Міністерство освіти і науки України
Рівненський державний гуманітарний університет
Кафедра інформаційних технологій та моделювання

Кваліфікаційна робота

за освітнім ступенем «магістр»

на тему:

ІНТЕРПРЕТАЦІЯ АРИФМЕТИЧНИХ ОПЕРАЦІЙ У ДВІЙКОВИХ КОДАХ

Виконав:
здобувач ІІ курсу
групи М-КН-21
спеціальності 122 Комп’ютерні науки
Артем-Даниїл Олександрович
Казимірський

Науковий керівник:
кандидат технічних наук, доцент
Степанія Михайлівна Бабич

Рівне – 2024

ЗМІСТ

ВСТУП

Одним з найбільших сучасних досягнень людства є винахід комп'ютерів і використання їх практично у всіх сферах людської діяльності. Комп'ютер або електронно-обчислювальна машина (ЕОМ) використовується у всіх сферах виробництва, у різноманітних наукових дослідженнях, у сфері економіки та банківської справи, у медицині, у космічній області, у педагогіці і, навіть, у мистецтві. ЕОМ вплинула на значне підвищення ефективності розумової праці людини в обчислювальних процесах, у сферах досліджень і моделювань, забезпечила автоматизацію виробничих технологічних операцій, допомогла вирішити проблеми управління і контролю виробничої діяльності практично в усіх галузях, а також стала незамінюваною щодо накопичення та обробки інформації. Взагалі поняття інформації має як філософське, так і суто технічне значення. Ми розуміємо під інформацією сукупність даних про об'єкт, його властивості, стан, зміни протікання процесів у ньому. Об'єктом інформації може бути будь-що: виробництво, природні процеси, суспільство і т. д. [15].

Інформацію можна передавати різноманітними сигналами або повідомленнями: звуковими, словесними, зображальними, світловими тощо. Тобто інформацію можна передавати, приймати, трансформувати. Такі властивості інформації сприяють побудові різноманітних кібернетичних систем, серед яких важливими є обчислювальні системи [12].

Винаходу електронно-обчислювальної техніки передувала довга тисячолітня історія розвитку наукової думки. Ще в давніх культурах людства виникала необхідність вести обчислювальні операції: перерахунок кількості предметів у обмінах товарами, вимірювання площі, кількісне визначення маси, визначення часу, тощо. Повсякденні обрахунки вимагали виникнення певних пристроїв для полегшення вирішення таких завдань. Першим таким пристроєм стали лічильні палички, які дотепер використовуються в початковій школі для навчання дітей рахунку. Перші обчислювальні пристрої були немеханічними: різновиди рахівниць у стародавньому Вавилоні і стародавньому Китаї. В епоху

середньовіччя з розвитком математики з’являються механічні пристрої на основі зубчастих передач, хоча перший такий пристрій був винайдений ще в античну епоху в Греції і використовувався для астрономічних обчислень, але дана технологія була призабута в наступні 1500 років. У 17 столітті створюють логарифмічну лінійку і перші цифрові обчислювальні пристрої. Німецький філософ і математик Готфрід Вільгельм Лейбніц створив механічний калькулятор, який на основі двійкової системи міг виконувати математичні дії: додавання, віднімання, множення, ділення. Лейбніц вважається засновником науки інформатики і двійкова система числення сьогодні є базовою в комп’ютерах. Прообразом майбутніх обчислювальних машин став проєкт великого арифмометра з програмним управлінням, який розробив у 1833 році англійський вчений Чарльз Беббіт. Потрібно зазначити, що його машина, як і багато інших розробок до 1940 х років була заснована на десятковій системі числення, яку було складніше реалізувати. До арифметично-обчислювальних пристроїв було додані умовні команди і збільшена пам'ять, що дозволило автоматизувати багато завдань як арифметичних так і текстових [27, 28].

Після 1940 року електронно-обчислювальна техніка (ЕОТ) почала стрімко розвиватись, змінюючись кожне десятиліття. За ступенем програмних і апаратних засобів розрізняють чотири стадії розвитку ЕОТ– чотири покоління: від використання електровакуумних ламп, транзисторів, інтегральних схем до мікропроцесорів [21].

Процесор – це електронна схема, яка виконує обробку даних. Наприклад: введення або виведення даних, виконання арифметичних або логічних дій з даними, тощо [24]. Процесор має внутрішні спільно працюючі пристрої. Операції з цілими числами виконує арифметико-логічний пристрій процесора. Арифметичні дії додавання, віднімання, множення, ділення, а також логічні операції виконуються за допомогою цього пристрою. Також процесор може містити декілька математичних співпроцесорів, що прискорює продуктивність обчислень [22, 23].

Актуальність дослідження. Початкові дані і завдання для комп'ютерів задаються зазвичай в десятковій системі числення і потребують в ній же відповідей. Але ЕОТ працює, як правило, в двійковій системі числення або в іншій, відмінній від десяткової. Тому наявність програми інтерпретації арифметичних операцій, переведення з однієї системи числення в іншу є необхідністю. Зокрема, у даному дослідженні розробляється програма інтерпретації арифметичних дій у двійкових кодах, які є найбільш використовуваними в ЕОТ. Існуючі програми інтерпретації арифметичних дій задовольняють лише частково, оскільки інтерпретують числа лише в одному напрямку. Але в роботі з ЕОМ є також необхідним інтерпретувати числа зворотно, а також переводити в машинні коди. Розроблена програма буде особливо помічною для студентів, які вивчають навчальну дисципліну «Комп'ютерна математика», а також для спеціалістів-програмістів для розв'язання задач в ЕОМ.

Мета дослідження. Розробка програмного застосунку для інтерпретації арифметичних операцій над цілими числами з використанням машинних кодів (прямого, зворотного, доповняльного) із коректним відображенням результатів у десятковій системі числення.

Завдання дослідження. Для досягнення мети поставлені до вирішення наступні завдання**:**

- дослідити системи числення, які використовуються в комп'ютерній техніці;
- проаналізувати алгоритми переведення цілих чисел з десяткової системи числення у двійкову систему і навпаки;
- дослідити алгоритми переведення цілих чисел з двійкової системи числення в машинні коди: прямий, зворотний і доповняльний, – і навпаки.
- для розробки застосунку вибрати одну з мов програмування високого рівня із середовищем візуальної розробки;
- розробити демонстраційну програму інтерпретації арифметичних операцій у двійкових кодах процесора.

Об’єкт дослідження. Процеси виконання арифметичних операцій у машинних кодах з використанням двійкової системи числення.

Предмет дослідження. Методи і алгоритми перетворення цілих чисел з десяткової системи числення у двійкову та в машинні коди, виконання арифметичних операцій над ними, а також зворотного перетворення результатів у десяткову систему числення.

Методи дослідження.

Теоретичні методи:

аналіз літератури та джерел – дослідження основ алгоритмів обчислень у двійкових кодах;

математичне моделювання – формалізація процесу виконання арифметичних операцій над числами у різних машинних кодах;

системний підхід – вивчення взаємодії етапів перетворення даних (десяткова система → двійкова система → машинний код → арифметичні дії → зворотне перетворення).

Практичні методи:

алгоритмізація – розробка алгоритмів перетворення цілих чисел і виконання арифметичних операцій над їх двійковими кодами;

програмування – реалізація алгоритмів у вигляді застосунку;

тестування програмного забезпечення – перевірка коректності роботи застосунку на тестових наборах даних;

експериментальний метод – порівняння результатів роботи програми з теоретично очікуваними.

Для розробки інформаційної системи інтерпретація арифметичних операцій у двійкових кодах процесора застосовано Visual Studio 2019. Сама розробка програмного продукту виконана на мові програмування C Sharp з застосуванням вікон візуалізації Windows Forms.

Практичне значення дослідження. Демонстраційна система інтерпретації арифметичних операцій може використовуватись викладачами і студентами на практичних заняттях з предмету «Комп’ютерна математика», а

також використовуватись спеціалістами для розв’язання різноманітних завдань на ЕОТ, які потребують переведення з десяткової системи числення в двійкову і навпаки, переведення в машинні коди, виконання арифметичних дій у двійковій системі числення, а саме додавання, віднімання, множення та ділення.

Апробація та впровадження результатів дослідження. Основні теоретичні та практичні результати дослідження доповідалися та обговорювалися на VI Всеукраїнської конференції молодих учених «Актуальні питання розвитку інформаційних технологій» (АПРІТ-2024) (м. Дніпро, 20 листопада 2024 р.), організована факультетом інформаційних технологій ДВНЗ «Приазовського державного технічного університету» [7].

Структура роботи. Робота складається з вступу, чотирьох розділів, висновків, списку літератури. Робота написана на 64 сторінках, містить 1 таблицю, 20 рисунків. У списку використаних джерел 38 найменувань.

РОЗДІЛ 1
АНАЛІЗ ПРОБЛЕМИ ДОСЛІДЖЕННЯ

1.1. Загальні відомості про системи числення

Система числення — це група правил, заходів і знаків, за допомогою яких можна кодувати будь-яке невід'ємне число, тобто найменувати і позначати числа. Кількість знаків цифрової абетки є кінцевою. На основі її складаються числа в певній системі числення. До систем числення висуваються вимоги, головною є вимога однозначного кодування невід'ємних чисел 0, 1,… з деякою їх кінцевою множиною — діапазону P за скінченне число дій, що надає можливість виконання арифметичних та логічних операцій. Вдалий вибір системи числення впливає на ефективне вирішення завдань і практичного їх використання [18, 19].

Існують непозиційні та позиційні системи числення.

Непозиційні системи числення виникли історично раніше. У цих системах цифра, що позначає величину, не залежить від позиції її у числі. Але можуть буди певні обмеження на позиції цифр, наприклад згрупування за значенням, розташування по спаданню і т. п., але це не принципова вимога розуміння запису чисел у таких системах. Можна навести наступні приклади непозиційних систем: числова система залишків, де представлення числа засноване на китайській теоремі про залишки, а операції з числами виконуються за допомогою модульної арифметики; римська система числення, у якій цифри позначаються латинськими літерами. А саме: 1 – I, 5 – V, 10 – X, 50 – L, 100 – C, 500 – D, 1000 – M. Натуральні числа в даній системі записуються за принципом повторення цих знаків. Якщо більша цифра стоїть перед меншою, то використовується принцип додавання, вони додаються, і навпаки, якщо менша перед більшою – то принцип віднімання, менша віднімається. Потрібно сказати, що римська система числення деякою мірою зберігається дотепер: нумерація століть, місяців у датах, на циферблатах деяких годинників, у позначення розділів книг і томів, у теорії музики та ін [18, 36].

Проте в обчислювальній техніці непозиційні системи використовуються дуже обмежено, оскільки потребують складні та громіздкі алгоритми подання чисел, що ускладнює виконання арифметичних операцій.

У позиційних системах значення кожної цифри залежить від місця (позиції) в послідовності цифр при записі числа. Позиційна система може повторно використовувати одні й ті ж символи, які набувають різного значення від їх позиції в послідовності. Як правило, значення кожної позиції кратне деякому натуральному числу: b, де b більше 1, яке називається основою системи числення [12, 36].

Існують також змішані системи числення. Такі системи є узагальненням системи числення з основою b. Але ці системи часто розглядаються як різновид позиційних систем числення.

У сучасних цифрових ЕОМ використовуються головним чином позиційні системи числення, тому що в них простіше реалізуються правила, ніж у непозиційних системах.

Історично позиційні системи числення виникли в декількох людських цивілізаціях незалежно одна від другої: у Вавилоні, у Стародавньому Китаї, у цивілізації ацтеків. Позиційною системою числення є широко розповсюджена звичайна десяткова система числення. Виникнення даної системи числення пояснюється природно, оскільки 10 пальців двох рук було першим доступним зручним пристроєм для рахунків у стародавніх людей [18, 19].

Вважають, що десяткова система запису виникла в Індії приблизно в VI ст. до н. е. Індійські математики до VIII ст. н. е. вдосконалили цю систему. Вона була запозичена арабами і протягом наступних століть у X – XIII ст. через арабських завойовників, торговців, вчених поступово з’явилась в Європі. Це була десяткова система числення з основою десять, у якій можна записувати будь-яке число за допомогою десяти унікальних знаків. Позиції цих знаків починаються справа і зростають у напрямку ліворуч. Ключовим досягненням цієї системи стало винайдення числа 0. Потрібно зазначити, що також число 0 було незалежно винайдено в цивілізації майя [18, 36].

Великою заслугою в популяризації десяткової системи в Європі були арабські вчені Абу-Рейхам-Мухаммед ібн-Ахмед аль-Біруні (973 – 1048) та Абул-Фатх Омар ібн-Ібрахім Хайям (1048 – 1131).

Десяткова нумерація на основі алфавітів кирилиці та глаголиці була поширена і в Київській Русі в IX – XII ст.. [18, 36].

Позиційні системи поділяють на два великих класи: однорідні, з рівною довжиною числа в основі; та неоднорідні, з рівною та нерівною довжиною чисел і більш складною основою ніж натуральні числа [12].

Перечислимо основні існуючі види однорідних систем числення:

- двійкова – використовується в програмуванні, в інформатиці, у дискретній математиці, є предметом нашого дослідження і більш детально представимо її нижче;
- трійкова – використовується в трійковому комп'ютері;
- п'ятіркова – використовувалась німецькими військовими у криптографії в період Першої світової війни, зустрічається в деяких народів Африки, Океанії, Австралії, Південної та Північної Америки, частково збереглись сліди в Азії та Європі як п'ятизначний усний рахунок;
- вісімкова – застосовується у програмуванні;
- десяткова – найбільш поширена в сучасній цивілізації;
- дванадцяткова – була поширена в давнину, але частково зустрічається дотепер, наприклад: дюжина, 12 місяців, 12 балів, 12 годин на циферблаті годинника, сервіз на 12 персон і т. п.
- шістнадцяткова – широко застосовується в сучасному програмуванні для більш компактного запису, у кодуванні шрифтів;
- двадцяткова – використовувалась у системах майя і ацтеків, може поєднуватись з п'ятірковою системою числення, дотепер використовується в деяких мовах індіанців, залишились сліди в кавказьких, азійських і навіть у деяких граматичних конструкціях європейських мов;

– шестидесяткова – виникла у 3 тисячолітті до н. е. у шумерів, використовувалась у Вавилонії, частково залишилась дотепер у вимірі географічних координат, кутів, часу (60 хвилин, 60 секунд) [18, 19].

В інформаційних технологіях застосовуються двійкова, вісімкова, десяткова, шістнадцяткова системи числення.

Зазвичай позиційні системи числення мають найменування, яке співпадає з кількістю цифр основи – Р, для запису числа використовується обмежена кількість знаків – цифр.

Основа позиційної системи числення – число, яке виявляє, у скільки разів одиниця старшого розряду більша одиниці сусіднього молодшого розряду.

Абетка цифр позиційної системи числення характеризує значення цифр, які використовуються для зображення чисел у даній системі числення. Звичайно цифри обираються таким чином, щоб вони складали відрізок натурального ряду чисел, включаючи число “0”. У цьому випадку максимальне значення цифри, яка використовується у системі числення, дорівнює Р – 1. У десятковій системі числення абетка характеризується значеннями цифр 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, тобто використовується десять цифр від 0 до 9. Наприклад, число 123 визначається як $1 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0$. З цього прикладу ми бачимо, що позиція кожної цифри має свій ваговий коефіцієнт, і перша позначає кількість сотень, а остання – кількість одиниць.

Двійкова або бінарна система числення – це одна з позиційних систем числення, основа якої Р = 2, тобто старший розряд числа у два рази більший від сусіднього молодшого розряду. У цій системі використовуються тільки дві цифри: 0 і 1. Тому будь-яке число у двійковій системі числення записується як комбінація цифр 0 і 1. Основа двійкової системи числення записується як 10, тобто $2_{10} = 10_2$ і зчитується: “один, нуль” [12, 19].

Дана система не відрізняється від інших позиційних систем з будь-якою іншою основою. Бінарна система числення набула широкого розповсюдження в сферах обчислювальної техніки завдяки простоті запису чисел, оскільки її алфавіт має лише два символи. Вибір системи числення для подання має

величезне практичне значення для обчислювальної техніки і пам’яті комп’ютера. При виборі системи числення враховують: надійність подання чисел; економічність, тобто має бути мінімальна кількість елементів для подання чисел з деякого діапазону. Двійкова система числення дозволяє показати, або закодувати будь-яке натуральне число як послідовність нулів і одиниць. Такий принцип кодування називається двійковим .

Програмістів двійкова система кодування інформації зацікавила легкістю реалізації, вона дозволяє подати будь-яку інформацію в одному і тому самому двійковому вигляді, який легко піддається обробці на ЕОМ [35]. Комп’ютеризація сучасної цивілізації була здійснена власне завдяки двійковій системі. Уся обчислювальна техніка посилає імпульси, які перетворюються в нулі або одиниці: Значення 1 – є імпульс; значення 0 – нема імпульсу. Процесор комп’ютера зчитує бінарний код і перетворює його в текст, звук або зображення і навпаки: вислані повідомлення, фотографії, аудіо- чи відео- зображення з комп’ютера або телефону перетворюються у двійковий код [20].

Двійкова система певною мірою виникала в історичному минулому у деяких цивілізаціях людства. У трактаті «Книга Змін» Стародавнього Китаю (II тисячоліття до н. е.), присвяченому ворожінню, використовуються 8 триграм і 64 гексаграм, які можна співставити з 3-бітними і 6-бітними числами. Гексаграми розташовані в певному порядку аналогічно значенню відповідних двійкових цифр від 0 до 63. У XI ст. китайський вчений і філософ Шао Юн розробив метод їх отримання, але ймовірно він не розумів правила двійкової арифметики [18, 19].

Індійський математик Пінгала в 200 р. до н. е. у своїй праці «Чандрас-сутра» розробив математичні основи для опису поезії, де використовує двійкову систему числення [27].

У I – II тисячоліттях н. е. в Перу, Болівії (Центральні Анди) існувала вузликова писемність інків – кіпу. Вона складалась як з числових записів десяткової системи, так і не числових записів у двійковій системі числення.

В Африці існує традиційне ворожіння Іфа, у якому використовуються комбінації двійкового числення [18, 19, 36].

Сучасна двійкова система була повністю описана в XVII ст. німецьким математиком Лейбніцем у праці «Explication de L'Arithmtique Binaire». У його системі числення були використані цифри 0 і 1 [28].

У 1854 р. англійський математик Джордж Буль опублікував працю, де описує алгебраїчні системи відповідно логіці. Його система сьогодні відома як алгебра логіки або булева алгебра, що зіграла значну роль у сучасних електронних цифрових схемах [27, 28].

У 1937 році вченим Клодом Шенноном у Массачусетсі була захищена дисертація «Символічний аналіз релейних і перемикальних схем», у якій двійкову арифметику і булеву алгебру було примірено до електронних реле і перемикачів. На даній дисертації фактично заснована вся сучасна обчислювальна техніка [27, 28].

Десяткова система застосовує десять символів, тобто 0 (нуль) та 1 (одиницю), 2 (два), 3 (три), 4 (чотири), 5 (п'ять), 6 (шість), 7 (сім), 8 (вісім), і 9 (дев'ять), щоб здійснити записування чисел. Вона дає можливість візуалізувати або закодувати будь-яке ціле число в якості послідовності чисел від нуля до дев'яти. Даний принцип кодування має назву десятковий.

Основою даної позиційної системи числення є P = 10, а саме старший розряд числа у десять разів більше сусіднього молодшого розряду. Для зображення цілих чисел від 1 до 9999 у десятковій системі достатньо трьох розрядів, а саме трьох елементів [12].

Економічність системи

Щоб візуалізувати цілі числі від 0 до 1000 у десятковій системі досить і трьох розрядів, а саме трьох елементів. І тому, що будь-який визначений об'єкт має можливість бути у десяти станах, тоді основне число станів буде 30, у двійковій системі числення 989 буде 11 1101 1101, потрібне число станів буде 20 У даному роздумі економнішою буде позиційна система з назвою трійкова.

Трійкова система числення частково виникла ще в стародавніх шумерів і стосувалась системи терезів, мір і грошей. Але як позиційна симетрична система

числення повною мірою була запропонована італійським математиком Фібоначчі (1170 р.—1250 р.) для вирішення задачі «про гирі». Ще з давніх часів використовували при зважуванні на терезах гирю, докладаючи її на чашу з товаром для симетричності ваги. Фібоначчі у своїй системі об’єднав властивості одиниць, що дорівнюють трьом [18].

Але, щоб здійснити запис цілих чисел від 1 до 999999999 у десятковій системі потрібно 90 станів, у двійковій – 60, у трійковій необхідно 57. З тієї причини система числення з основою 3 не отримала розповсюдження через труднощі реалізації у фізичному вигляді [37].

Через описане вище, двійкова система буде близькою до оптимальної за економічністю, ще, окрім того, у цій системі є дуже простими таблички додавання і множення.

Приклад. Оскільки $2^3=8$, а $2^4=16$, то окремих три двійкових розряди візуалізації числа створюють один вісімковий, а окремі чотири двійкових розряди – один шістнадцятковий. Тоді, щоб скоротити записування адрес і вмісту оперативної пам'яті комп'ютера застосовують шістнадцяткову та вісімкову системи числення [12].

1.2. Переведення чисел з однієї позиційної системи числення в іншу

У цифровій обчислювальній техніці використовується машинна арифметика, і, як правило, досить часто виникає необхідність переводити числа з однієї системи в іншу, оскільки вихідні дані можуть бути записані в іншій системі, ніж запрограмована в комп’ютері або існує необхідність отримання результату у відмінній системі числення. Для цього існують певні алгоритми, які залежать власне від самої системи числення. Одним з існуючих методів переведення є *алгоритм безпосередньої заміни*. Операції переведення можуть виконуватись з додатними та від’ємними цілими і дробовими числами. Перше правило переведення числових символів між системами числення – переведення в десяткову систему. Для переведення числа з позиційної системи з основою р у

десяткову систему необхідно записати його у вигляді суми степенів р і виконати обчислення в десятковій системі числення. Тобто, для переведення двійкового числа в десяткове необхідно його записати у вигляді многочлена, що складається з числа А і відповідного степеня числа 2, та обчислити за правилами десяткової арифметики [12, 13, 34].

Конвертація з двійкової системи числення у десяткову:

$X_2 = A_n * 2^{n-1} + A_{n-1} * 2^{n-2} + A_n * 2^{n-3} + A_2 * 2^1 + A_1 * 2^0$

При переведенні зручно користуватися таблицею степенів двійки:

Таблиця 1.1. Степені числа 2

Степінь	0	1	2	3	4	5	6	7	8	9	10
2^n	1	2	4	8	16	32	64	128	256	512	1024

Приклад. Число 1001010101_2 необхідно перевести в десяткову систему числення.

$1001010101_2 = 1 * 2^9 + 1 * 2^8 + 1 * 2^7 + 0 * 2^6 + 0 * 2^5 + 0 * 2^4 + 1 * 2^3 + 1 * 2^2 + + 1 * 2^1 + 1 * 2^0 = 597_{10}$.

Приклад. Число 1001000_2 необхідно перевести в десяткову систему числення.

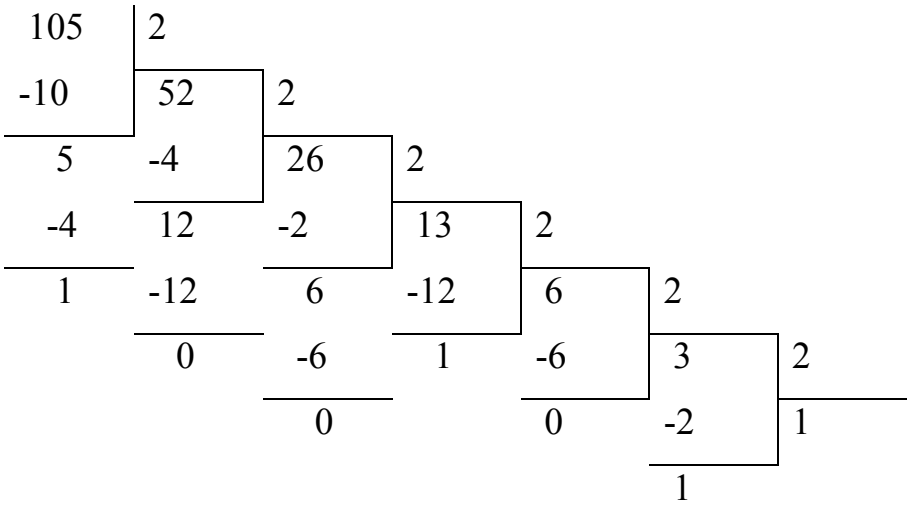
$1001000_2 = 1 * 2^9 + 1 * 2^8 + 1 * 2^7 + 0 * 2^6 + 0 * 2^5 + 0 * 2^4 + 1 * 2^3 = 155_{10}$.

Приклад. Число 101011000_2 необхідно перевести в десяткову систему числення.

$101011000_2 = 1 * 2^9 + 1 * 2^8 + 1 * 2^7 + 0 * 2^6 + 0 * 2^5 + 0 * 2^4 + 1 * 2^3 + 1 * 2^2 + 1 * 2^1 = 428_{10}$

Навпаки, щоб перевести десяткове число у двійкову систему, для цього потрібно дане число послідовно ділити на 2 до тих пір, поки не залишиться залишок, менший або рівний 1. Число у двійковій системі виконує запис як перелік кінцевого результату ділення і залишків від ділення у зворотному порядку.

Приклад. Конвертація десяткового числа $X_{10} = 105$ у двійкову систему числення (P = 2).



Відповідь: $105_{10} = 1101001_2$

Для правильних дробів

Перетворення правильного дробу з позиційної системи числення з основою **p** в десяткову систему передбачає його представлення у вигляді суми степенів основи **p** з подальшим виконанням відповідних обчислень у десятковій системі.

Для переведення правильного дробу з десяткової системи числення до системи з основою p слід поетапно множити початковий дріб і дробові частини проміжних добутків на основу p. Процес триває до тих пір, поки дробова частина не стане рівною нулю або не буде досягнуто необхідного рівня точності. Точність обчислення визначається значенням $\mathbf{p}^{-m}$, де m — кількість виконаних множень. У новій системі числення правильний дріб формується з цілих частин проміжних добутків, починаючи з першого.

Для мішаних дробів

Для перетворення мішаного дробу з однієї системи числення з основою p в іншу з основою q, необхідно виконати два окремих етапи: спочатку перевести цілу частину дробу за правилами для цілих чисел, а потім обчислити його дробову частину за алгоритмом переводу правильних дробів. Після цього результати обох переведень, розділені комою, об'єднуються в один мішаний дріб у новій системі числення.

1.3. Арифметичні операції у 2-й СЧ

Будь-які арифметичні операції у двійковій системі числення виконується згідно з визначеними правилами використання арифметичних операцій у загальноприйнятій системі десяткового числення, але при цьому виконується застосування таблиці додавання, віднімання, множення і ділення, які утворені для системи двійкового числення.

Для створення таблиці додавання в кожній системі числення з основою P користуються таким правилом: у сумі дві цифри a_i і a_j

$a_i + a_j = a_k$, якщо $a_i + a_j < P$;

$1a_k$, якщо $a_i + a_j > P$ (є одиниця перенесення в наступний старший розряд), де $a_k = a_i + a_j - P$ є цифрою даної системи числення.

І потрібно пам’ятати, що у двійковій системі числення до будь-якого старшого розряду входять дві одиниці сусіднього молодшого розряду.

Таблиця двійкового додавання

$0 + 0 = 0$

$0 + 1 = 1$

$1 + 0 = 1$

$1 + 1 = 10$

Таблиця двійкового віднімання

0 - 0 = 0

1 - 0 = 1

1 – 1 = 0

10 – 1 = 1

Таблиця двійкового множення

0 * 0 = 0

0 * 1 = 1

1 * 0 = 1

1 * 1 = 1

Додавання чисел у двійковому численні можна виконувати стовпцем, стартуючи з молодшого розряду. У будь-якому розряді згідно з нормами і правилами, означеними таблицею двійкового додавання, виконується процес додавання двох чисел доданків і одиниці перенесення з сусіднього молодшого розряду, коли вона є.

У підсумку виходить цифра відповідного розряду суми, і, ймовірно, одиниця пересування в сусідній старший розряд.

Приклад. Додати число X_2 =1001000 і число Y_2 = 11010 у двійковій системі числення.

Розв’язання:

одиниці перенесення 11 - - -

X = 1001000
+
Y = 0011010

X + Y = 1100010

Відповідь: X + Y = 1100010_2.

У процесі віднімання чисел у двійковій системі числення виникає необхідність "позичити" одиницю з сусіднього старшого розряду, якщо цифра у відповідному розряді від’ємника перевищує цифру у тому ж розряді

зменшуваного. Позичена одиниця еквівалентна двом одиницям у поточному розряді.

Приклад. Провести операцію віднімання числа $Y_2 = 10011011$ від числа $X_2 = 10101100$ у двійковій системі числення.

Розв'язання:

одиниці позички 1 1

X = 10101100
−
Y = 10011011

X − Y = 00110111

Відповідь: $X - Y = 00110111_2$.

Множення двійкових чисел виконується та само, як при множенні десяткових, спершу одержують часткові добутки, а потім визначають їхню суму. На підставі таблиці двійкового множення будь-який частковий добуток рівний нулю, коли в даному розряді множника стоїть нуль, або одне множене, зсунуте на відповідну кількість розрядів вліво у випадку одиниці в розряді множника.

Приклад. Помножити число $X=1010_2$ на число $Y=1101_2$.

Розв'язання:

1010
×
1101

1010
0000
1010
1010

10000010

Відповідь: $X * Y = 10000010_2$.

Отже, операція знаходження добутку двійкових багаторозрядних чисел зводиться до операцій зсуву і додавання. Кількість розрядів дробової частини у

добутку виділяється комою, що дорівнює сумі розрядів дробових частин співмножників [12].

Ділення двійкових чисел здійснюється за тими ж правилами, що й для десяткових. При цьому використовуються таблиці двійкового множення і віднімання.

Приклад. Поділити число $X=10000010_2$ на число $Y=1010_2$.

```
 10000010 | 1010
-  1010   | 1101
  ------
    1100
-   1010
  ------
     1010
-    1010
  ------
       0
```

Відповідь: $X : Y = 1101_2$.

Для виконання операції множення і ділення в комп'ютері використовуються спеціальні алгоритми.

Завдячуючи простій структурі правил двійкової арифметики, використання в комп'ютерах двійкової системи дозволяє суттєво підвищити простоту схеми арифметичних пристроїв.

1.4. Подання даних у комп'ютерах

Машинне слово — це одиниця даних, яка може бути записана або зчитана з оперативної пам'яті комп'ютера за одне звернення.

У більшості сучасних комп'ютерів використовуються такі типи машинних слів:

байт: складається з 8 бітів;

слово: дорівнює 2 байтам або 16 бітам;

подвійне слово: містить 4 байти або 32 біти;

зчетверене слово: включає 8 байтів або 64 біти.

Розрядна сітка комп’ютера — це кількість бітів, необхідних для зберігання одного машинного слова в оперативній пам’яті. Вона є фіксованою для кожного типу комп’ютера та визначає його ключову технічну характеристику.

Машинне подання чисел описує спосіб розташування бітів числа у розрядній сітці. У комп’ютерах використовують два основні формати подання чисел:

з фіксованою комою. У цьому форматі положення коми в розрядній сітці задається апаратно:

– якщо кома фіксується перед старшим бітом, число представляється як правильний дріб;

– якщо кома знаходиться після молодшого біта, число інтерпретується як ціле;

з плаваючою комою. У цьому форматі число записується у вигляді добутку мантиси та степеня основи системи числення:

$X = \pm m\, q^{\pm p}$, де

m — мантиса (зазвичай правильний дріб),

q — основа системи числення,

p — порядок числа.

У разі, якщо результат обчислень перевищує максимально допустиме значення машинного слова, відбувається переповнення розрядної сітки.

Переваги та застосування форматів: формат із фіксованою комою зазвичай використовується для представлення цілих чисел, тоді як формат із плаваючою комою — для дійсних чисел. Сучасні цифрові комп’ютери часто підтримують обидва формати, що забезпечує гнучкість у виконанні різних обчислень.

1.5. Машинні коди двійкових чисел

1.5.1. Прямий код

Прямий код представляє число у вигляді його абсолютного значення із зазначенням знака. У цьому форматі мантиса числа збігається з дробовою частиною двійкового подання. Знак числа позначається: 0 для додатного числа та 1 для від'ємного.

Приклад. Числа $X_{10} = 45$, $Y_{10} = -73$ записати в прямому коді.

X_2: +101101; $[X]_п$: 0 0101101.

Y_2: -1001001; $[Y]_п$: 1 1001001.

При додаванні чисел у прямому коді з однаковими знаками, мантиси додаються, а результату присвоюється знак доданків [12].

Приклад. Додати числа $X_{10} = 45$, $Y_{10} = 73$ в прямому коді.

```
[X]п:   0  0101101
        +
[Y]п:   0  1001001
        ---------------
         0  1110110
```

Прямий код широко використовується в комп'ютерах для запису та зберігання чисел у пам'яті, введення даних, виведення результатів, а також під час операцій множення і ділення, де працюють з абсолютними значеннями чисел. Операцію віднімання часто замінюють додаванням чисел у зворотному чи доповняльному кодах [12].

1.5.2. Обернений (зворотний, інверсний) код

Зворотний код для додатного числа ідентичний його прямому коду. Для від'ємного числа він формується шляхом інвертування (заміни на протилежні) всіх бітів абсолютного значення числа, при цьому знаковий розряд встановлюється в 1 [6].

Приклад. Числа $X_{10} = 45$, $Y_{10} = -73$ записати у зворотному коді.

Розв'язання:

$[X]_п$: 0 0101101; $[X]_з$: 0 0101101.

$[Y]_п$: 1 1001001; $[Y]_з$: 1 0110110.

При алгебраїчному додаванні чисел у зворотному коді виконується додавання їхніх бітів, включно з розрядом знака, який обробляється як звичайний цифровий розряд. Якщо виникає перенесення одиниці за межі знакового розряду, вона циклічно додається до молодшого розряду.

Приклад. Додати числа $X_{10} = 45$, $Y_{10} = -73$ у зворотному коді.

Розв'язання:

```
[X]з:   0  0101101
        +
[Y]з:   1  0110110
        ---------------
         1  1100011
```

1.5.3. Доповняльний (додатковий) код

Доповняльний код для додатного числа також збігається з його прямим кодом. Для від'ємного числа спершу створюється зворотний код, після чого до молодшого розряду додається одиниця [12].

Приклад. Числа $X_{10} = 45$, $Y_{10} = -73$ записати в доповняльному коді.

Розв'язання:

$[X]_п$: 0 0101101; $[X]_з$: 0 0101101; $[X]_д$: 0 0101101.

$[Y]_п$: 1 1001001; $[Y]_з$: 1 0110110; $[Y]_д$: 1 0110111.

Алгебраїчне додавання чисел у доповняльному коді виконується аналогічно: додаються всі розряди, включаючи знаковий, як звичайні біти. Якщо виникає перенесення зі знакового розряду, воно ігнорується.

Приклад. Додати числа $X_{10} = 45$, $Y_{10} = -73$ в доповняльному коді. Результат подати в прямому коді.

Розв'язання:

```
[X]д:   0  0101101
        +
[Y]д:   1  0110111
        ---------------
```

1 1100100

$[X+Y]_з$: 1 1100011;

$[X+Y]_п$: 1 0011100.

1.5.4. Модифіковані коди

Модифіковані коди застосовуються для прискорення перевірки переповнення розрядної сітки. Вони відрізняються тим, що для позначення знака числа використовується два біти:

для додатного числа знакові біти містять 00;

для від'ємного числа записуються 11;

Приклад. Подати числа X_{10} = 45, Y_{10} =-73 в прямому, зворотному і додатковому модифікованих кодах.

Розв'язання:

$[X]_п$: 00 0101101; $[X]_з$: 00 0101101; $[X]_д$: 00 0101101.

$[Y]_п$: 11 1001001; $[Y]_з$: 11 0110110; $[Y]_д$: 11 0110111.

Алгебраїчне додавання чисел у модифікованих кодах виконується за такими ж правилами, як і в звичайних кодах, однак знакові розряди аналізуються як частина цілого числа.

Приклад. Додати числа X_{10} = 45, Y_{10} =-73 в модифікованому доповняльному коді. Результат подати в модифікованому прямому коді.

Розв'язання:

$[X]_{мд}$: 00 0101101

\+

$[Y]_{мд}$: 11 0110111

---

11 1100100

$[X+Y]_{мз}$: 11 1100011;

$[X+Y]_{мп}$: 11 0011100.

1.6. Постановка задачі

Розробити алгоритм та на одній із алгоритмічних мов програмування написати комп’ютерну програму для інтерпретації арифметичних обчислень процесора над цілими беззнаковими числами. Вхідні дані зчитати в десятковій системі числення. Обрати розмір розрядної сітки. Перетворити десяткові числа у двійкові коди, а потім у машинні коди: прямий, зворотний і доповняльний. Провести операції додавання, віднімання, множення і ділення. Зробити зворотне перетворення отриманих машинних кодів у двійкову систему числення, а потім у десяткову. Візуалізувати результати роботи процесора за допомогою вікон.

Здійснення арифметичних операцій процесора повинно відбуватися шляхом вибору відповідного пункту з меню. Вихід з програми має здійснюватися натисканням кнопки «Вихід» або клавіші Esc.

Для реалізації застосунку вибрати самостійно одну із мов програмування високого рівня із середовищем візуальної розробки.

РОЗДІЛ 2
АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

2.1. Програми-калькулятори.

Windows Calculator – це офіційний застосунок для операційної системи Windows [38]. Він має велику кількість конвертерів, працює в декількох режимах (стандартний, науковий, програмований та графічний). Це робить Windows Calculator одним із найфункціональніших для Windows.

Програмований калькулятор

- дозволяє здійснювати переведення цілого числа між системами числення: десятковою, двійковою, вісімковою, шістнадцятковою;
- дозволяє здійснювати арифметичні операції над цілими числами: додавання, віднімання, множення, ділення.

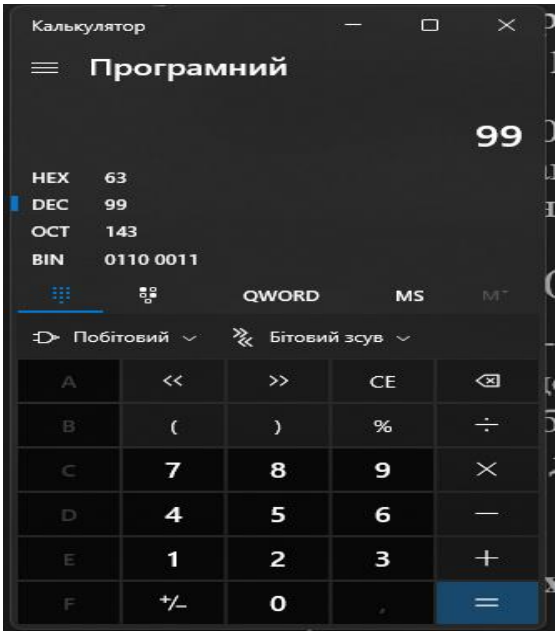


Рисунок 2.1.Калькулятор Windows

Андроїд-додаток «Двійковий калькулятор» від Code Builders Apps (рис. 2.2).

Є додатком для андроїд системи. Цей інтуїтивно зрозумілий двійковий калькулятор виконує складні двійкові операції. За своєю суттю він полегшує перетворення двійкових чисел у різні числові системи, включаючи

шістнадцяткову (перетворення десяткової у вісімкову), десяткову та навіть текстову [29].

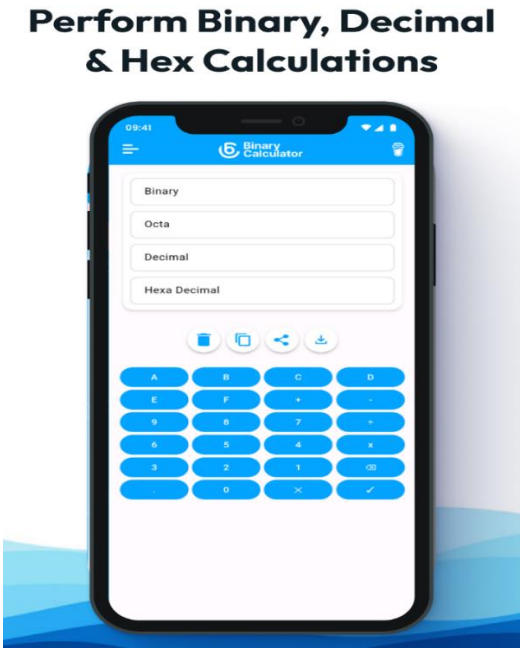


Рисунок 2.2. Андроїд-додаток «Двійковий калькулятор» *Code Builders Apps*

Андроїд-додаток «Двійковий калькулятор» від Enzipe Apps (рис.2.3).

Цей калькулятор дозволяє здійснювати обчислення всіх арифметичних операцій над двійковими числами і надавати результати за декілька секунд [30].

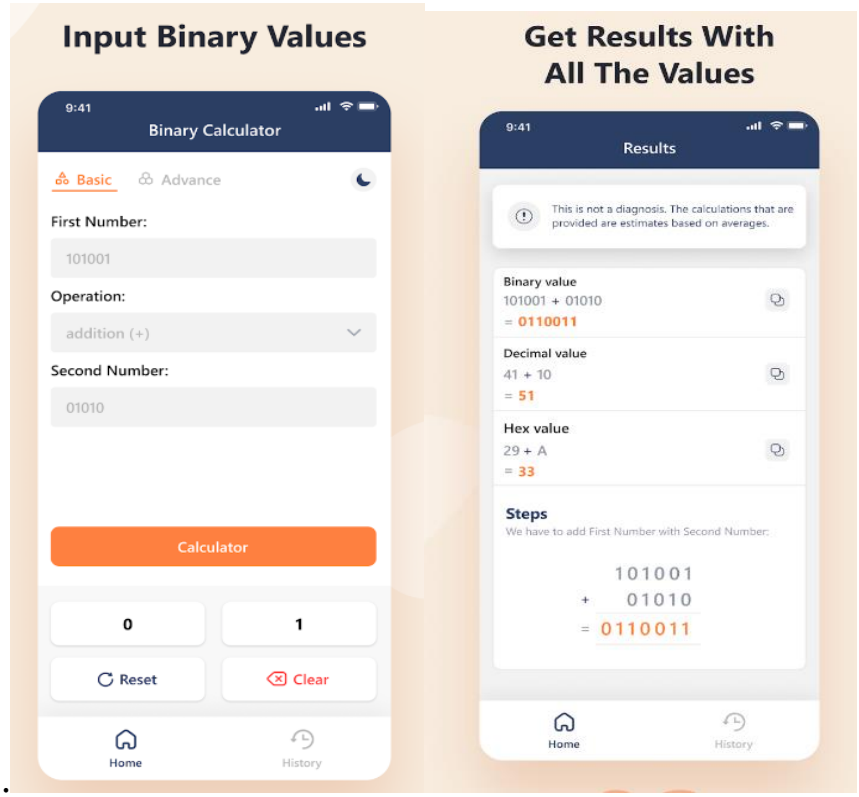


Рисунок 2.3. Андроїд-додаток «Двійковий калькулятор» від Enzipe Apps

2.2. Сервіси-калькулятори

Convertworld – безкоштовний веб-застосунок, один з найпопулярніших у світі. Доступний більш, ніж на двадцяти мовах [31]. Дозволяє перетворювати числа з десяткової системи числення у двійкову, вісімкову, шіснадцяткову.

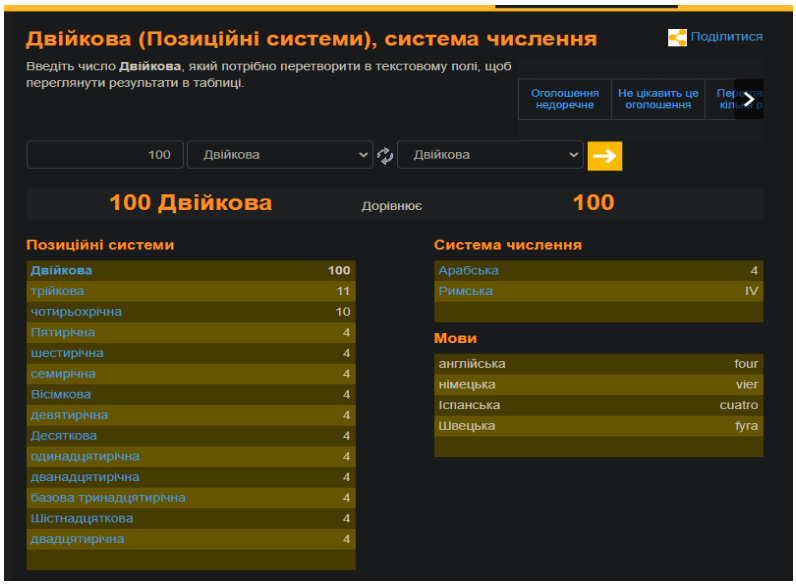


Рисунок 2.4. Сторінка Convertworld

Конвертер чисел, що переводить числа з однієї величини в іншу (рис. 2.5). Конвертер підтримує двійкову, вісімкову і шістнадцяткову систему числення для переведення в десяткову систему [32].

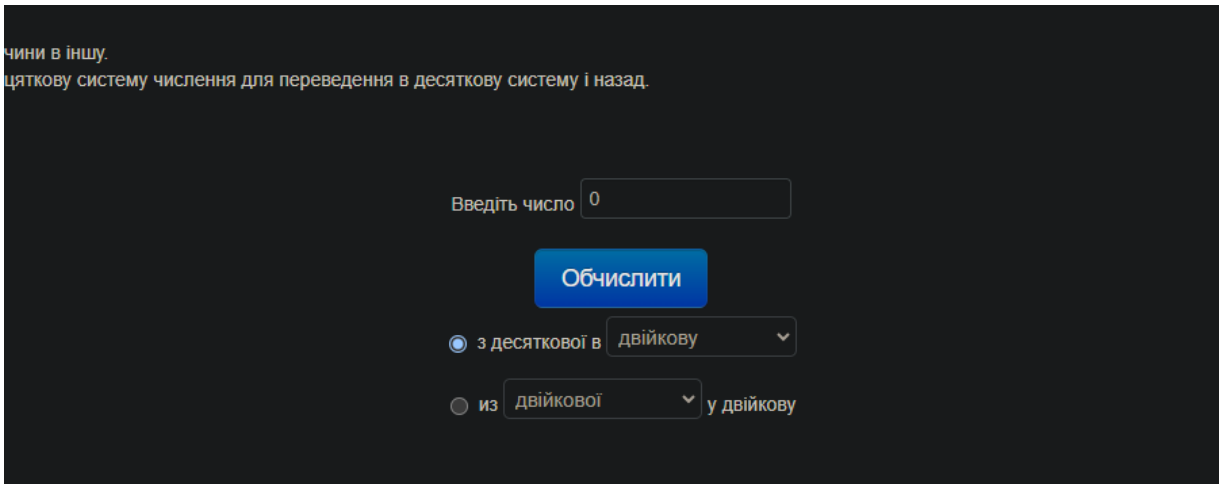


Рисунок 2.5. Головна сторінка сервісу «Конвертер чисел»

Калькулятор систем числення – онлайн-сервіс, що виконує конвертацію числа з однієї системи числення в іншу. Число має бути більшим або рівним 0. Основа системи числення може бути від 2 до 36 включно [33].

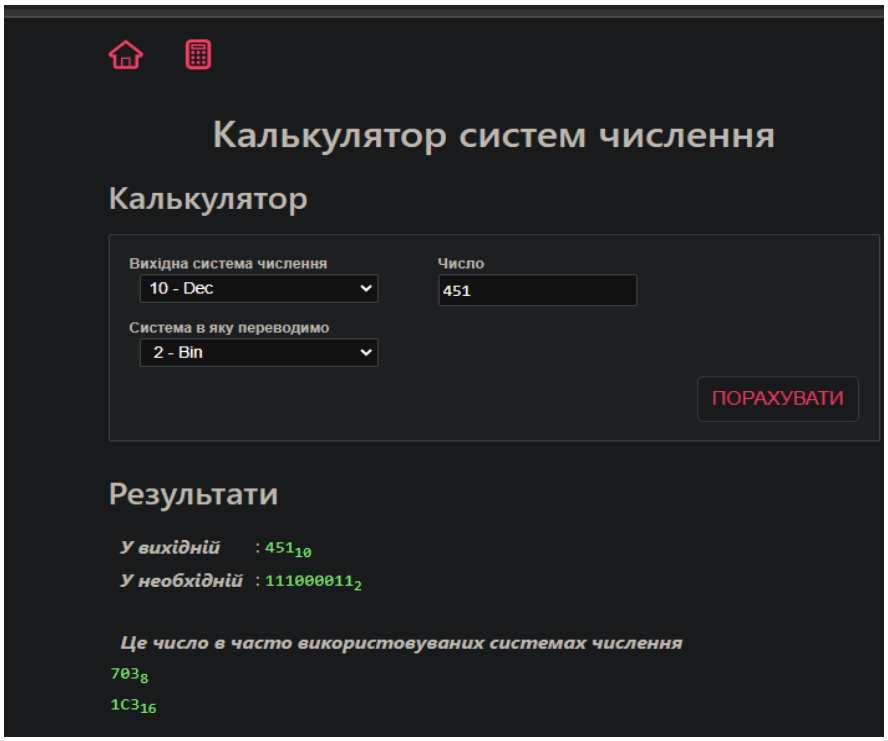


Рисунок 2.6. Головна сторінка сервісу «Калькулятор систем числення»

РОЗДІЛ 3
ОГЛЯД ІНСТРУМЕНТАРІЮ РОЗРОБКИ

3.1. Середовище розробки

Для створення програмного забезпечення було обрано Visual Studio – інтегроване середовище розробки (IDE), яке підтримує C#. Visual Studio забезпечує зручність написання, тестування та налагодження коду, а також містить багатофункціональні інструменти для створення інтерфейсів користувача та управління проєктами.

Основні переваги Visual Studio для цієї задачі:

- інтеграція з системою управління версіями (Git);
- вбудований налагоджувач для тестування алгоритмів;
- підтримка розширень, таких як інструменти для роботи з UI (Windows Forms, WPF).

Щоб розробити інформаційну систему для інтерпретації арифметичних операцій у двійкових кодах процесора застосовуємо Visual Studio 2019, а програмний продукт пишемо на мові програмування Сі шарп.

Microsoft Visual Studio — це набір продуктів компанії Microsoft, який включає інтегроване середовище розробки програмного забезпечення (IDE) та інші інструменти. Visual Studio дозволяє створювати як консольні, так і графічні програми з підтримкою Windows Forms. Крім того, середовище підтримує розробку веб-сайтів, веб-додатків, веб-служб у різних кодах, як власному, так і керованому, для платформ, сумісних із Microsoft Windows, Windows Mobile, Windows Phone, Windows CE, .NET Framework, .NET Compact Framework і Microsoft Silverlight.

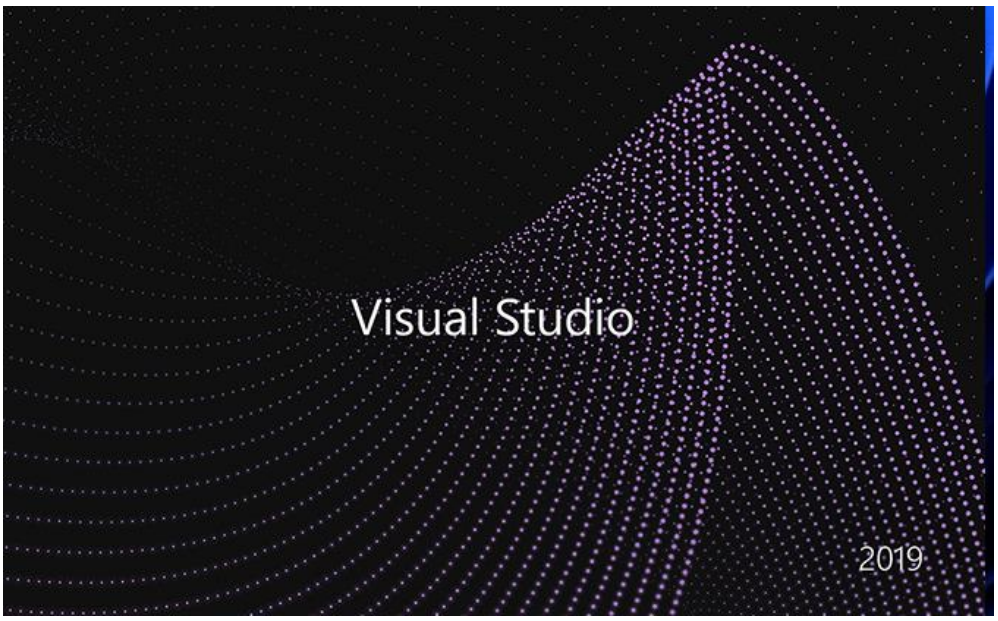


Рисунок 3.1. Логотип при запуску середовища програмування Visual Studio 2019

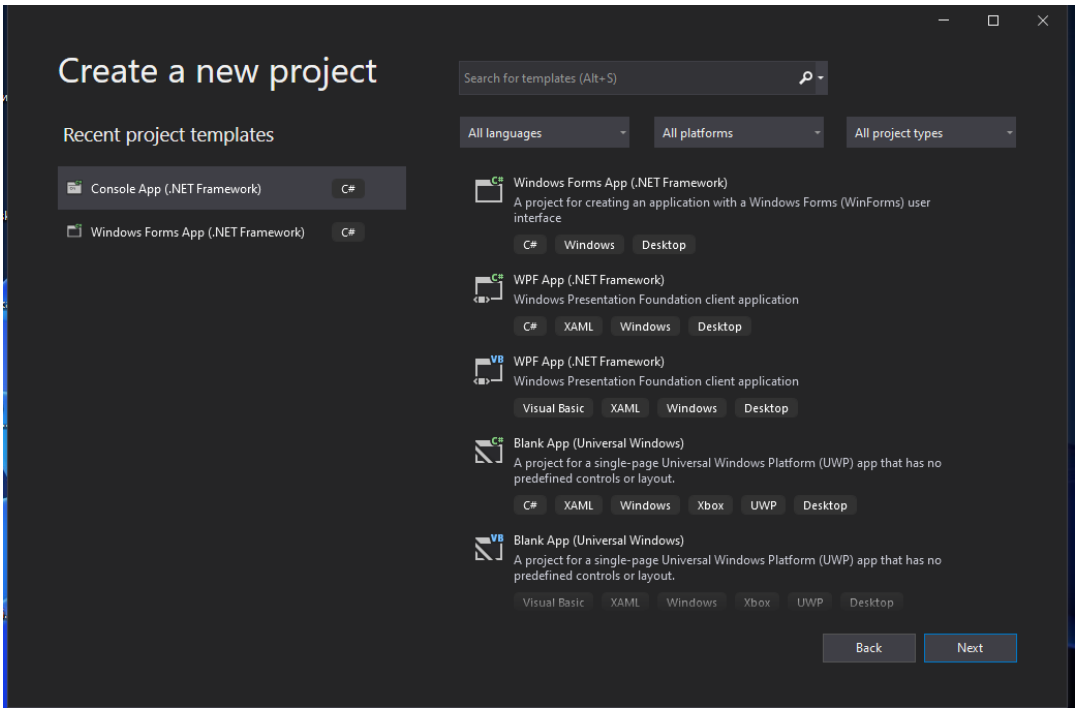


Рисунок 3.2. Вікно створення нового проєкту

Visual Studio 2019 була анонсована у грудні 2018 року з випуском Preview-версії, яка проходила через п’ять етапів. Версія Release Candidate (RC) з'явилася 2 березня 2019 року, а фінальна стабільна версія стала доступною 2 квітня 2019 року для Windows і macOS. Visual Studio традиційно пропонується у трьох редакціях: Community, Professional і Enterprise.

3.2. Мова розробки

Обрана мова розробки – C# (рис. 3.3). Вона надає потужні інструменти для роботи з числами, алгоритмами та взаємодії з користувачем. Її перевагами є:

- висока продуктивність та зручність реалізації математичних обчислень;
- легкість реалізації алгоритмів перетворення чисел між різними системами числення;
- підтримка графічного інтерфейсу для взаємодії з користувачем.

Рисунок 3.3. Логотип мови C#

C# (читається як "Сі-Шарп") — це об’єктно-орієнтована мова програмування з системою статичної типізації, розроблена для платформи .NET. Її створенням займалися Андерс Гейлсберг, Скот Вілтамут і Пітер Гольде під егідою Microsoft Research.

Синтаксис C# близький до мов C++ та Java. Мова підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, а також коментарі у форматі XML. Вона поєднує найкращі практики мов C++, Object Pascal, Modula, Smalltalk, але усуває конструкції, які можуть викликати складнощі при розробці, наприклад, множинне успадкування класів, що присутнє в C++, у C# відсутнє.

На вересень 2023 року актуальною стабільною версією мови є C# 11.0, випущена у 2022 році в складі платформи .NET 7.0.

Походження назви мови. Назва "C#" походить від символу #, який символізує дві пари плюсових знаків (++), що вказує на розвиток мови порівняно з C++, подібно до еволюції від C до C++. Також символ асоціюється з музичним дієзом (♯), що додає назві стилістичної особливості. Хоча символ # на клавіатурі

зазвичай використовується для позначення номера, Microsoft неодноразово зверталася до спільноти із закликом прийняти таку стилізацію назви.

3.3. Технології та засоби реалізації застосунку

Windows Forms: використовується для створення графічного інтерфейсу користувача і входить до складу Microsoft .NET Framework. Цей інтерфейс забезпечує спрощений доступ до об'єктів інтерфейсу Microsoft Windows, створюючи обгортку для API Win32 у керованому коді. Завдяки цьому класи, які реалізують API Windows Forms, не залежать від конкретної мови програмування. Тому розробники можуть використовувати Windows Forms незалежно від того, чи пишуть на C#, C++, VB.Net, J# або інших мовах.

Windows Forms, з одного боку, є спрощенням та заміною старішої бібліотеки MFC, яка була розроблена мовою C++. З іншого боку, WF не підтримує парадигми, подібної до MVC. Для вирішення цього недоліку існують сторонні бібліотеки, які додають таку функціональність. Однією з найпопулярніших бібліотек такого типу є User Interface Process Application Block, яку розробила спеціалізована команда Microsoft, що створює приклади та інструменти для безкоштовного використання. Ця бібліотека включає вихідний код та навчальні приклади, що допомагають швидше освоїти роботу з Windows Forms.

NET Framework або .NET Core: Використовується як платформа для роботи додатка, забезпечуючи доступ до бібліотек для роботи з даними, математичними функціями та інтерфейсом.

Приклад створення віконних форм (рис.3.4):

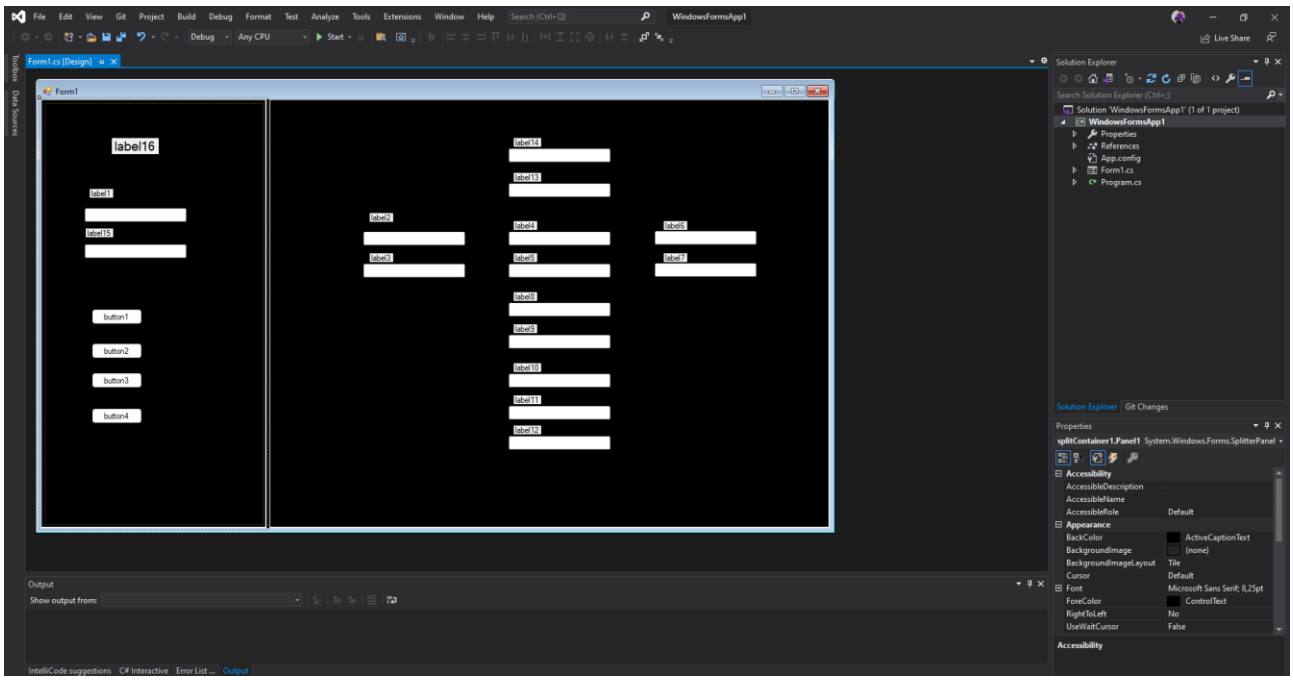


Рисунок 3.4. Створення віконної форми у Visual Studio 2019

Засоби тестування:

- вбудований налагоджувач Visual Studio дозволяє виявляти помилки на етапі виконання програми;
- Unit-тести (наприклад, за допомогою MSTest або NUnit): для перевірки правильності роботи кожного з алгоритмів окремо;
- Logger: для збереження журналу операцій, що дозволяє аналізувати, як програма виконує арифметичні обчислення.

Ці засоби та технології забезпечують повний цикл розробки додатку – від створення та тестування алгоритмів до реалізації інтуїтивного інтерфейсу для кінцевого користувача.

РОЗДІЛ 4
ПРОЄКТУВАННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ

4.1. Визначення функціональних та нефункціональних вимог до системи

Функціональні вимоги

1. Перетворення чисел: програма повинна

– приймати вхідні дані в десятковій системі числення;

– забезпечити переведення чисел з десяткової системи в двійкову;

– реалізувати перетворення чисел у машинний прямий, зворотний та додатковий коди.

2. Арифметичні операції: додавання, віднімання, множення і цілочисельне ділення.

3. Вивід результатів: результати повинні виводитися в зрозумілому для користувача форматі.

4. Обробка помилок:

– перевірка переповнення розрядної сітки;

– інформування користувача про некоректний ввід або помилки під час обчислень (наприклад, ділення на нуль).

5. Гнучкість налаштувань:

– можливість вибору користувачем операції, що виконується (додавання, віднімання, множення, ділення);

– підтримка як позитивних, так і від’ємних чисел.

Нефункціональні вимоги

1. Продуктивність:

– швидка обробка введених даних та виконання обчислень;

– мінімальні затримки навіть для чисел з великим розрядом.

2. Зручність використання:

– інтуїтивно зрозумілий інтерфейс користувача;

– покрокове відображення перетворень (за потреби користувача).

3. Надійність:

– захист від некоректних даних та дій, які можуть викликати збої;

– відсутність аварійних завершень програми.

4. Кросплатформеність:

програма повинна працювати на основних операційних системах (Windows, macOS, Linux).

5. Масштабованість:

легкість внесення змін чи додавання нових функцій (наприклад, підтримка інших систем числення).

6. Тестованість:

програма повинна бути протестована на різних наборах даних для виявлення та виправлення помилок.

7. Документація:

– наявність інструкцій для користувача;

– технічна документація для розробників.

8. Безпека:

використання механізмів для захисту від некоректного або зловмисного введення даних.

4.2. Проєктування застосунку

Алгоритми та математичні перетворення:

- перетворення десяткових чисел у двійкову систему числення.
- формування прямих, зворотних і доповнювальних кодів.
- виконання арифметичних операцій (додавання, віднімання, множення, ділення) у різних кодах.
- зворотного перетворення результату в десяткову систему числення.

Для роботи з двійковими числами використовуються масиви, що дозволяє точно реалізувати операції в машинних кодах.

Функціональна схема застосунку (рис. 4.1)

Інтерфейс введення даних:

приймає два цілі числа в десятковій системі числення;

дозволяє обрати арифметичну операцію: додавання, віднімання, множення, цілочисельне ділення;

дозволяє обрати тип машинного коду: прямий, зворотний або доповняльний.

Модуль перетворення вхідних даних:

перетворює числа з десяткової системи у двійкову;

створює представлення чисел у машинних кодах (прямий, зворотний, доповняльний).

Модуль арифметичних операцій:

виконує обрані арифметичні операції в межах машинних кодів;

реалізує алгоритми для кожного типу машинного коду (з урахуванням переповнень тощо).

Модуль перетворення результатів:

перетворює результат із машинного коду в двійкову систему;

переводить результат із двійкової системи в десяткову.

Інтерфейс виведення результатів:

виводить проміжні результати (машинний код, двійкове представлення) та фінальний результат у десятковій системі.

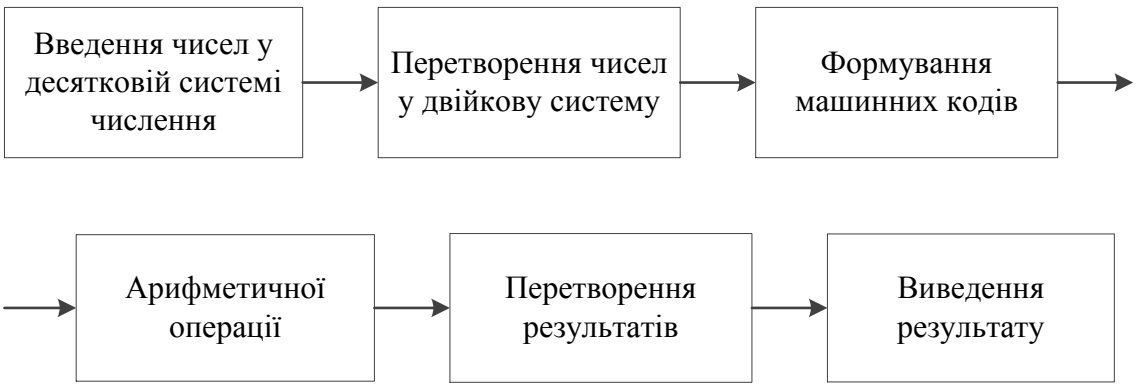


Рисунок 4.1. Функціональна схема застосунку

Логічна схема застосунку (рис. 4.2.)

Кроки роботи програми:

Введення даних:

два десяткових числа: a і b;

обирається операція (+, -, *, /);

обирається тип машинного коду (прямий, зворотний, доповняльний).

Перетворення у двійкову систему:

обидва числа переводяться у двійкову систему числення;

для від'ємних чисел враховується знак.

Формування машинних кодів:

створюється прямий код: додається біт знаку;

створюється зворотний код: інвертуються біти після біта знаку;

створюється доповняльний код: додається одиниця до зворотного коду.

Виконання арифметичної операції:

у вибраному машинному коді виконується арифметична операція;

ураховується можливість переповнення (для результату виділяється фіксована кількість бітів).

Перетворення результату:

результат операції у машинному коді перетворюється у двійкову систему;

двійковий результат переводиться у десяткову систему числення.

Виведення результату:

виводиться результат у десятковій системі, а також проміжні дані (машинний код, двійкове представлення).

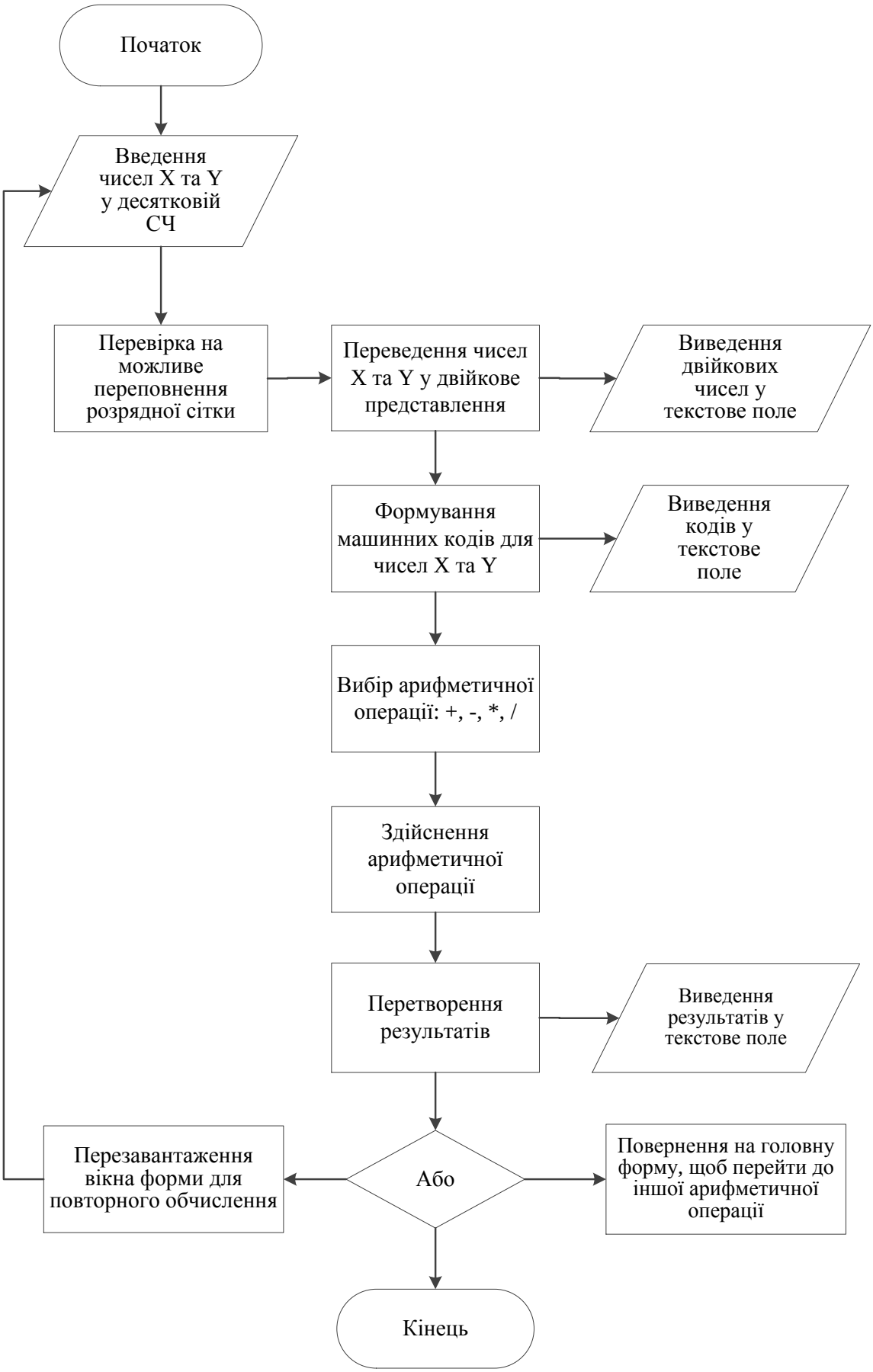


Рисунок 4.2. Логічна схема застосунку

4.3. Реалізація графічного інтерфейсу

У розроблюваній програмі було створено 6-форм, які в подальшому були приєднані до програмного продукту за допомогою компонентів Visual Studio 2019.

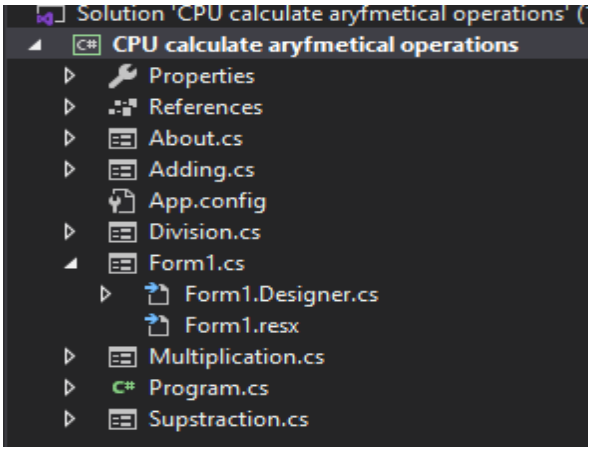


Рисунок 4.3. Склад проєкту застосунку

Характеристика розробленого продукту:

- є аналогом або подібною програмою, інформаційною системою ;
- дає можливість переходити на форми *Додавання*, *Віднімання*, *Множення*, *Ділення* для здійснення арифметичних операцій та відображення даних у двійковому і десятковому численні та в машинних кодах (рис. 4.4).

Рисунок 4.4. Головна форма застосунку

Характеристика форми *Додавання* (рис.4.5): дає можливість

- здійснювати введення чисел X та Y у десятковій системі;
- здійснювати обраховувати суму чисел X та Y;
- виводити значення чисел X та Y на текстові поля в двійковому коді у прямому, зворотному, і доповненому кодах;
- виводити результати операції на текстові поля у двійковому численні та в прямому, зворотному і доповняльному кодах.

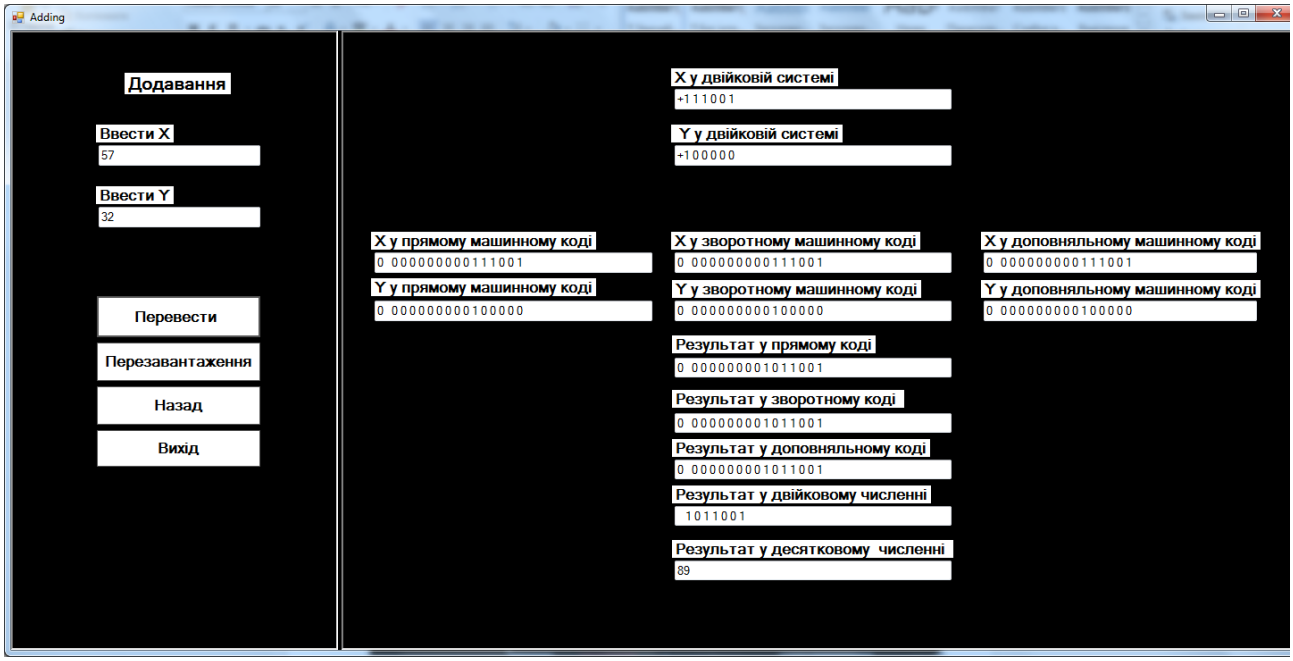


Рисунок 4.5. Форма додавання

Характеристика форми *Віднімання* (рис.4.6): дає можливість

- здійснювати введення чисел X та Y у десятковій системі;
- здійснювати обрахунок різниці чисел X та Y;
- виводити значення чисел X та Y на текстові поля в двійковому численні та в прямому, зворотному і доповняльному кодах;
- виводити результати операції на текстові поля у двійковому численні та в прямому, зворотному і доповняльному кодах.

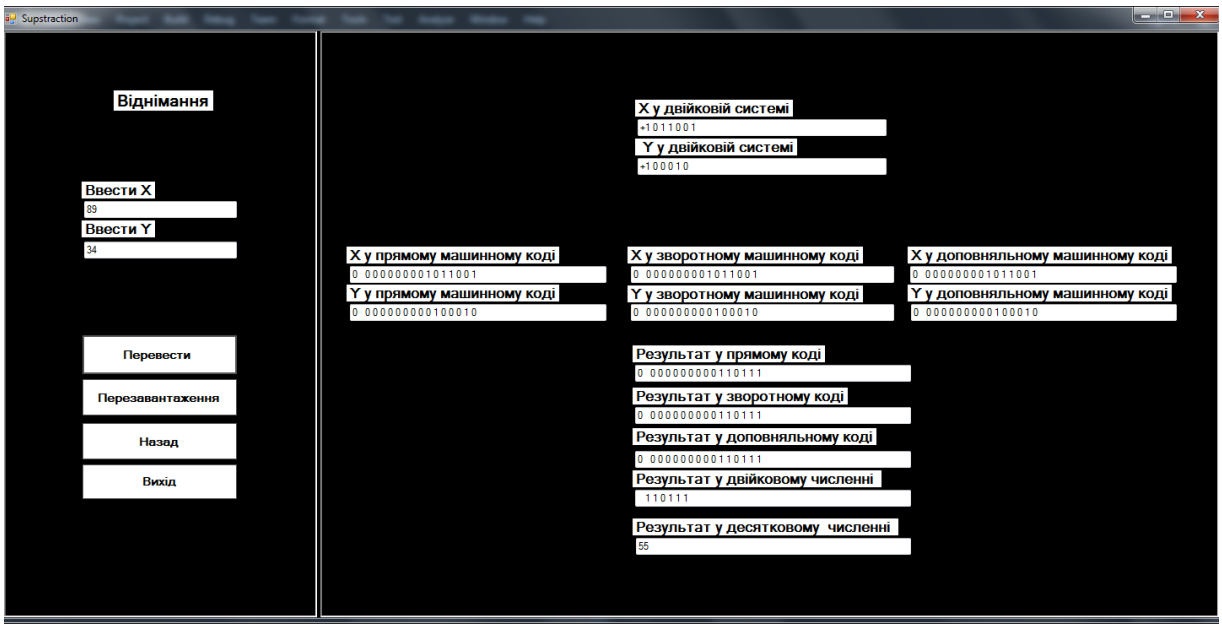


Рисунок 4.6. Форма віднімання

Характеристика форми *Множення* (рис.4.7): дає можливість

- здійснювати введення чисел X та Y у десятковій системі;
- здійснювати обрахунок добутку чисел X та Y;
- виводити значення чисел X та Y на текстові поля в двійковому численні та в прямому, зворотному і доповняльному кодах;
- виводити результати операції на текстові поля у двійковому численні та в прямому, зворотному і доповняльному кодах.

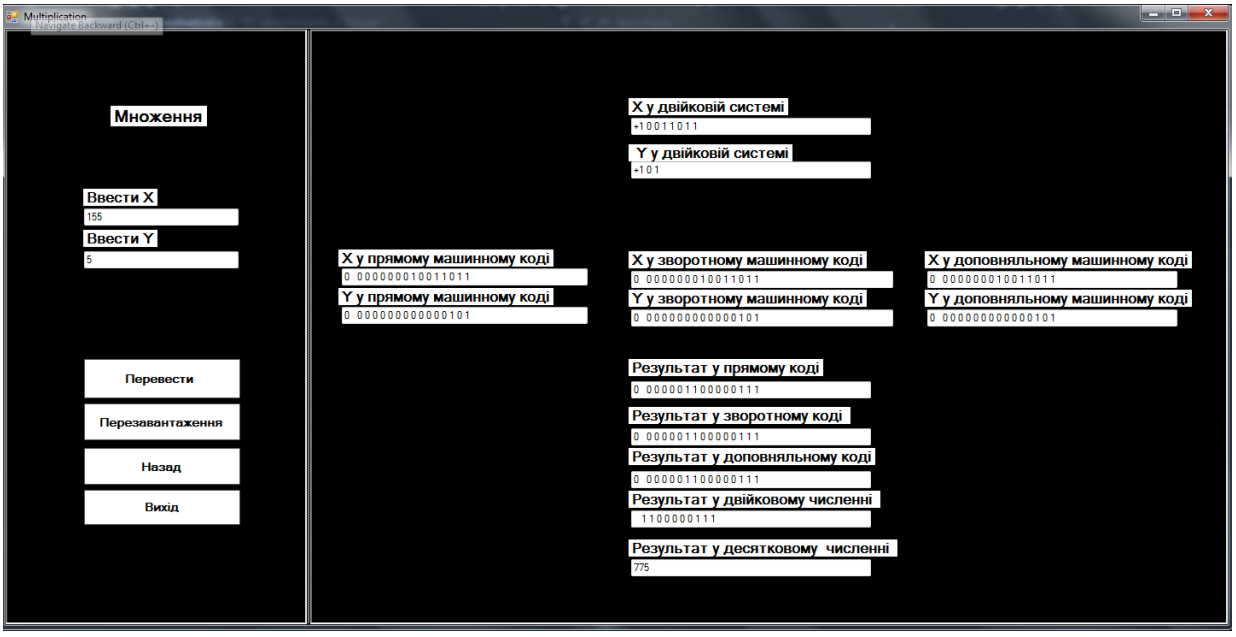


Рисунок 4.7. Форма множення

Характеристика форми *Ділення* (рис.4.8): дає можливість

- здійснювати введення чисел X та Y у десятковій системі;
- здійснювати обрахунок цілої частини від ділення чисел X та Y;
- виводити значення чисел X та Y на текстові поля в двійковому численні та в прямому, зворотному і доповняльному кодах;
- виводити результати операції на текстові поля у двійковому численні та в прямому, зворотному і доповняльному кодах.

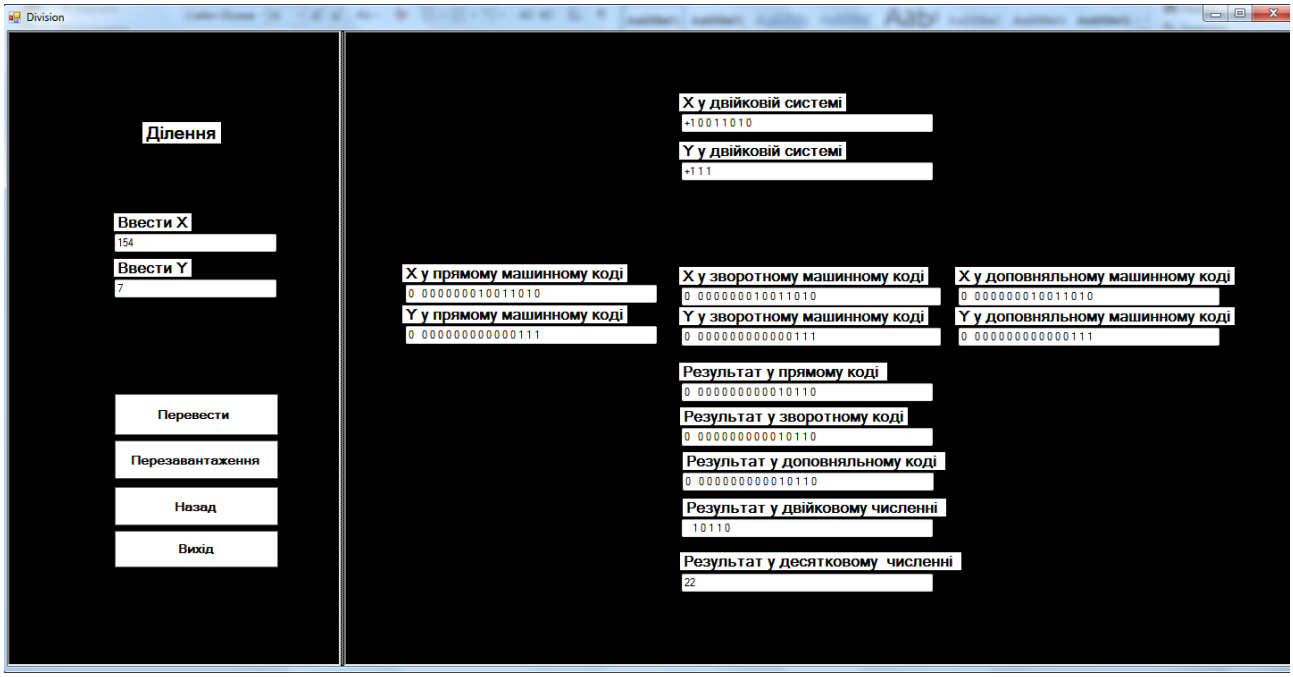


Рисунок 4.8. Форма ділення

У випадку виключної ситуації, коли відбувається переповнення розрядної сітки, на екран з’являється інформаційне повідомлення (рис. 4.9).

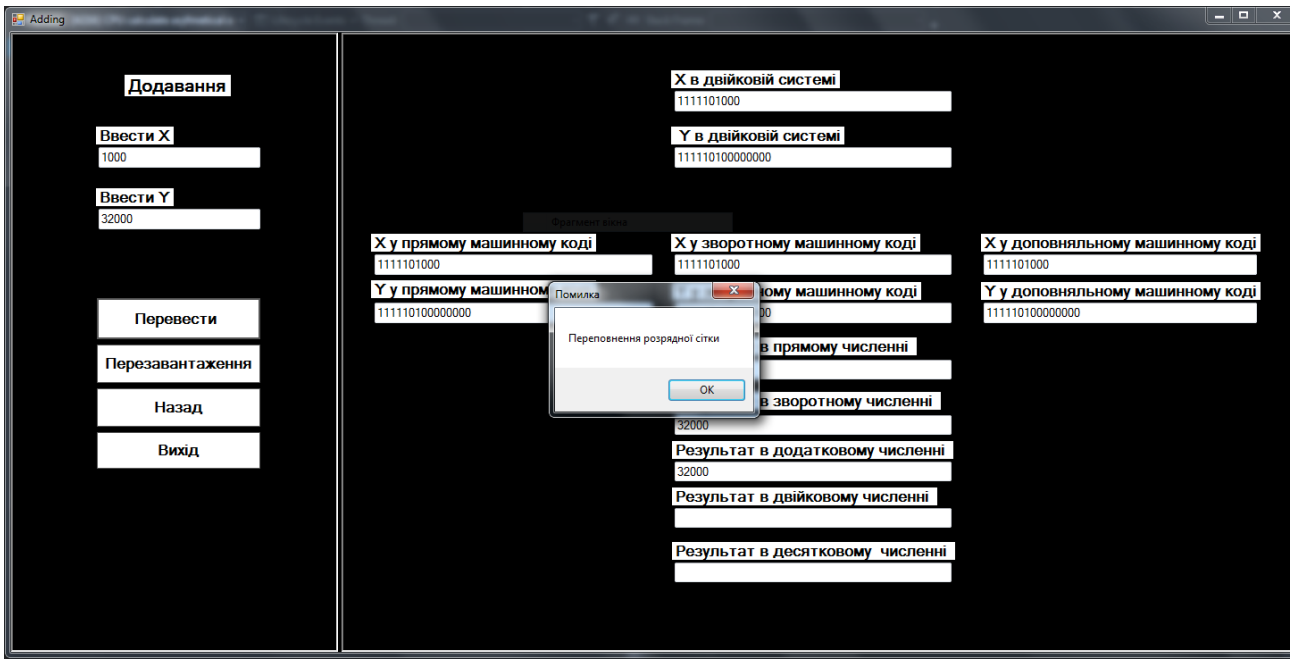


Рисунок 4.9. Повідомлення про переповнення розрядної сітки

Характеристика форми *Про програму* (рис. 4.10): на формі розміщена інформація

- про застосунок (назва і призначення, версія);
- про розробника (ПІБ, статус, фото).

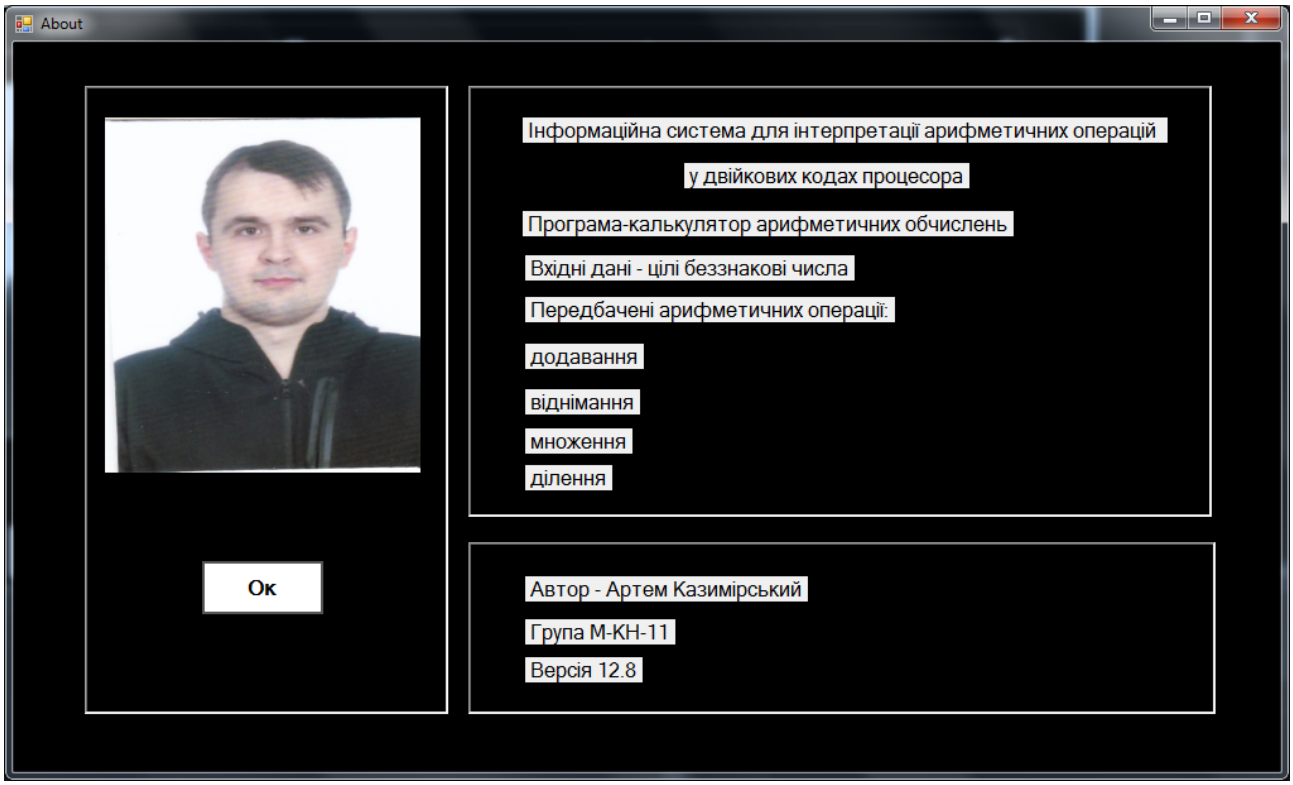


Рисунок 4.10. Графічний інтерфейс форми *Про програму*

4.4. Програмна реалізація застосунку

Проаналізуємо програмну реалізацію алгоритмів поставленої задачі, наприклад, для форми, що реалізує операцію додавання цілих беззнакових чисел. Вміст відповідного файлу Adding.cs наступний:

```
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 CPU_calculate_aryfmetical_operations
{
    public partial class Adding : Form
    {
        public int K;
        public int X;
        public int Y;
        public int P;
        public int Z;
        public int D;
        public int MPX;
        public int MPY;
        public int MZX;
```

```
        public int MZY;
        public int MDX;
        public int MDY;
//Встановлюємо розмір розрядної сітки:
       int N = 16;

        private int i;
//Масиви для збереження двійкового представлення операндів
операції та результату
        private int[] a2;
        private int[] a3;
        private int[] a;

        public int K2 { get; private set; }
        public int K3 { get; private set; }
//Змінна, у якій зберігатиметься результат проведеної операції
        public int R { get; private set; }
        public Adding()
        {
            InitializeComponent();
        }
        private void textBox13_TextChanged(object sender,
EventArgs e)
        {
        }
private void textBox1_TextChanged(object sender, EventArgs e)
        {
            X = Convert.ToInt32(textBox1.Text);
           textBox5.Text = Convert.ToString(X, 2);
```

```
            textBox8.Text = Convert.ToString(X, 2);
            textBox11.Text = Convert.ToString(X, 2);
            textBox17.Text = Convert.ToString(X, 2);
            textBox9.Text = Convert.ToString(X);
            textBox10.Text = Convert.ToString(X);
            textBox25.Text = Convert.ToString(X);
        }

        private void textBox2_TextChanged(object sender,
EventArgs e)
        {
            Y = Convert.ToInt32(textBox2.Text);
            textBox4.Text = Convert.ToString(Y, 2);
            textBox15.Text = Convert.ToString(Y, 2);
            textBox14.Text = Convert.ToString(Y, 2);
            textBox19.Text = Convert.ToString(Y, 2);
            textBox9.Text = Convert.ToString(Y);
            textBox10.Text = Convert.ToString(Y);
            textBox25.Text = Convert.ToString(Y);
        }
//Код, що реалізує основний алгоритм задачі:
private void bConvert_Click(object sender, EventArgs e)
        {
            int j, m, xx;
//Введення чисел X та Y
            X = Convert.ToInt32(textBox1.Text);
            Y = Convert.ToInt32(textBox2.Text);
```

```
// Умова, яка застосовується для перевірки переповнення розрядної сітки та виведення відповідного інформаційного сповіщення:
            if ((X + Y) > 32767 || X * Y > 32767)
            {
                MessageBox.Show(" Переповнення розрядної сітки", "Помилка");
                Form ifrm = new Adding();
                ifrm.Show();
                this.Hide();
            }
//масиви прямого, зворотного та доповняльного коду числа X відповідно
            uint[] p = new uint[N];
            uint[] z = new uint[N];
            uint[] d = new uint[N];
            for (int i = 0; i < N; i++) p[i] = 0;
            for (int i = 0; i < N; i++) z[i] = 0;
            for (int i = 0; i < N; i++) d[i] = 0;
//Визначення розміру масиву для зберігання числа X у двійковому численні:
            if (X != 0)
            {
                K = (int)Math.Log(Math.Abs(X), 2) + 1;
            }
            else
            {
                K = 1;
            }
```

```
// Оголошення динамічного масиву для двійкового числа X
            a = new int[K];
            xx = Math.Abs(X);
            i = K - 1;
// Заповнення бітами динамічного масиву двійкового числа X
            while (xx > 1)
            {
                m = xx % 2;
                a[i] = m; i--;
                xx = xx / 2;
            }
            a[i] = xx;
            textBox5.Text = Convert.ToString(+X, 2);
            if (X < 0) textBox5.Text = Convert.ToString('-');
                if (X >= 0) textBox5.Text =
Convert.ToString('+');
            for (int i = 0; i < K; i++)
                        textBox5.Text = Convert.ToString(a[i],
2);
// Встановлення знакових бітів у масиви машинних кодів числа
X:
            if (X < 0)
            {
                p[0] = 1;
                z[0] = 1;
                d[0] = 1;
            }
// Визначення значущих бітів машинних кодів числа X, запис їх
у відповідні масиви:
```

```
            for (i = K - 1, j = N - 1; i >= 0 && j > 0; i--,
j--)
            {
                p[j] = (uint)a[i];
            }
            if (X < 0)
            {
                for (j = 1; j < N; j++)
                {
                    if (p[j] == 0) z[j] = 1; else z[j] = 0;
                }
                j = N - 1; m = 1;
                for (j = N - 1; j > 0; j--)
                {
                    d[j] = (uint)((z[j] + m) % 2);
                    m = (int)((z[j] + m) / 2);
                }
            }
            else for (j = 1; j < N; j++)
                {
                    z[j] = p[j]; d[j] = p[j];
                }
            textBox8.Text = Convert.ToString(p[0]);
            textBox8.AppendText("   ");
            for (int i = 1; i < N; i++)
            {
                textBox8.AppendText(p[i].ToString() + " ");
            }
```

```
            textBox11.Text = Convert.ToString(z[0]);
            textBox11.AppendText("   ");

            for (int i = 1; i < N; i++)
            {
               textBox11.AppendText(z[i].ToString() + " ");
            }
            textBox17.Text = Convert.ToString(d[0]);
            textBox17.AppendText("   ");
            for (int i = 1; i < N; i++)
            {
              textBox17.AppendText(d[i].ToString() + " ");
            }
//Масиви прямого, зворотного та доповняльного коду числа Y
відповідно:
            uint[] p2 = new uint[N];
            uint[] z2 = new uint[N];
            uint[] d2 = new uint[N];
            for (int i = 0; i < N; i++) p2[i] = 0;
            for (int i = 0; i < N; i++) z2[i] = 0;
            for (int i = 0; i < N; i++) d2[i] = 0;
//Розмір масиву для двійкового представлення числа Y
            if (Y != 0)
            {
                K2 = (int)Math.Log(Math.Abs(Y), 2) + 1;
            }
            else
            {
                K2 = 1;
```

```
            }
            a2 = new int[K2];
            xx = Math.Abs(Y);
            i = K2 - 1;
            while (xx > 1)
            {
                m = xx % 2;
                a2[i] = m; i--;
                xx = xx / 2;
            }
            a2[i] = xx;
            textBox4.Text = Convert.ToString(+Y, 2);
            if (Y < 0) textBox4.Text = Convert.ToString('-');
            if (Y >= 0) textBox4.Text = Convert.ToString('+');
            for (uint i = 0; i < K2; i++)
                        textBox4.Text =
Convert.ToString(a2[i], 2);
// Встановлення знакових бітів у масиви машинних кодів числа
X:
            if (Y < 0)
            {
                p2[0] = 1;
                z2[0] = 1;
                d2[0] = 1;
            }
// Обчислення і запис у масиви значущих бітів двійкових кодів
числа Y:
```

```
            for (i = K2 - 1, j = N - 1; i >= 0 && j > 0; i--,
j--)
            {
                p2[j] = (uint)a2[i];
            }
            if (Y < 0)
            {
                for (j = 1; j < N; j++)
                {
                    if (p2[j] == 0) z2[j] = 1; else z2[j] = 0;
                }
                j = N - 1; m = 1;
                for (j = N - 1; j > 0; j--)
                {
                    d2[j] = (uint)((z2[j] + m) % 2);
                    m = (int)((z2[j] + m) / 2);
                }
            }
            else for (j = 1; j < N; j++)
                {
                    z2[j] = p2[j]; d2[j] = p2[j];
                }
            textBox15.Text = Convert.ToString(p2[0]);
            textBox15.AppendText("   ");
            for (int i = 1; i < N; i++)
            {
                textBox15.AppendText(p2[i].ToString() + " ");
            }
            textBox14.Text = Convert.ToString(z2[0]);
```

```
            textBox14.AppendText("   ");
            for (int i = 1; i < N; i++)
            {
                textBox14.AppendText(z2[i].ToString() + " ");
            }
            textBox19.Text = Convert.ToString(d2[0]);
            textBox19.AppendText("   ");
            for (int i = 1; i < N; i++)
            {
                textBox19.AppendText(d2[i].ToString() + " ");
            }
            uint[] p3 = new uint[N];    // масив прямого коду
результату операції
            uint[] z3 = new uint[N];    // масив зворотного
коду
            uint[] d3 = new uint[N];    // масив доповняльного
коду
// Початкове обнулення масивів:
            for (int i = 0; i < N; i++) p3[i] = 0;
            for (int i = 0; i < N; i++) z3[i] = 0;
            for (int i = 0; i < N; i++) d3[i] = 0;
            int R;
            R = X + Y; // або R=x-y;, або R=y-x;, або R=-x-y;,
або  R=x*y; , або R=x/y
            if (R != 0)
            {
                K3 = (int)Math.Log(Math.Abs(R), 2) + 1;
            }
            else
```

```
            {
                K3 = 1;
            }
            a3 = new int[K3]; // Динамічний масив двійкового
числа R
            xx = Math.Abs(R);
            i = K3 - 1;
            while (xx > 1)
            {
                m = xx % 2;
                a3[i] = m; i--;
                xx = xx / 2;
            }
            a3[i] = xx;
// Запис знакових бітів у масиви машинних кодів результуючого
значення:
            if (R < 0)
            {
                p3[0] = 1;
                z3[0] = 1;
                d3[0] = 1;
            }
            for (i = K3 - 1, j = N - 1; i >= 0 && j > 0; i--,
j--)
                p3[j] = (uint)a3[i];

            if (R < 0)
            {
                for (j = 1; j < N; j++)
```

```
                {
                    if (p3[j] == 0) z3[j] = 1;
                    else z3[j] = 0;
                }
                j = N - 1; m = 1;

                for (j = N - 1; j > 0; j--)
                {
                    d3[j] = (uint)((z3[j] + m) % 2);
                    m = (int)((z3[j] + m) / 2);
                }
            }

            else for (j = 1; j < N; j++)
                {
                    z3[j] = p3[j];
                    d3[j] = p3[j];
                }
textBox9.Text = Convert.ToString(p3[0]);
            textBox9.AppendText("   ");
            for (int i = 1; i < N; i++)
            {
                textBox9.AppendText(p3[i].ToString() + " ");
            }
            textBox10.Text = Convert.ToString(z3[0]);
            textBox10.AppendText("   ");
            for (int i = 1; i < N; i++)
            {
                textBox10.AppendText(z3[i].ToString() + " ");
```

```
            }
            textBox25.Text = Convert.ToString(d3[0]);
            textBox25.AppendText("   ");
            for (int i = 1; i < N; i++)
            {
                textBox25.AppendText(d3[i].ToString() + " ");
            }
            if (R < 0) //cout << "-  ";
                if (R >= 0) //cout << "+  ";
            textBox16.Text = Convert.ToString(a3[0]);
            textBox16.AppendText("   ");
            for (int i = 0; i < K3; i++)
                    {
                        textBox16.AppendText(a3[i].ToString()
+ " ");
                    }
            textBox7.Text = Convert.ToString(R);
        }
     private void bReset_Click(object sender, EventArgs e)
        {
            Form ifrm = new Adding();
            ifrm.Show();
            this.Hide();
        }
        private void bBack_Click(object sender, EventArgs e)
        {
            Form ifrm = new Form1();
            ifrm.Show();
            this.Hide();
```

```
        }
        private void bExit_Click(object sender, EventArgs e)
        {
            Close();
        }
        private void label10_Click(object sender, EventArgs e)
        {
        }
    }
}
```

ВИСНОВКИ

Підсумовуючи результати даної магістерської роботи, можна зазначити, що основна мета була досягнута.

Виконанню головної цілі сприяли поставлені завдання, які виконувались поступово і ґрунтовно. Вивчались теоретичні джерела та дослідження провідних вітчизняних науковців, здійснено огляд позиційних систем числення, які використовуються в ЕОТ, було проаналізовано існуючі програми та сервіси, що переводять з десяткової у двійкову та деякі інші системи числення; проаналізована двійкова система числення комп'ютера, досліджений процес переведення цілих чисел з десяткової в двійкову систему і в зворотному напрямку. Було досліджено і застосовано переведення цілих чисел в машинні коди: прямий, зворотний і доповняльний, а також процес здійснення арифметичних операцій в машинних кодах: додавання, віднімання, множення, ділення і зворотній процес переведення результатів у десяткову систему числення. Були застосовані відповідні алгоритми переведення цілих чисел з десяткової в двійкову систему числення та алгоритми виконання арифметичних дій для створення застосунку.

Розробка програмного продукту виконана на мові програмування C Sharp із застосуванням вікон візуалізації Windows Forms. Для розробки інформаційної системи «Інтерпретація арифметичних операцій у двійкових кодах» застосовано Visual Studio 2019. Програма має зручний інтерфейс для керування операціями.

Завдяки розробленій програмі, знання про переведення з десяткової системи числення у двійкову систему, переведення навпаки і виконання арифметичних операцій в двійковій системі та в машинному коді будуть систематизовані. Доступ до них і використання в процесі занять і вивчення студентами вимагатиме менше часу, що сприятиме більш ефективному процесу навчання для досягнення цілей, і розширення розуміння переведення і виконання арифметичних операцій. У свою чергу, завдання освіти: професійне,

інтелектуальне вдосконалення і задоволення освітніх потреб студентів у цих знаннях – буде реалізовано.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Баркалов О. О., Барбаков Л. М. «Прикладна теорія цифрових автоматів в сучасній українській науці». Матеріали наукової конференції (2019 – 2020 рр.), ДНУ ім. Василя Стуса, 2021. С. 318 – 320.
2. Білак Ю. Ю., Данько-Товтин Л. Я. Системи числення: методичні рекомендації з базової теми дисципліни «Інформатика». Ужгород: ДВНЗ «УжНУ», 2015. 24 с.
3. Божко Н. В. Методичні матеріали для вивчення тем, запланованих на самостійне опрацювання з навчальної дисципліни «Прикладна теорія цифрових автоматів». Коледж МНУ ім. В. О. Сухомлинського. Миколаїв, 2019. 210 с.
4. Гавриленко С. Ю., Клименко А. М. Комп’ютерна логіка. Лабораторний практикум: навчально-методичний посібник. Харків: НТУ «ХПІ», 2015. 74 с.
5. Дичка І. А., Тарасенко В. П., Онай М. В. Основи прикладної теорії цифрових автоматів: підручник. Київ : КПІ ім. Ігоря Сікорського. Вид-во «Політехніка», 2019. 508 с.
6. Жабін В. І., Жуков І. А., Клименко І. А., Ткаченко В. В. Прикладна теорія цифрових автоматів. 2-ге видання, доопрацьоване. Київ : «НАУ— друк», 2009. 360 с.
7. Казимірський А.-Д. О., Бабич С. М. Розробка інформаційної системи «Інтерпретація математичних обчислень у двійкових кодах»// VI Всеукраїнська конференція молодих вчених «Актуальні питання розвитку інформаційних технологій»: матеріали конференції (м. Дніпро, 20 листопада 2024 р.)
8. Матвієнко М.П. Комп’ютерна логіка. Навчальний посібник. К.: "Ліра-К", 2012. 288 с.
9. Методичні рекомендації щодо виконання та оформлення кваліфікаційної роботи для здобувачів ступеня «магістр» за спеціальністю

122 Комп’ютерні науки / уклад.: С. В. Петренко, В. А. Сяський, С. М. Бабич, Рівне: РДГУ, 2024. 24 с.

10. Муринюк В. В. Комп’ютерна логіка: навчальний посібник з дисципліни для ІІІ курсу спеціальності 123 «Комп’ютерна інженерія». Чернівці: ДВНЗ «ЧІК», 2018. 60 с.
11. Обчислювальна техніка. Конспект лекцій. Частина 1. / Уклад. Ю. В. Зілінський. Кременчук: КНУ ім. М. Остроградського, 2010. 73 с.
12. Тарарака В. Д. Прикладна теорія цифрових автоматів: навчальний посібник. Житомир: ЖДТУ, 2019. 183 с.
13. Трохименко В. С. Конспект лекцій з математичної логіки та теорії алгоритмів. Вінниця: ВДПУ, 2007. 85 с.
14. Шкільняк С. С. Математична логіка. Основи теорії алгоритмів: навчальний посібник. Київ: ДП «Вид. дім «Персонал», 2009. 280 с.
15. Щербаков А. Н.. Проскурін М. П., Грушко С. С. Прикладна теорія цифрових автоматів. Частина 1. Комп’ютерна арифметика. Тексти лекцій. Запоріжжя: ЗНТУ, 2010. 102 с.
16. Яцків В.В. Прикладна теорія цифрових автоматів. Методичні вказівки до лабораторних робіт. Тернопіль: Економічна думка, 2005. 60 с.
17. Прикладна теорія цифрових автоматів. Опорний конспект лекцій. / Уклад. В. В. Яцків, Н. Г. Яцків. Тернопіль: Економічна думка, 2006. 89 с.
18. Системи числення. Позиційна і непозиційна система числення. URL: https://naurok.com.ua/prezentaciya-sistemi-chislennya-poziciyna-i-nepoziciyna-sistema-chislennya-222861.html (дата звернення: 19.02.2024)
19. Системи числення та подання даних у комп’ютері. URL: http://nkkep.com/wp-content/uploads/2020/03/L8_P-12_Informaty-ka_16.13.20_Popelyuk.pdf (дата звернення: 27.03.2024).
20. Дії в двійковій системі числення. Арифметичні дії в двійковій системі. URL: https://dviykov-sustema.blogspot.com/p/blog-page_9923.html (дата звернення: 24.04.2024).

21. Процесор, пам'ять ЕОМ. URL: https://www.slideshare.net/slideshow/ss-10210462/10210462 (дата звернення: 17.11.2023).
22. Процесор та його властивості. тема: Поняття, призначення та характеристики процесорів. URL: https://vseosvita.ua/lesson/protsesor-ta-ioho-vlastyvosti-273136.html (дата звернення: 13.10.2023).
23. Поняття, призначення та характеристики процесорів. URL: https://naurok.com.ua/urok-na-temu-ponyattya-priznachennya-ta-harakteristiki-procesoriv-50579.html (дата звернення: 11.08.2024).
24. Що таке процесор. URL: https://uk.wikipedia.org/wiki/Процесор (дата звернення: 24.04.2024).
25. Що таке центральний процесор. URL: https://uk.wikipedia.org/wiki/Центральний_процесор (дата звернення: 05.06.2024).
26. Система команд процесора. Основні групи команд процесора Особливості виконання різних команд. Методи організації підпрограм. URL: https://elib.lntu.edu.ua/sites/default/files/elib_upload/16%2C05/other/tema_8__sistema_komand_proczesora.pdf (дата звернення: 26.04.2024).
27. Історія обчислювальної техніки. URL: https://uk.wikipedia.org/wiki/Історія_обчислювальної_техніки (дата звернення: 19.10.2023).
28. Історія створення і розвитку засобів обчислювальної техніки. URL: https://itmuseum.sumdu.edu.ua/istoriya-stvorennya-i-rozvitku-zasobiv-obchislyuvalnoyi-tehniki (дата звернення: 05.11.2023).
29. Опис додатку про двійковий калькулятор виконує складні двійкові операції, перетворення двійкових чисел у різні числові системи. Автор Code Builders Apps. URL: https://play.google.com/store/apps/details?id=binarycalculator.converter.calculadorabinaria&hl=uk (дата звернення: 22.02.2024).
30. Опис додатку про обчислення двійкових чисел всіх арифметичних операцій над двійковими числами і надавати результати за пару секунд. Автор Enzipe Apps. URL: https://play.google.com/store/apps/details?id=com.ecalculator.binarycalculator&hl=uk (дата звернення: 08.05.2024).

31. Сервіс, що перекладає числа із десяткової системі числення у двійкову, вісімкову, шістнадцяткову URL: https://www.convertworld.com/uk/systema-chyslennya/dviykova.html (дата звернення: 10.10.2024).
32. Конвертер чисел, що переводить числа з однієї величини в іншу. URL: https://tools.sochka.com/numcalk.html (дата звернення: 10.10.2024).
33. Онлайн-сервіс що виконує переклад числа з однієї системи числення в іншу. URL: https://fin calc.org.ua/ua/calculator/conversion/notation/any/#google_vignette (дата звернення: 11.10.2024).
34. Переведення чисел. URL: https://komplogika.jimdofree.com/системи-числення-теорія/перевід-чисел/ (дата звернення: 23.09.2024).
35. Системи числення, що використовуються в комп'ютерах. URL: https://studfile.net/preview/9308999/page:2/ (дата звернення: 15.09.2023).
36. Системи числення. Арифметичні основи комп’ютерів. URL: https://moodle.znu.edu.ua/mod/resource/view.php?id=254940 (дата звернення: 01.12.2024).
37. Системи числення. URL: https://nrs.rozh2sch.org.ua (дата звернення: 02.12.2024).
38. Microsoft-підтримка. Використання калькулятора у Windows. URL: https://support.microsoft.com/uk-ua/windows/використання-калькулятора-у-windows-8dc0eb59-a45f-72b6-71bd-e752920f36c3 (дата звернення: 02.12.2024).