fl_y - 30 + height+height / 2, 60, 60);
               gr.FillRectangle(Brushes.White, fl_x + width - 40, fl_y - 70 +height+ height / 2, 20,
30);
               gr.DrawString("1", new Font("Arial", 20), Brushes.Green, fl_x + width - 40, fl_y - 70
+height+ height / 2);
            }
            else
            {
               gr.FillRectangle(Brushes.White, fl_x + width - 60, fl_y - 30 +height+ height / 2, 60,
60);
               gr.DrawRectangle(Pens.Black, fl_x + width - 60, fl_y - 30 + height+height / 2, 60, 60);
               gr.FillRectangle(Brushes.White, fl_x + width - 40, fl_y - 70 +height+ height / 2, 20,
30);
               gr.DrawString("0", new Font("Arial", 20), Brushes.Black, fl_x + width - 40, fl_y - 70 +
height+height / 2);
```

```
            }
            if (res1||res2==true)
            {
               res = true;
            }
            else
            {
              res = false;
            }
            if (res == true)
            {
               gr.FillRectangle(Brushes.Green, fl_x + 2 * width - 60, fl_y - 30 +2*height / 2, 60, 60);
               gr.DrawRectangle(Pens.Black, fl_x+ 2 * width - 60, fl_y - 30 + 2 * height / 2, 60, 60);
               gr.FillRectangle(Brushes.White, fl_x +width+ width - 40, fl_y - 70 + 2 * height / 2, 20,
30);
           gr.DrawString("1", new Font("Arial", 20), Brushes.Green, fl_x +width+ width - 40, fl_y -
70 + 2 * height / 2);
            }
            else
            {
               gr.FillRectangle(Brushes.White, fl_x + 2*width - 60, fl_y - 30 + 2 * height / 2, 60, 60);
               gr.DrawRectangle(Pens.Black, fl_x + 2 * width - 60, fl_y - 30 + 2 * height / 2, 60, 60);
               gr.FillRectangle(Brushes.White, fl_x + width + width - 40, fl_y - 70 + 2 * height / 2,
20, 30);
               gr.DrawString("0", new Font("Arial", 20), Brushes.Black, fl_x + 2 * width - 40, fl_y -
70 + 2 * height / 2);
            }
         }
         else if(func=="xor")
         {
                  //a
            bool nota, notb;
            col = 2;
            width = pictureBox1.Width / col; height = pictureBox1.Height / col;
            if ((e.X >= x && e.Y >= y + (height / 3) - fl_lengtha) && (e.X <= x + 2 * fl_lengtha
&& e.Y <= y + (height / 3) + 2 * fl_lengtha) && a == false)
            {
               gr.FillRectangle(Brushes.Orange, fl_x, fl_y + (height / 3) - fl_lengtha, 2 * fl_lengtha,
2 * fl_lengtha);
               gr.DrawRectangle(Pens.Black, fl_x, fl_y + (height / 3) - fl_lengtha, 2 * fl_lengtha, 2
* fl_lengtha);
        gr.FillRectangle(Brushes.Orange, fl_x, fl_y + height + (height / 3) - fl_lengtha, 2 *
fl_lengtha, 2 * fl_lengtha);
              gr.DrawRectangle(Pens.Black, fl_x, fl_y + height + (height / 3) - fl_lengtha, 2 *
fl_lengtha, 2 * fl_lengtha);
               gr.FillRectangle(Brushes.White, fl_x, fl_y + (height / 3) - fl_lengtha - 16, 50, 16);
               gr.DrawString("A=1", new Font("Arial", 16), Brushes.Black, fl_x, fl_y + (height / 3)
- fl_lengtha - 20);
               a = true;
            }
            else if ((e.X >= x && e.Y >= y + (height / 3) - fl_lengtha) && (e.X <= x + 2 *
fl_lengtha && e.Y <= y + (height / 3) + fl_lengtha) && a == true)
```

```
                {
                    gr.FillRectangle(Brushes.White, fl_x, fl_y + (height / 3) - fl_lengtha, 2 * fl_lengtha,
2 * fl_lengtha);
                    gr.DrawRectangle(Pens.Black, fl_x, fl_y + (height / 3) - fl_lengtha, 2 * fl_lengtha, 2
* fl_lengtha);

                gr.FillRectangle(Brushes.White, fl_x, fl_y + height + (height / 3) - fl_lengtha, 2 *
fl_lengtha, 2 * fl_lengtha);
                 gr.DrawRectangle(Pens.Black, fl_x, fl_y + height + (height / 3) - fl_lengtha, 2 *
fl_lengtha, 2 * fl_lengtha);
                  gr.FillRectangle(Brushes.White, fl_x, fl_y + (height / 3) - fl_lengtha - 16, 50, 16);
                    gr.DrawString("A=0", new Font("Arial", 16), Brushes.Black, fl_x, fl_y + (height / 3)
- fl_lengtha - 20);
                    a = false;
                }
                if (a == true)
                {
                    nota = false;
                }
                else
                {
                    nota = true;
                }
                //b
                if ((e.X >= x && e.Y >= y + 2 * (height / 3) - fl_lengtha) && (e.X <= x + 2 *
fl_lengtha && e.Y <= y + (2 * (height / 3)) + 2 * fl_lengtha) && b == false)
                {
               gr.FillRectangle(Brushes.Orange, fl_x, fl_y + ((2 * height) / 3) - fl_lengtha, 2 *
fl_lengtha, 2 * fl_lengtha);
                    gr.DrawRectangle(Pens.Black, fl_x, fl_y + ((2 * height) / 3) - fl_lengtha, 2 *
fl_lengtha, 2 * fl_lengtha);
     gr.FillRectangle(Brushes.Orange, fl_x, fl_y + height + ((2 * height) / 3) - fl_lengtha, 2 *
fl_lengtha, 2 * fl_lengtha);
  gr.DrawRectangle(Pens.Black, fl_x, fl_y + height + ((2 * height) / 3) - fl_lengtha, 2 * fl_lengtha, 2
* fl_lengtha);
                    gr.FillRectangle(Brushes.White, fl_x, fl_y + (2 * height / 3) - fl_lengtha - 16, 50, 16);
    gr.DrawString("B=1", new Font("Arial", 16), Brushes.Black, fl_x, fl_y + ((2 * height) / 3) -
fl_lengtha - 20);
                    b = true;
                }
                else if ((e.X >= x && e.Y >= y + ((2 * height) / 3) - fl_lengtha) && (e.X <= x + 2 *
fl_lengtha && e.Y <= y + ((2 * height) / 3) + fl_lengtha) && b == true)
                {
gr.FillRectangle(Brushes.White, fl_x, fl_y + ((2 * height) / 3) - fl_lengtha, 2 * fl_lengtha, 2 *
fl_lengtha);
                    gr.DrawRectangle(Pens.Black, fl_x, fl_y + ((2 * height) / 3) - fl_lengtha, 2 *
fl_lengtha, 2 * fl_lengtha);
  gr.FillRectangle(Brushes.White, fl_x, fl_y + height + ((2 * height) / 3) - fl_lengtha, 2 * fl_lengtha,
2 * fl_lengtha);
  gr.DrawRectangle(Pens.Black, fl_x, fl_y + height + ((2 * height) / 3) - fl_lengtha, 2 * fl_lengtha, 2
* fl_lengtha);
```

```
                gr.FillRectangle(Brushes.White, fl_x, fl_y + (2 * height / 3) - fl_lengtha - 16, 50,
16);
            gr.DrawString("B=0", new Font("Arial", 16), Brushes.Black, fl_x, fl_y + ((2 * height) /
3) - fl_lengtha - 20);
                b = false;
              }
              if (b == true)
              {
                notb = false;
              }
              else
              {
                notb = true;
              }
              //nota
              if (nota == true)
              {
      {
         button1.Text = Encoding.UTF8.GetString(Encoding.Convert(Encoding.Unicode,
Encoding.UTF8, Encoding.Unicode.GetBytes("\u2227")))+ " (and)";
         button2.Text = Encoding.UTF8.GetString(Encoding.Convert(Encoding.Unicode,
Encoding.UTF8, Encoding.Unicode.GetBytes("\u2228"))) + " (or)";
         button3.Text = Encoding.UTF8.GetString(Encoding.Convert(Encoding.Unicode,
Encoding.UTF8, Encoding.Unicode.GetBytes("\u21D2"))) + " (if-then)";
         button4.Text = Encoding.UTF8.GetString(Encoding.Convert(Encoding.Unicode,
Encoding.UTF8, Encoding.Unicode.GetBytes("\u00AC"))) + " (not)";
         button5.Text = "↔" + " (equal)";
         button6.Text= Encoding.UTF8.GetString(Encoding.Convert(Encoding.Unicode,
Encoding.UTF8, Encoding.Unicode.GetBytes("\u2295"))) + " (xor)";
      }   }}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace Kval_
{
   public partial class Калькулятор : Form
   {
      Закони frmzak = new Закони();
      Elektroschemes elektro = new Elektroschemes();
      public Калькулятор()
      {
         InitializeComponent();
      }
       private void Form1_Load(object sender, EventArgs e)
      {
```

```
        button_disagr.Text = Encoding.UTF8.GetString(Encoding.Convert(Encoding.Unicode,
Encoding.UTF8, Encoding.Unicode.GetBytes("\u00AC")));
        buttton_conun.Text = Encoding.UTF8.GetString(Encoding.Convert(Encoding.Unicode,
Encoding.UTF8, Encoding.Unicode.GetBytes("\u2227")));
        button_disun.Text= Encoding.UTF8.GetString(Encoding.Convert(Encoding.Unicode,
Encoding.UTF8, Encoding.Unicode.GetBytes("\u2228")));
        button_impl.Text = Encoding.UTF8.GetString(Encoding.Convert(Encoding.Unicode,
Encoding.UTF8, Encoding.Unicode.GetBytes("\u21D2")));
        textBox_main.Text = "A"+con+"B";
    }
    string destodv(int a)
    {
        string ans = "";
        while (a != 0)
        {
            ans += (a % 2).ToString();
            a /= 2;
        }
        return ans;
    }
    string con = Encoding.UTF8.GetString(Encoding.Convert(Encoding.Unicode,
Encoding.UTF8, Encoding.Unicode.GetBytes("\u2227")));
    string disun = Encoding.UTF8.GetString(Encoding.Convert(Encoding.Unicode,
Encoding.UTF8, Encoding.Unicode.GetBytes("\u2228")));
    string impl = Encoding.UTF8.GetString(Encoding.Convert(Encoding.Unicode,
Encoding.UTF8, Encoding.Unicode.GetBytes("\u21D2")));
    string equal = "↔";
    string disagr = Encoding.UTF8.GetString(Encoding.Convert(Encoding.Unicode,
Encoding.UTF8, Encoding.Unicode.GetBytes("\u00AC")));
    bool is_log_op(char a)
    {
        string c = a.ToString();
        if (c == con) return true;
        if (c == disun) return true;
        if (c == equal) return true;
        if (c == disagr) return true;
        if (c == impl) return true;
        return false;
    }
    int colzminn = 0;
    private void button1_Click(object sender, EventArgs e)
    {
        colzminn = 0;
        string vir = textBox_main.Text;
        string con = Encoding.UTF8.GetString(Encoding.Convert(Encoding.Unicode,
Encoding.UTF8, Encoding.Unicode.GetBytes("\u2227")));
        string disun = Encoding.UTF8.GetString(Encoding.Convert(Encoding.Unicode,
Encoding.UTF8, Encoding.Unicode.GetBytes("\u2228")));
        string impl = Encoding.UTF8.GetString(Encoding.Convert(Encoding.Unicode,
Encoding.UTF8, Encoding.Unicode.GetBytes("\u21D2"))); ;
        string equal = "↔";
```

```
        string disagr = Encoding.UTF8.GetString(Encoding.Convert(Encoding.Unicode,
Encoding.UTF8, Encoding.Unicode.GetBytes("\u00AC")));
        string xor = Encoding.UTF8.GetString(Encoding.Convert(Encoding.Unicode,
Encoding.UTF8, Encoding.Unicode.GetBytes("\u2295")));
        int colotkskob = 0, colzakskob = 0;
        if (textBox_main.Text!="")
        {
        for (int i = 0; i < vir.Length; i++)
        {
           if (vir[i] == '(')
           {
              colotkskob++;
           }
           if (vir[i] == ')')
           {
              colzakskob++;
           }
        }
       else
        {
           if ((colotkskob != colzakskob))
           {
           MessageBox.Show("Неправильно розставлені скобки!");
              dataGridView1.Rows.Clear();
              dataGridView1.ColumnCount = 0;
              return;
           }
         else
        {
           {
              if (textBox_main.Text.Contains("A"))
              {
                  colzminn++;
              }
              if (textBox_main.Text.Contains("B"))
              {
                 colzminn++;
              }
              if (textBox_main.Text.Contains("C"))
              {
                colzminn++;
              }
              if (textBox_main.Text.Contains("D"))
              {
              colzminn++;
              }
              if (((colzminn == 3) || (colzminn == 4)) && ((colzakskob == 0) || (colotkskob == 0)))
              {
                 MessageBox.Show("Неправильная скобочная последовательность!");
                 dataGridView1.Rows.Clear();
                 dataGridView1.ColumnCount = 0;
                 return;
```

```
  }
  else
{
      dataGridView1.RowCount = (int)Math.Pow((double)2, (double)colzminn);
      int zn = (int)Math.Pow((double)2, (double)colzminn);
      dataGridView1.ColumnCount = colzminn;
      string[] names = { "A", "B", "C", "D" };
      for (int i = 0; i < colzminn; i++)
      {
         dataGridView1.Columns[i].HeaderText = names[i];
      }
      for (int i = 0; i < zn; i++)
      {
         for (int j = 0; j < colzminn; j++)
         {

            if (i == 0)
            {
               dataGridView1.Rows[i].Cells[j].Value = 0;
            }                        }                    }
      for (int i = 0; i < zn; i++)
      {
         string cur = destodv(zn - i - 1);
         for (int j = 0; j < colzminn; j++)
         {
               if (j < cur.Length)
               {
                  dataGridView1.Rows[i].Cells[j].Value = cur[j] - '0';
               }
               else
               {
                  dataGridView1.Rows[i].Cells[j].Value = 0;
               }                        }                    }
      Stack<char> ans = new Stack<char>();
      string[] log_op = { con, disun, impl, equal, disagr };
      int colskob = 1, dd = 0;
      for (int i = 0; i < textBox_main.Text.Length; i++)
      {
         if (textBox_main.Text[i] == '(')
         {
            colskob++;
         }
         if (textBox_main.Text[i].ToString() == disagr)
         {
            ++dd;
         }                    }
      //MessageBox.Show(dataGridView1.ColumnCount.ToString());
      dataGridView1.ColumnCount = colskob + colzminn + dd;
      Stack<int> q = new Stack<int>();
      int k = 1;
      for (int i = 0; i < textBox_main.Text.Length; i++)
      {
```

```
                    if (textBox_main.Text[i] == '(')
                        q.Push(i);
                    else if (textBox_main.Text[i] == ')')
                    {
                        int newcolstolb1 = colzminn + k;
                        if (newcolstolb1 != dataGridView1.ColumnCount)
 dataGridView1.Columns[newcolstolb1 - 1].HeaderText = textBox_main.Text.Substring(q.First() +
1, i - q.First() - 1);
                        q.Pop();
                        ++k;
                    }
                    if (textBox_main.Text[i].ToString() == disagr)
                    {
                        q.Push(i);
                    }
                    else if (q.Count() > 0 && textBox_main.Text[q.First()].ToString() == disagr)
                    {
                        int newcolstolb1 = colzminn + k;
                        if (newcolstolb1 != dataGridView1.ColumnCount)
  dataGridView1.Columns[newcolstolb1 - 1].HeaderText = textBox_main.Text.Substring(q.First(),
i - q.First() + 1);
                        q.Pop();
                        ++k;
                    }
                }
                for (int num = 0; num < zn; num++)
                {
                    k = 1;
                    string S = "("; string stitle = "";
                    for (int i = 0; i < textBox_main.Text.Length; i++)
                    {
                        stitle += textBox_main.Text[i];
                        if (textBox_main.Text[i] >= 'A' && textBox_main.Text[i] <= 'D')
                        {
                            S += dataGridView1.Rows[num].Cells[textBox_main.Text[i] - 'A'].Value;
                        }
                        else S += textBox_main.Text[i];
                    }
                    S += ")";
                    // MessageBox.Show(S);
                    for (int i = 0; i < S.Length; i++)
                    {
                        if (S[i] != ')')
                        {
                            ans.Push(S[i]);
                        }
                        else
                        {
                            char c = ans.First();
                            string sum1 = "";
                            while (c != '(')
                            {
```

```
    sum1 += c;
    ans.Pop();
    c = ans.First();
}
string sum = "";
for (int l = sum1.Count() - 1; l >= 0; l--)
{
    sum += sum1[l];
}
ans.Pop();
//MessageBox.Show(sum);
Stack<string> last = new Stack<string>();
char cur = '0';
for (int j = 0; j < sum.Length; j++)
{
    if (is_log_op(sum[j]) == true)
    {
        last.Push(sum[j].ToString());
    }
    else
    {
        char ch = sum[j];
        while (last.Count != 0)
        {
            string op1 = last.First();
            if (op1 != disagr) break;
            last.Pop();
            if (ch == '1')
            {
                ch = '0';
            }
            else
            {
                ch = '1';
            }
            int newcolstolb2 = colzminn + k;
            dataGridView1.Rows[num].Cells[newcolstolb2 - 1].Value = ch;
            ++k;
        }
        string op = "";
        if (last.Count == 0) {
            cur = ch;
        }
        else
        {
            op = last.First();
            last.Pop();
        }
        if (op == con)
        {
            if (cur == ch && cur == '1')
            {
```

```
                    cur = '1';
                }
                else
                {
                    cur = '0';
                }                                   }

            if (op == disun)
            {
              if (cur == ch && cur == '0')
                {
                    cur = '0';
                }
                else
                {
                    cur = '1';
                }                                   }

            if (op == impl)
            {
              if (cur == '1' && ch == '0')
                {
                    cur = '0';
                }
                else cur = '1';
            }

            if (op == equal)
            {
                if (cur == ch)
                {
                    cur = '1';
                }
                else cur = '0';
            }                                   }                       }
        //MessageBox.Show(cur.ToString());
        int newcolstolb1 = colzminn + k;
        ans.Push(cur.ToString()[0]);
        dataGridView1.Rows[num].Cells[newcolstolb1 - 1].Value = cur;
        ++k;
      }                     }
    // MessageBox.Show(ans.First().ToString());
    ans.Pop();
  }
  int newcolstolb = dataGridView1.ColumnCount - 1;
    for (int i = 0; i < newcolstolb; i++)
    {
      dataGridView1.Columns[i].DefaultCellStyle.ForeColor = Color.Black;
    }
    dataGridView1.Columns[newcolstolb].DefaultCellStyle.ForeColor = Color.Red;
  dataGridView1.Columns[newcolstolb].HeaderText = textBox_main.Text;
else return;
```

```
            }
        private void textBox_main_TextChanged(object sender, EventArgs e)
        {
            if (textBox_main.Text.Contains("A"))
            {
                button_B.Enabled = true;
            }
            if (textBox_main.Text.Contains("A") && textBox_main.Text.Contains("B"))
            {
                button_C.Enabled = true;
            }
    if      (textBox_main.Text.Contains("A")      &&      textBox_main.Text.Contains("B")      &&
textBox_main.Text.Contains("C"))
            {
                button_D.Enabled = true;
            }       }
        private void button4_Click(object sender, EventArgs e)
        {

            frmzak.StartPosition = FormStartPosition.CenterScreen;
            frmzak.Show();
        }
        private void button5_Click(object sender, EventArgs e)
        {
            elektro.StartPosition = FormStartPosition.CenterScreen;
            elektro.Show();
        }
        private void Калькулятор_FormClosing(object sender, FormClosingEventArgs e)
        {
            if (e.CloseReason == CloseReason.UserClosing)
            {
                e.Cancel = true;
                Hide();
            }       }          }}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace Kval_
{
    public partial class Form_main : Form
    {
        public Form_main()
        {
            InitializeComponent();
        }
```

```
Калькулятор frmcal = new Калькулятор();
Закони frmazakon = new Закони();
Elektroschemes frmelek = new Elektroschemes();
private void button1_Click(object sender, EventArgs e)
{
   frmcal.StartPosition = FormStartPosition.CenterScreen;
   frmcal.Show();
}
private void button1_MouseHover(object sender, EventArgs e)
{
   button1.BackColor=Color.DarkOrange;
}
private void button1_MouseLeave(object sender, EventArgs e)
{
   button1.BackColor = Color.SandyBrown;
}
private void button2_MouseHover(object sender, EventArgs e)
{
   button2.BackColor = Color.DarkOrange;
}
private void button2_MouseLeave(object sender, EventArgs e)
{
   button2.BackColor = Color.SandyBrown;
}
private void button3_MouseHover(object sender, EventArgs e)
{
   button3.BackColor = Color.DarkOrange;
}
private void button3_Leave(object sender, EventArgs e)
{
}
private void button3_MouseLeave(object sender, EventArgs e)
{
   button3.BackColor = Color.SandyBrown;
}
private void button2_Click(object sender, EventArgs e)
{
   frmazakon.StartPosition = FormStartPosition.CenterScreen;
   frmazakon.Show();
}
private void button3_Click(object sender, EventArgs e)
{
   frmelek.StartPosition = FormStartPosition.CenterScreen;
   frmelek.Show();
}
private void Form_main_Load(object sender, EventArgs e)
{
}   }}

using System;
using System.Collections.Generic;
using System.ComponentModel;
```

```
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace Kval_
{
   public partial class Закони : Form
   {
      public Закони()
      {
         InitializeComponent();
      }
      string[] data1;
      string[] data2;
string con = Encoding.UTF8.GetString(Encoding.Convert(Encoding.Unicode, Encoding.UTF8,
Encoding.Unicode.GetBytes("\u2227")));
      string disun = Encoding.UTF8.GetString(Encoding.Convert(Encoding.Unicode,
Encoding.UTF8, Encoding.Unicode.GetBytes("\u2228")));
      string impl = Encoding.UTF8.GetString(Encoding.Convert(Encoding.Unicode,
Encoding.UTF8, Encoding.Unicode.GetBytes("\u21D2")));
      string equal = "↔";
      string disagr = Encoding.UTF8.GetString(Encoding.Convert(Encoding.Unicode,
Encoding.UTF8, Encoding.Unicode.GetBytes("\u00AC")));
      bool is_log_op(char a)
      {
         string c = a.ToString();
         if (c == con) return true;
         if (c == disun) return true;
         if (c == equal) return true; if (c == disagr) return true;
         if (c == impl) return true;
         return false;
      }
      string destodv(int a)
      {
         string ans = "";
         while (a != 0)
         {
            ans += (a % 2).ToString();
            a /= 2;
         }
         return ans;
      }
      private void Form2_Load(object sender, EventArgs e)
      {
         comboBox1.Text = "Коммутативність";
         textBox1.Text = "A" + con + "B";
         textBox2.Text = "B" + con + "A";
      }
      private void radioButton1_CheckedChanged(object sender, EventArgs e)
      {
```

```
}
private void textBox1_TextChanged(object sender, EventArgs e)
{
   dataGridView1.Rows.Clear();
   int colzminn = 0;
   if (textBox1.Text.Contains("A"))
   {
      colzminn++;
   }
   if (textBox1.Text.Contains("B"))
   {
      colzminn++;
   }
   if (textBox1.Text.Contains("C"))
   {
      colzminn++;
   }
   if (textBox1.Text.Contains("D"))
   {
      colzminn++;
   }
   int zn = (int)Math.Pow((double)2, (double)colzminn);
   dataGridView1.RowCount = zn;
   dataGridView1.ColumnCount = colzminn;
   string[] names = { "A", "B", "C", "D" };
          for (int i = 0; i < colzminn; i++)
   {
      dataGridView1.Columns[i].HeaderText = names[i];
   }
   for (int i = 0; i < zn; i++)
   {
      for (int j = 0; j < colzminn; j++)
      {
        if (i == 0)
          {
             dataGridView1.Rows[i].Cells[j].Value = 0;
          }              }           }
   for (int i = 0; i < zn; i++)
   {
      string cur = destodv(zn - i - 1);
      for (int j = 0; j < colzminn; j++)
      {
         dataGridView1.Rows[i].Cells[j].Value = j < cur.Length ? cur[j] - '0' : 0;
      }           }
   char[] a = new char[textBox1.Text.Length];
   for (int i = 0; i < textBox1.Text.Length; i++)
   {
      a[i] = textBox1.Text[i];
   }
   Stack<char> ans = new Stack<char>();
   string[] log_op = { con, disun, impl, equal, disagr };
   int colskob = 1, dd = 0;
```

```
        for (int i = 0; i < textBox1.Text.Length; i++)
        {
           if (textBox1.Text[i] == '(')
           {
              colskob++;
           }
           if (textBox1.Text[i].ToString() == disagr) dd++;
        }
        //MessageBox.Show(dataGridView1.ColumnCount.ToString());
        dataGridView1.ColumnCount = colskob + colzminn + dd;
        Stack<int> q = new Stack<int>();
        int k = 1;
        for (int i = 0; i < textBox1.Text.Length; i++)
        {
           if (textBox1.Text[i] == '('
              q.Push(i);
           else if (textBox1.Text[i] == ')')
           {
              int newcolstolb1 = colzminn + k;
              if (newcolstolb1 != dataGridView1.ColumnCount)
       dataGridView1.Columns[newcolstolb1 - 1].HeaderText = textBox1.Text.Substring(q.First()
+ 1, i - q.First() - 1);
                 q.Pop();
           ++k;
           }
           if (textBox1.Text[i].ToString() == disagr)
           {
              q.Push(i);
           }
           else if (q.Count() > 0 && textBox1.Text[q.First()].ToString() == disagr)
           {
              int newcolstolb1 = colzminn + k;
              if (newcolstolb1 != dataGridView1.ColumnCount)
         dataGridView1.Columns[newcolstolb1 - 1].HeaderText =
textBox1.Text.Substring(q.First(), i - q.First() + 1);
              q.Pop();
              ++k;
           }          }
        for (int num = 0; num < zn; num++)
        {
           k = 1;
           string S = "("; string stitle = "";
           for (int i = 0; i < textBox1.Text.Length; i++)
           {
              stitle += textBox1.Text[i];
              if (textBox1.Text[i] >= 'A' && textBox1.Text[i] <= 'D')
              {
                 S += dataGridView1.Rows[num].Cells[textBox1.Text[i] - 'A'].Value;
              }
              else S += textBox1.Text[i];
           }
           S += ")";
```

```
// MessageBox.Show(S);
for (int i = 0; i < S.Length; i++)
{
   if (S[i] != ')')
   {
      ans.Push(S[i]);
   }
   else
   {
      char c = ans.First();
      string sum1 = "";
      while (c != '(')
      {
         sum1 += c;
         ans.Pop();
         c = ans.First();
      }
      string sum = "";
      for (int l = sum1.Count() - 1; l >= 0; l--)
      {
         sum += sum1[l];
      }
      ans.Pop();
      //MessageBox.Show(sum);
      Stack<string> last = new Stack<string>();
      char cur = '0';
      for (int j = 0; j < sum.Length; j++)
      {
         if (is_log_op(sum[j]) == true)
         {
            last.Push(sum[j].ToString());
         }
         else
         {
            char ch = sum[j];
            while (last.Count != 0)
            {
               string op1 = last.First();
               if (op1 != disagr) break;
               last.Pop();
               if (ch == '1')
               {
                  ch = '0';
               }
               else
               {
                  ch = '1';
               }
               int newcolstolb2 = colzminn + k;
               dataGridView1.Rows[num].Cells[newcolstolb2 - 1].Value = ch;
               ++k;
            }
```

```
            string op = "";
            if (last.Count == 0)
            {
                cur = ch;
            }
            else
            {
                op = last.First();
                last.Pop();
            }
            if (op == con)
            {
                if (cur == ch && cur == '1')
                {
                    cur = '1';
                }
                else
                {
                    cur = '0';
                }                           }
            if (op == disun)
            {
                if (cur == ch && cur == '0')
                {
                    cur = '0';
                }
                else
                {
                    cur = '1';
                }                           }
            if (op == impl)
            {
                if (cur == '1' && ch == '0')
                {
                    cur = '0';
                }
                else cur = '1';
            }
            if (op == equal)
            {
                if (cur == ch)
                {
                    cur = '1';
                }
                else cur = '0';
            }                       }                   }
        //MessageBox.Show(cur.ToString());
        int newcolstolb1 = colzminn + k;
        ans.Push(cur.ToString()[0]);
        dataGridView1.Rows[num].Cells[newcolstolb1 - 1].Value = cur;
        ++k;
    }
```

```
            }
            // MessageBox.Show(ans.First().ToString());
            ans.Pop();
        }
        if (dataGridView1.Columns[dataGridView1.ColumnCount-2].HeaderText==
dataGridView1.Columns[dataGridView1.ColumnCount - 1].HeaderText)
        {
            dataGridView1.ColumnCount--;
        }
        int newcolstolb = dataGridView1.ColumnCount - 1;
        for (int i = 0; i < newcolstolb; i++)
        {
            dataGridView1.Columns[i].DefaultCellStyle.ForeColor = Color.Black;
        }
        dataGridView1.Columns[newcolstolb].DefaultCellStyle.ForeColor = Color.Blue;
        dataGridView1.Columns[newcolstolb].HeaderText = textBox1.Text;
        data1 = new string[zn];
        for (int i = 0; i < zn; i++)
        {
            data1[i] = dataGridView1.Rows[i].Cells[dataGridView1.ColumnCount-
1].Value.ToString();
        }
    }
    private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
    {
            if (radioButton1.Checked == true)
        {
            if (comboBox1.Text == "Тавтологія")
            {
                textBox1.Text = "A" + con + "A";
                textBox2.Text = "A";
            }
            if (comboBox1.Text == "Коммутативність")
            {
                textBox1.Text = "A" + con + "B";
                textBox2.Text = "B" + con + "A";
            }
            if (comboBox1.Text == "Асоціативність")
            {
                textBox1.Text = "(A" + con + "B)" + con + "C";
                textBox2.Text = "A" + con + "(B" + con + "C)";
            }
            if (comboBox1.Text == "Дистрибутивність")
            {
                textBox1.Text = "A" + con + "(B" + disun + "C)";
                textBox2.Text = "(A" + con + "B)" + disun + "(A" + con + "C)";
            }
            if (comboBox1.Text == "Закон де Моргана")
            {
                textBox1.Text = disagr + "(A" + con + "B)";
                textBox2.Text = disagr + "A" + disun + disagr + "B";
            }
```

```
        }           }       }
private void textBox2_TextChanged(object sender, EventArgs e)
{
    int colzminn = 0;
    if (textBox2.Text.Contains("A"))
    {
        colzminn++;
    }
    if (textBox2.Text.Contains("B"))
    {
        colzminn++;
    }
    if (textBox2.Text.Contains("C"))
    {
        colzminn++;
    }
    if (textBox2.Text.Contains("D"))
    {
        colzminn++;
    }
    if (comboBox1.Text == "Закон поглинання"|| comboBox1.Text == "Закон склеювання")
    {
        colzminn = 2;
    }
    dataGridView2.RowCount = (int)Math.Pow((double)2, (double)colzminn);
    int zn = (int)Math.Pow((double)2, (double)colzminn);
    dataGridView2.ColumnCount = colzminn;
    string[] names = { "A", "B", "C", "D" };
    for (int i = 0; i < colzminn; i++)
    {
        dataGridView2.Columns[i].HeaderText = names[i];
    }
    for (int i = 0; i < zn; i++)
    {
        for (int j = 0; j < colzminn; j++)
        {
            if (i == 0)
            {
                dataGridView2.Rows[i].Cells[j].Value = 0;
            }               }           }
    for (int i = 0; i < zn; i++)
    {
        string cur = destodv(zn - i - 1);
        for (int j = 0; j < colzminn; j++)
        {
            dataGridView2.Rows[i].Cells[j].Value = j < cur.Length ? cur[j] - '0' : 0;
        }           }
    char[] a = new char[textBox2.Text.Length];
    for (int i = 0; i < textBox2.Text.Length; i++)
    {
        a[i] = textBox2.Text[i];
    }
```

```
        Stack<char> ans = new Stack<char>();
        string[] log_op = { con, disun, impl, equal, disagr };
        int colskob = 1, dd = 0;
        for (int i = 0; i < textBox2.Text.Length; i++)
        {
           if (textBox2.Text[i] == '(')
           {
              colskob++;
           }
           if (textBox2.Text[i].ToString() == disagr) ++dd;
        }
        //MessageBox.Show(dataGridView2.ColumnCount.ToString());
        dataGridView2.ColumnCount = colskob + colzminn + dd;
        Stack<int> q = new Stack<int>();
        int k = 1;
        for (int i = 0; i < textBox2.Text.Length; i++)
        {
           if (textBox2.Text[i] == '('
              q.Push(i);
           else if (textBox2.Text[i] == ')')
           {
              int newcolstolb1 = colzminn + k;
              if (newcolstolb1 != dataGridView2.ColumnCount)
       dataGridView2.Columns[newcolstolb1 - 1].HeaderText = textBox2.Text.Substring(q.First()
+ 1, i - q.First() - 1);
              q.Pop();
              ++k;
           }
           if (textBox2.Text[i].ToString() == disagr)
           {
              q.Push(i);
           }
           else if (q.Count() > 0 && textBox2.Text[q.First()].ToString() == disagr)
           {
              int newcolstolb1 = colzminn + k;
              if (newcolstolb1 != dataGridView2.ColumnCount)
    dataGridView2.Columns[newcolstolb1 - 1].HeaderText = textBox2.Text.Substring(q.First(), i -
q.First() + 1);
              q.Pop();
              ++k;
           }          }
        for (int num = 0; num < zn; num++)
        {
           k = 1;
           string S = "("; string stitle = "";
           for (int i = 0; i < textBox2.Text.Length; i++)
           {
              stitle += textBox2.Text[i];
              if (textBox2.Text[i] >= 'A' && textBox2.Text[i] <= 'D')
              {
                 S += dataGridView2.Rows[num].Cells[textBox2.Text[i] - 'A'].Value;
              }
```

```
    else S += textBox2.Text[i];
}
S += ")";
// MessageBox.Show(S);
for (int i = 0; i < S.Length; i++)
{
    if (S[i] != ')')
    {
        ans.Push(S[i]);
    }
    else
    {
        char c = ans.First();
        string sum1 = "";
        while (c != '(')
        {
            sum1 += c;
            ans.Pop();
            c = ans.First();
        }
        string sum = "";
        for (int l = sum1.Count() - 1; l >= 0; l--)
        {
            sum += sum1[l];
        }
        ans.Pop();
        //MessageBox.Show(sum);
        Stack<string> last = new Stack<string>();
        char cur = '0';
        for (int j = 0; j < sum.Length; j++)
        {
            if (is_log_op(sum[j]) == true)
            {
                last.Push(sum[j].ToString());
            }
            else
            {
                char ch = sum[j];
                while (last.Count != 0)
                {
                    string op1 = last.First();
                    if (op1 != disagr) break;
                    last.Pop();
                    if (ch == '1')
                    {
                        ch = '0';
                    }
                    else
                    {
                        ch = '1';
                    }
                    int newcolstolb2 = colzminn + k;
```

```
            dataGridView2.Rows[num].Cells[newcolstolb2 - 1].Value = ch;
            ++k;
        }
        string op = "";
        if (last.Count == 0)
        {
            cur = ch;
        }
        else
        {
            op = last.First();
            last.Pop();
        }
        if (op == con)
        {
            if (cur == ch && cur == '1')
            {
                cur = '1';
            }
            else
            {
                cur = '0';
            }                           }

        if (op == disun)
        {
            if (cur == ch && cur == '0')
            {
                cur = '0';
            }
            else
            {
                cur = '1';
            }                           }
        if (op == impl)
        {
            if (cur == '1' && ch == '0')
            {
                cur = '0';
            }
            else cur = '1';
        }
        if (op == equal)
        {
            if (cur == ch)
            {
                cur = '1';
            }
            else cur = '0';
        }                       }                   }
//MessageBox.Show(cur.ToString());
int newcolstolb1 = colzminn + k;
```

```
                ans.Push(cur.ToString()[0]);
                dataGridView2.Rows[num].Cells[newcolstolb1 - 1].Value = cur;
                ++k;
            }
        }
        // MessageBox.Show(ans.First().ToString());
        ans.Pop();
    }
      int newcolstolb = dataGridView2.ColumnCount - 1;
    for (int i = 0; i < newcolstolb; i++)
    {
        dataGridView2.Columns[i].DefaultCellStyle.ForeColor = Color.Black;
    }
    if (comboBox1.Text == "Закон поглинання")
    {
        dataGridView2.Columns.Remove(dataGridView2.Columns[newcolstolb]);
        dataGridView2.ColumnCount--;
        dataGridView2.Columns[0].DefaultCellStyle.ForeColor = Color.Red;
        for (int i = 0; i < zn; i++)
        {
            data2[i] = dataGridView2.Rows[i].Cells[newcolstolb - 2].Value.ToString();
        }
        dataGridView3.ColumnCount = 2;
        dataGridView3.RowCount = zn;
        dataGridView3.Columns[0].HeaderText = textBox1.Text;
        dataGridView3.Columns[1].HeaderText = textBox2.Text;
        dataGridView3.Columns[0].DefaultCellStyle.ForeColor = Color.Blue;
        dataGridView3.Columns[1].DefaultCellStyle.ForeColor = Color.Red;
        for (int j = 0; j < zn; j++)
        {
            dataGridView3.Rows[j].Cells[0].Value = data1[j];
            dataGridView3.Rows[j].Cells[1].Value = data2[j];
        }           }
    else
    {
        if (comboBox1.Text == "Закон склеювання")
    {
        dataGridView2.Columns.Remove(dataGridView2.Columns[1]);
        dataGridView2.ColumnCount--;
        dataGridView2.Columns[0].DefaultCellStyle.ForeColor = Color.Red;
        for (int i = 0; i < zn; i++)
        {
            data2[i] = dataGridView2.Rows[i].Cells[dataGridView2.ColumnCount-
1].Value.ToString();
        }
        dataGridView3.ColumnCount = 2;
        dataGridView3.RowCount = zn;
        dataGridView3.Columns[0].HeaderText = textBox1.Text;
        dataGridView3.Columns[1].HeaderText = textBox2.Text;
        dataGridView3.Columns[0].DefaultCellStyle.ForeColor = Color.Blue;
        dataGridView3.Columns[1].DefaultCellStyle.ForeColor = Color.Red;
        for (int j = 0; j < zn; j++)
```

```
            {
                dataGridView3.Rows[j].Cells[0].Value = data1[j];
                dataGridView3.Rows[j].Cells[1].Value = data2[j];
            }
        }
        else
        {
            dataGridView2.Columns[newcolstolb].DefaultCellStyle.ForeColor = Color.Red;
            dataGridView2.Columns[newcolstolb].HeaderText = textBox2.Text;
            data2 = new string[zn];
            for (int i = 0; i < zn; i++)
            {
                data2[i] = dataGridView2.Rows[i].Cells[newcolstolb].Value.ToString();
            }
            dataGridView3.ColumnCount = 2;
            dataGridView3.RowCount = zn;
            dataGridView3.Columns[0].HeaderText = textBox1.Text;
            dataGridView3.Columns[1].HeaderText = textBox2.Text;
            dataGridView3.Columns[0].DefaultCellStyle.ForeColor = Color.Blue;
            dataGridView3.Columns[1].DefaultCellStyle.ForeColor = Color.Red;
            for (int j = 0; j < zn; j++)
            {
                dataGridView3.Rows[j].Cells[0].Value = data1[j];
                dataGridView3.Rows[j].Cells[1].Value = data2[j];
            }           }       }    }
    private void button1_Click(object sender, EventArgs e)
    {
        this.Hide();
    }
    private void Закони_FormClosing(object sender, FormClosingEventArgs e)
    {
        if (e.CloseReason == CloseReason.UserClosing)
        {
            e.Cancel = true;
            Hide();
        }       }    }}

using System;
using System.IO;
using System.Collections.Generic;
using System.Linq;

/// <summary>
/// Базовый класс для логических функций
/// </summary>
public abstract class LogicFunction
{
  //Значение символа "склееной" позиции '*'
  public const byte cStarSymb = 2;

  //Дизъюнкции или конъюнкции функции
  public readonly ICollection<byte[]> Terms = new LinkedList<byte[]>();
```

```
  //Вычисление значения функции
  public abstract bool Calculate(bool[] X);
  //Вычисление значения функции
  public abstract bool Calculate(char[] X);
  //Вычисление значения функции
  public abstract bool Calculate(byte[] X);
}

/// <summary>
/// Дизъюнктивная нормальная форма
/// </summary>
public class Dnf : LogicFunction
{
  public override bool Calculate(byte[] X)
  {
    bool bResult = false;
    foreach (byte[] term in Terms)
    {
      bool bTermVal = true;
      for (int i = 0; i < term.Length; i++)
      {
        if ((term[i] >= cStarSymb) || (term[i] == X[i])) continue;
        bTermVal = false;
        break;
      }
      //bResult |= bTermVal;
      if (bTermVal)
      {
        bResult = true;
        break;
      }
    }
    return bResult;
  }

  public override bool Calculate(char[] X)
  {
    bool bResult = false;
    foreach (byte[] term in Terms)
    {
      bool bTermVal = true;
      for (int i = 0; i < term.Length; i++)
      {
        if ((term[i] >= cStarSymb) || (term[i] == (byte)(X[i] == '0' ? 0 : 1))) continue;
        bTermVal = false;
        break;
      }
      //bResult |= bTermVal;
      if (bTermVal)
      {
        bResult = true;
        break;
```

```
      }
    }
    return bResult;
  }

  public override bool Calculate(bool[] X)
  {
    bool bResult = false;
    foreach (byte[] term in Terms)
    {
      bool bTermVal = true;
      for (int i = 0; i < term.Length; i++)
      {
        if ((term[i] >= cStarSymb) || ((term[i] != 0) == X[i])) continue;
        bTermVal = false;
        break;
      }
      //bResult |= bTermVal;
      if (bTermVal)
      {
        bResult = true;
        break;
      }
    }
    return bResult;
  }
}

/// <summary>
/// Минимизация логической функции методом Квайна\Мак-Класки
/// </summary>
public class Quine_McCluskey
{
  //Максимальное кол-во элементов, при котором используется Dictionary,
  //чтобы не произошло переполнение максимальной ёмкости контейнера
  private static readonly int DICT_MAX_ITEMS = 8000000;

  //Коллекция "склеенных" термов
  private readonly Dnf _result = new Dnf();
  public Dnf Result
  {
    get { return _result; }
  }

  //Запуск метода
  public void Start(IEnumerable<byte[]> TermsInput)
  {
    LogicFuncMinimize(TermsInput, _result.Terms);
  }

  //Запуск метода
  public void Start(IEnumerable<char[]> TermsInput)
```

```
{
  Start(TermsInput.Select(t => t.Select(p => (byte)(p == '0' ? 0 : 1)).ToArray()));
}

//Запуск метода
public void Start(IEnumerable<bool[]> TermsInput)
{
  Start(TermsInput.Select(t => t.Select(p => (byte)(p ? 1 : 0)).ToArray()));
}

//Запуск метода с чтением ДСНФ из файла
public void Start(string sFileName)
{
  Start(LoadDsnfFromFile(sFileName));
}

//Нахождение минимальной логической функции
private static void LogicFuncMinimize(
  IEnumerable<byte[]> InpTerms, ICollection<byte[]> OutTerms)
{
  LinkedList<TreeNodeEnd> OutTemp = new LinkedList<TreeNodeEnd>();
  if (InpTerms.First().Length < 40)
  {
    IDictionary<UInt64, TreeNodeEnd> X1Tree = (InpTerms.Count() < DICT_MAX_ITEMS ?
      (IDictionary<UInt64, TreeNodeEnd>)(new Dictionary<UInt64, TreeNodeEnd>()) :
      new SortedDictionary<UInt64, TreeNodeEnd>());
    DeleteDublicatingTerms(InpTerms, X1Tree);
    //Повтор до тех пор пока будут оставаться термы
    while (X1Tree.Count != 0)
    {
      IDictionary<UInt64, TreeNodeEnd> X2Tree = (X1Tree.Count < DICT_MAX_ITEMS ?
        (IDictionary<UInt64, TreeNodeEnd>)(new Dictionary<UInt64, TreeNodeEnd>()) :
        new SortedDictionary<UInt64, TreeNodeEnd>());
      Skleivanie(X1Tree, X2Tree, OutTemp);
      X1Tree.Clear();
      X1Tree = X2Tree;
      GC.Collect(); //Сборка мусора очень сильно влияет на время работы!!!
    }
  }
  else
  {
    TreeFuncTerm X1Tree = new TreeFuncTerm();
    DeleteDublicatingTerms(InpTerms, X1Tree);
    //Повтор до тех пор пока будут оставаться термы
    while (X1Tree.Count != 0)
    {
      TreeFuncTerm X2Tree = new TreeFuncTerm();
      Skleivanie(X1Tree, X2Tree, OutTemp);
      X1Tree.Clear();
      X1Tree = X2Tree;
      GC.Collect(); //Сборка мусора очень сильно влияет на время работы!!!
    }
```

```
  }
  ReduceRedundancyTerms(OutTemp, OutTerms);
}

//Процедура чтения ДСНФ из файла и запись в матрицу
private static ICollection<byte[]> LoadDsnfFromFile(string sFullFileName)
{
  ICollection<byte[]> DSNF = new LinkedList<byte[]>();
  //Чтение строк из файла
  using (StreamReader InFile = new StreamReader(sFullFileName))
  {
    string sLine = "";
    while ((sLine = InFile.ReadLine()) != null)
    {
      DSNF.Add(sLine.Select(p => (byte)(p == '0' ? 0 : 1)).ToArray());
    }
  }
  return DSNF;
}

//Удаление дубликатов термов из входного списка
//В выходной словарь добавляются только уникальные термы
private static void DeleteDublicatingTerms(IEnumerable<byte[]> InX1,
  IDictionary<UInt64, TreeNodeEnd> OutX2Tree)
{
  OutX2Tree.Clear();
  foreach (byte[] x1 in InX1)
  {
    UInt64 iCode = GetTermCode(x1);
    if (OutX2Tree.ContainsKey(iCode)) continue;
    OutX2Tree.Add(iCode, new TreeNodeEnd(x1, null, null));
  }
}

//Удаление дубликатов термов из входного списка
//В выходное дерево добавляются только уникальные термы
private static void DeleteDublicatingTerms(IEnumerable<byte[]> InX1,
  TreeFuncTerm OutX2Tree)
{
  OutX2Tree.Clear();
  foreach (byte[] x1 in InX1)
  {
    OutX2Tree.Add(x1, null, null);
  }
}

//Склеивание строк с одним различием
private static void Skleivanie(
  TreeFuncTerm X1Tree, TreeFuncTerm X2Tree,
  ICollection<TreeNodeEnd> OutResult)
{
  Dictionary<int, TreeNodeEnd> FindTerms = new Dictionary<int, TreeNodeEnd>();
```

```
  TreeNodeEnd x1 = X1Tree.Last;
  while (x1 != null)
  {
    bool bIsSkleiv = false;
    for (int iPos = 0; iPos < x1.Term.Length; iPos++)
    {
      byte cSymbSav = x1.Term[iPos];
      if (cSymbSav == LogicFunction.cStarSymb) continue;
      //Склеивание двух термов с одним различием
      x1.Term[iPos] = (byte)(1 - cSymbSav);
      TreeNodeEnd pSkleivNode = X1Tree.Contains(x1.Term);
      if (pSkleivNode != null)
      {
        bIsSkleiv = true;
        //Проверка, чтобы комбинации термов обрабатывались только по одному разу
        if (cSymbSav == 1)
        {
          x1.Term[iPos] = LogicFunction.cStarSymb;
          X2Tree.Add(x1.Term, x1, pSkleivNode);
        }
      }
      x1.Term[iPos] = cSymbSav;
    }
    //Добавление на выход тех термов, которые ни с кем не склеились
    if (!bIsSkleiv) OutResult.Add(x1);
    //Переход к следующему листу дерева
    x1 = x1.PrevNode;
  }
}

//Возвращает уникальный код для терма
private static UInt64 GetTermCode(byte[] pTerm)
{
  UInt64 iMultip = 1, iCode = 0;
  for (int i = 0; i < pTerm.Length; i++)
  {
    iCode += (iMultip * pTerm[i]);
    iMultip *= 3;
  }
  return iCode;
}

//Склеивание строк с одним различием
private static void Skleivanie(
  IDictionary<UInt64, TreeNodeEnd> X1Tree,
  IDictionary<UInt64, TreeNodeEnd> X2Tree,
  ICollection<TreeNodeEnd> OutResult)
{
  foreach (KeyValuePair<UInt64, TreeNodeEnd> x1 in X1Tree)
  {
    bool bIsSkleiv = false;
    UInt64 iMultip = 1;
```

```
      for (int iPos = 0; iPos < x1.Value.Term.Length; iPos++)
      {
       byte cSymbSav = x1.Value.Term[iPos];
       if (cSymbSav != LogicFunction.cStarSymb)
       {
        UInt64 iCode;
        if (cSymbSav == 0)
         iCode = x1.Key + iMultip;
        else //_if (cSymbSav == 1)
         iCode = x1.Key - iMultip;
        TreeNodeEnd pSkleivNode = null;
        X1Tree.TryGetValue(iCode, out pSkleivNode);
        if (pSkleivNode != null)
        {
         bIsSkleiv = true;
         //Проверка, чтобы комбинации термов обрабатывались только по одному разу
         if (cSymbSav == 1)
         {
          //Склеивание двух термов с одним различием
          iCode = x1.Key + iMultip;
          if (!X2Tree.ContainsKey(iCode))
          {
           x1.Value.Term[iPos] = LogicFunction.cStarSymb; //Метка склеивания
           X2Tree.Add(iCode, new TreeNodeEnd(x1.Value.Term, x1.Value, pSkleivNode));
           x1.Value.Term[iPos] = cSymbSav;
          }
         }
        }
       }
       iMultip *= 3;
      }
      //Добавление на выход тех термов, которые ни с кем не склеились
      if (!bIsSkleiv) OutResult.Add(x1.Value);
     }
    }

    //Отбрасывание избыточных терм с помощью алгоритма приближенного решения задачи о
покрытии
    private static void ReduceRedundancyTerms(
     IEnumerable<TreeNodeEnd> SkleivTerms, ICollection<byte[]> ResultTerms)
    {
     //Подготовка результирующего контейнера
     ResultTerms.Clear();
     //Контейнер для соответствия конечного терма к списку первичных термов, которые его
образовали
     IDictionary<TreeNodeEnd, HashSet<TreeNodeEnd>> Outputs2Inputs =
      new Dictionary<TreeNodeEnd, HashSet<TreeNodeEnd>>();
     //Контейнер для соответствия первичных входных терм к тем выходным, которые их
покрывают
     IDictionary<TreeNodeEnd, HashSet<TreeNodeEnd>> Inputs2Outputs =
      new Dictionary<TreeNodeEnd, HashSet<TreeNodeEnd>>();
     //Сбор статистики об покрытии выходными термами входных
```

```
    foreach (TreeNodeEnd outTerm in SkleivTerms)
    {
      //Контейнер входных термов, которые покрывает данный выходной терм term
      HashSet<TreeNodeEnd> InpTermsLst = new HashSet<TreeNodeEnd>();
      HashSet<TreeNodeEnd> ListNumbers = new HashSet<TreeNodeEnd>();
      ListNumbers.Add(outTerm);
      while (ListNumbers.Count > 0)
      {
        TreeNodeEnd pCurrNode = ListNumbers.First();
        ListNumbers.Remove(pCurrNode);
        if (pCurrNode.Parent1 != null && pCurrNode.Parent2 != null)
        {
          ListNumbers.Add(pCurrNode.Parent1);
          ListNumbers.Add(pCurrNode.Parent2);
        }
        else
        {
          InpTermsLst.Add(pCurrNode);
          if (!Inputs2Outputs.ContainsKey(pCurrNode))
          {
            Inputs2Outputs.Add(pCurrNode, new HashSet<TreeNodeEnd>());
          }
          Inputs2Outputs[pCurrNode].Add(outTerm);
        }
      }
      Outputs2Inputs.Add(outTerm, InpTermsLst);
    }
    //Сортировка словаря исходных термов по возрастанию кол-ва их покрытий выходными
    Inputs2Outputs = Inputs2Outputs.OrderBy(p => p.Value.Count).ToDictionary(p => p.Key, v =>
v.Value);
    //Перебор всех входных термов, отсортированных по кол-ву покрывавших их выходных
    while (Inputs2Outputs.Count > 0)
    {
      TreeNodeEnd outTerm = Inputs2Outputs.First().Value.OrderByDescending(q =>
Outputs2Inputs[q].Count()).First();
      ResultTerms.Add(outTerm.Term);
      foreach (TreeNodeEnd inpTerm in Outputs2Inputs[outTerm].ToArray())
      {
        foreach (TreeNodeEnd outTerm2Del in Inputs2Outputs[inpTerm])
        {
          Outputs2Inputs[outTerm2Del].Remove(inpTerm);
        }
        Inputs2Outputs.Remove(inpTerm);
      }
    }
  }
}

/// <summary>
/// Дерево термов
/// </summary>
public class TreeFuncTerm
```

```
{
 //Корень
 private readonly TreeNodeMiddle rootNode = new TreeNodeMiddle();
 //Ссылка на завершающий узел последовательности конечных листьев дерева
 private TreeNodeEnd _lastTreeNode = null;
 public TreeNodeEnd Last
 {
  get { return _lastTreeNode; }
 }
 //Возвращает количество термов в дереве
 private int _count = 0;
 public int Count
 {
  get { return _count; }
 }

 //Конструктор
 public TreeFuncTerm()
 {
  Clear();
 }

 //Очистить дерево
 public void Clear()
 {
  rootNode.Clear();
  _count = 0;
  _lastTreeNode = null;
 }

 //Добавление в дерево нового терма
 public void Add(byte[] term, TreeNodeEnd pParent1, TreeNodeEnd pParent2)
 {
  TreeNodeMiddle pCurrNode = rootNode;
  int iTermLength1 = term.Length - 1;
  for (int i = 0; i < iTermLength1; i++)
  {
   byte cSymb = term[i];
   TreeNodeBase item = pCurrNode.Childs[cSymb];
   if (item == null)
   {
    item = new TreeNodeMiddle();
    pCurrNode.Childs[cSymb] = item;
   }
   pCurrNode = (TreeNodeMiddle)item;
  }
  TreeNodeBase pNewNode = pCurrNode.Childs[term[iTermLength1]];
  if (pNewNode == null)
  {
   pNewNode = new TreeNodeEnd(term, pParent1, pParent2, _lastTreeNode);
   pCurrNode.Childs[term[iTermLength1]] = pNewNode;
   _lastTreeNode = (TreeNodeEnd)pNewNode;
```

```
      _count++;
    }
  }

  //Проверка нахождения последовательности в контейнере
  public TreeNodeEnd Contains(byte[] term)
  {
    TreeNodeBase pCurrNode = rootNode;
    foreach (byte cSymb in term)
    {
      pCurrNode = ((TreeNodeMiddle)pCurrNode).Childs[cSymb];
      if (pCurrNode == null) break;
    }
    return ((pCurrNode != null) && (pCurrNode is TreeNodeEnd) ?
      (TreeNodeEnd)pCurrNode : null);
  }
}

/// <summary>
/// Базовый интерфейс узла дерева термов
/// </summary>
public interface TreeNodeBase
{
  //Очистка выделенных ресурсов
  void Clear();
}

/// <summary>
/// Конечный узел дерева термов
/// </summary>
public class TreeNodeEnd : TreeNodeBase
{
  //Терм, сопоставленный узлу
  public readonly byte[] Term;
  //Ссылка на предыдущий конечный узел дерева текущего уровня
  //для создания одностороннего связанного списка
  public readonly TreeNodeEnd PrevNode;
  //Ссылка на родительский конечный узел дерева предыдущего уровня
  public readonly TreeNodeEnd Parent1;
  //Ссылка на родительский конечный узел дерева предыдущего уровня
  public readonly TreeNodeEnd Parent2;

  //Конструктор
  public TreeNodeEnd(byte[] pTermRef, TreeNodeEnd pParent1, TreeNodeEnd pParent2,
    TreeNodeEnd pPrevTreeNode = null)
  {
    pTermRef.CopyTo(Term = new byte[pTermRef.Length], 0);
    Parent1 = pParent1;
    Parent2 = pParent2;
    PrevNode = pPrevTreeNode;
  }
```

```
  //Очистка выделенных ресурсов - отсутствует
  public void Clear() { }
}

/// <summary>
/// Промежуточный узел дерева термов
/// </summary>
public class TreeNodeMiddle : TreeNodeBase
{
  //Дочерние узлы
  public readonly TreeNodeBase[] Childs = new TreeNodeBase[3];

  //Очистка выделенных ресурсов
  public void Clear()
  {
    for (int i = 0; i < 3; i++)
    {
      if (Childs[i] != null)
      {
        Childs[i].Clear();
        Childs[i] = null;
      }
    }
  }
}
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
Диплом_ doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Диплом_.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Program.rar	Архів. Містить коди програми і откомпільовану програму
Презентація	
Презентація_.ppt	Презентація кваліфікаційної роботи