МІНІСТЕРСТВО НАУКИ І ОСВІТИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КПІ»

ФАКУЛЬЕТ ІНФОРМАТКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

КАФЕДРА ТЕХНІЧНОЇ КІБЕРНЕТИКИ

**Пояснювальна записка**

до курсової роботи

з дисципліни

“ Основи програмування - 2. Фундаментальні структури даних ”

на тему

«Операції з двадцятип’ятизначними числами»

|  |  |  |
| --- | --- | --- |
| Керівник : Тимофєєва Ю.С. |  | Виконав : Говоровський С.В. |
| Допущений до захисту |  | студент гр. ІТ-42, ФІОТ |
| І\_\_\_І \_\_\_\_\_\_\_\_ 201\_ \_\_\_\_\_\_  підпис |  | 1 курс  № IТ-4203 |
| Захистив з оцінкою  \_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_  оцінка підпис  І\_\_\_І \_\_\_\_\_\_\_\_\_ 201\_ |  |  |

Київ 2015

**Національний технічний університет України**

**“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”**

Факультет Інформатики та обчислювальної техніки

Кафедра Технічної Кібернетики

Напрям підготовки Програмна Інженерія

Спеціальність Програмне Забезпечення Систем

**ЗАВДАННЯ**

**на курсову роботу з дисципліни** «Основи програмування - 2. Фундаментальні структури даних»

Студенту Говоровському Сергію Валентиновичу

(прізвище, ім'я, по батькові)

**1.Тема роботи:** Операції з двадцятип’ятизначними числами

**2.Термін здачі студентом закінченого проекту** « » 20 р.

**3.Вихідні дані до роботи**

* Мова програмування C#
* Середовище програмування MS Visual Studio 2013
* Можливість проведення операцій з двадцятип’ятизначними числами
* Зручність інтерфейсу

**4.Перелік питань, які мають бути розроблені:**

Розробка та реалізація алгоритмів додавання, віднімання, множення та ділення

**5.Перелік графічного матеріалу**

Зовнішній вигляд інтерфейсу

**КАЛЕНДАРНИЙ ПЛАН - ГРАФІК**

виконання курсової роботи

|  |  |  |  |
| --- | --- | --- | --- |
| Пор  № | Назва етапів роботи та питань, які мають бути розроблені відповідно до завдань курсової роботи | Термін виконання | Позначки керівника про виконання завдань |
| 1 | Розробка технічного завдання | 25.04-29.04 |  |
| 2 | Вивчення арифметики довгих чисел з боку математики | 01.05-04.05 |  |
| 3 | Визначення вхідних та вихідних даних | 06.05-10.05 |  |
| 4 | Розробка алгоритму | 11.05-13.05 |  |
| 5 | Реалізація алгоритму | 14.05-17.05 |  |
| 6 | Реалізація необов’язкових функцій програми (необхідно, для полегшення використання) | 17.05-19.05 |  |
| 7 | Оформлення пояснювальної записки | 20.05-23.05 |  |

**Дата видачі завдання** .

**Керівник** \_\_\_\_\_\_\_\_\_\_\_

(підпис)

**Завдання прийняв до виконання** \_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

АНОТАЦІЯ

Темою курсової роботи є операції з двадцятип’ятизначними числами. В ході роботи було розроблено та реалізовано алгоритми додавання, віднімання, множення та ділення чисел, розряд яких є більшим ніж 25 (тобто числа більші, ніж 1\*1025).

В результаті виконання курсової роботи створено додаток, який може проводити основні операції з великими числами. Окремо було спроектовано та втілено у програмі інтуїтивно зрозумілий та зручний інтерфейс з додатковими функціями. Додаток написаний на мові програмування C# з використанням технології Windows Presentation Foundation (WPF) для створення графічної оболонки програми.

Створена програма є захищеною від неправильного вводу, а саме не можна вводити символи, що не є цифрами.

У першому розділі сформульовано вимоги до програми. Другий розділ містить основну теоретичну інформацію про арифметику та арифметику довгих чисел. У третьому розділі описуються використані алгоритми додавання, віднімання, множення та ділення великих чисел. Четвертий розділ пояснює реалізацію алгоритму на мові програмування C#. Інтерфейс та елементи інтерфейсу програми описано у п’ятому розділі. В кінці пояснювальної записки підведено підсумки та сформульований висновок, а також наведений список використаних джерел з додатками.

ЗМІСТ

ВСТУП……………………………………………………………………………………………….6

1 ПОСТАНОВКА ТА ІНФОРМАЦІЙНА СУТНІСТЬ ЗАВДАННЯ……………………….……7

2 ТЕОРЕТИЧНІ ВІДОМОСТІ ПРО АРИФМЕТИКУ ДОВГИХ ЧИСЕЛ ТА АЛГОРИТМИ ОПЕРАЦІЙ З НИМИ……………………...…………………………….……………………….8

2.1 Теоретичні відомості про арифметику довгих чисел…………………………..………8

2.2 Алгоритми операцій з довгими числами………..………………………………..……..9

3 ВИБІР ТА ОПИС АЛГОРИТМІВ, РЕАЛІЗОВАНИХ У ПРОГРАМІ………………………..11

3.1 Вибір алгоритму для роботи програми………………………………………………...11

3.2 Опис алгоритмів, реалізованих у програмі…………………………………………….11

4 ОПИС ПРОГРАМИ……….……….……….……….….….……….……….……….…………..15

4.1 Опис методів та подій……….……….……….……….…………….……….…………..15

4.2 Опис методів додатку……….……….…….………...…….……….……….……….…..16

4.3 Події у програмі……….……….……….…………....……….……….……….………...18

4.4 Опис класів у програмі……….…..…….……….….……….……….……….………….20

5 ОПИС ІНТЕРФЕЙСУ ПРОГРАМИ ТА КЕРІВНИЦТВО ДЛЯ КОРИСТУВАЧА…………..21

5.1 Опис інтерфейсу……….…...……….……….……….……….……….……….………..21

5.2 Інструкція по користуванню програмою для користувача……….…….….…………22

5.3 Інструкція для програміста……………………………………………………………...23

5.4 Системні вимоги……….……….……….……….……….……….……….……….……24

ВИСНОВОК……….……….……….……….……….……….……….……….……….……….…25

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ……….……….……….……….……….……….……..26

Додаток А……….……….……….……….……….……….……….……….……….……….……27

Додаток Б……….……….……….……….……….……….……….……….……….……….…….32

Додаток В…………………………………………………………………………………………..43

ВСТУП

Довга арифметика щодня стає все більш актуальною. За допомогою довгих чисел в криптографії кодується інформація, знаходяться прості числа. Необхідна вона також в фінансових та математичних операціях.

На сьогоднішній день головними арифметичними операціями є додавання, віднімання, множення та ділення.

Оскільки в мовах програмування числа стандартних типів є обмеженими, то існує необхідність в нових типах, що зможуть зберігати в собі великі числа (наприклад, числа більші ніж 1020, тобто розрядність більше 20).

Метою даної курсової є створення програми, що здатна буде здійснювати операції з числами, розряд яких більше 25. Інтерфейс розроблений таким чином, щоб програмою могли користуватись звичайні користувачі.

Програма написана на мові програмування С# за допомогою технології WPF в інтегрованому середовищі розробки MS Visual Studio 2013.

1 ПОСТАНОВКА ТА ІНФОРМАЦІЙНА СУТНІСТЬ ЗАВДАННЯ

Завданням курсової роботи є розроблення та реалізація алгоритму додавання, віднімання, множення та ділення двадцятип’ятизначних чисел.

Функціональна частина повинна відповідати таким вимогам:

* швидкий та зручний запуск програми;
* обробка виняткових ситуацій під час виконання програми;
* стійкість до введення непередбачених даних (невідповідність типів, помилкове натиснення кнопок на клавіатурі і т.д.);
* стабільна робота програми після коректного запуску (без зависань, злетів, виведення повідомлень невідомого характеру та змісту тощо);

Інтерфейсна частина програми повинна відповідати таким вимогам:

* дружність інтерфейсу (лаконічні та інформативні повідомлення), що прискорює розуміння правил користування програмою;
* мінімізація об'єктів управління програмою;
* відображення усіх даних із назвою, знаком та у зручному форматі;
* можливість використання елементів допомоги.

2 ТЕОРЕТИЧНІ ВІДОМОСТІ ПРО АРИФМЕТИКУ ДОВГИХ ЧИСЕЛ ТА

АЛГОРИТМИ ОПЕРАЦІЙ З НИМИ

2.1 Теоретичні відомості про арифметику довгих чисел

Довга арифметика — в обчислювальній техніці операції над числами, розрядність яких перевищує довжину машинного слова даної обчислювальної машини. По суті арифметика з великими числами являє собою набір алгоритмів виконання базових операцій (додавання, множення, зведення в степінь …) над числами, реалізованими не апаратно, а програмно, використовуючи більш базові апаратні засоби роботи з числами менших порядків [1].

Арифметика довгих чисел перш за все використовується у:

* Комп'ютерах низької розрядності, мікроконтролери. Наприклад, мікроконтролери серії AVR мають 8-бітний регістр і 10-бітний Аналогово-цифровий перетворювач (АЦП) — так що при обробці інформації з АЦП без довгої арифметики не обійтися.
* Криптографії. Зокрема довга арифметика застосовується в алгоритмах авторизації по відкритому ключу — таких, як RSA і Diffie-Hellman. З метою запобігання відомих атак, розміри чисел повинні перевищувати 10309 . Втім досить поширені мови програмування, такі як ISO C і JAVA, вміють працювати тільки з числами, заданої десятковій точності. Зокрема для C99 максимальна довжина вбудованого типу даних long не перевищує число 1019.
* Математичному та фінансовому програмному забезпеченні (ПЗ), яке вимагає, щоб результат обчислення на комп'ютері збігся до останнього розряду з результатом обчислення на папері. Зокрема, калькулятор Windows (починаючи з Windows 95).
* Числах з плаваючою комою. У комп'ютерах числа з плаваючою комою представлені у вигляді n = s \* m \* bx, де n — саме число, s — знак числа, m — мантиса, b — підстава показової функції, x — показник степеня. При обробці чисел підвищеної точності, розміри мантиси можуть перевищити апаратно допустимі норми. У цих випадках для роботи з мантисами можна використовувати алгоритми роботи з великими числами.

У мовах програмування завжди існує кілька стандартних типів змінних, які представляють ціле число. Однак, у більшості мов програмування не знайдеться типу даних, який працював би з великими числами [5]. Наприклад, тип даних decimal в мові програмування C# працює тільки з числами від  - 79228162514264337593543950335 до 79228162514264337593543950335. Саме існує гостра необхідність в реалізації типу даних, що зможе працювати з числами більшими, ніж ті, що були наведені в прикладі [2].

2.2 Алгоритми операцій з довгими числами

Додавання та віднімання. При додаванні, як і при відніманні довгих чисел найчастіше використовують алгоритм додавання «в стовпчик», тобто додають (віднімають) всі цифри з права на ліво. Якщо сума цифр у випадку додавання виходить більше 10, то до наступної цифри додають 1, а у відповідь зносять тільки одиниці від суми. У випадку віднімання, якщо різниця менше одиниці, то у наступної цифри віднімають 1, а до різниці додають 10 і зносять результат у відповідь.

Множення. Множення довгих чисел може здійснюватися за декількома алгоритмами: «в стовпчик», алгоритм Карацуби, алгоритм Тоома.

Найпростішим є алгоритм множення «в стовпчик». Суть його полягає у тому, щоб перемножити «верхнє» (більше) число на цифри з меншого числа, а потім додавати утворені від перемножувань числа. Сума і буде результатом множення.

Метод множення Карацуби відноситься до парадигми «розділяй і володарюй». Даний алгоритм являє собою просту реалізацію ідеї поділу вхідних даних, яка стала базисною для нижчеперелічених алгоритмів. Ідея полягає в поділі однієї операції множення над n-значними числами на три операції множення над числами довжини n / 2 плюс O (n) [3].

Для перемноження двох чисел, що перевищують довжину машинного слова, алгоритм Карацуби викликається рекурсивно доти, поки ці числа не стануть достатньо маленькими, щоб їх можна було перемножити безпосередньо.

Алгоритм Тоома є узагальненням алгоритму Карацуби і працює швидше. Для двох даних цілих чисел A і B алгоритм Тоомa ділить їх на k менших частин, довжина кожної з яких дорівнює довжині машинного слова, і виробляє операції над цими частинами. Складність обчислення алгоритму: O(n^{ln(2k-1)/ln(k)}) [4].

Ділення. Основними алгоритмами ділення довгих чисел є метод ділення «в стовпчик» («куточком») та за допомогою алгоритму Кнута [6].

Суть ділення «в стовпчик» полягає у діленні простих або складних багатозначних чисел за рахунок розбивання поділу на ряд більш простих кроків.

Алгоритм ділення Кнута:

По даним невід’ємним цілим числам  і , представлених за основою *b*, якщо і, знаходимо в системі числення за основою *b*, частку і залишок від частки .

3 ВИБІР ТА ОПИС АЛГОРИТМІВ, РЕАЛІЗОВАНИХ У ПРОГРАМІ

3.1 Вибір алгоритму для роботи програми

У створеній програмі для додавання та віднімання довгих чисел було обрано алгоритм операцій з ними «в стовпчик», тому що він є досить ефективним, простим та зрозумілим у реалізації.

Алгоритм множення схожий на алгоритм множення «в стовпчик», відмінність полягає у тому, що довге число ми розбиваємо на масив менших чисел (типу int). Це покращує ефективність виконання операції, оскільки складність операції зменшується в декілька разів, порівняно зі звичайним множенням «в стовпчик», де перемножується кожна цифра числа.

В якості алгоритму для операції ділення довгих чисел було обрано алгоритм ділення Кнута, оскільки серед основних алгоритмів ділення таких чисел він є найбільш швидким та ефективним.

3.2 Опис алгоритмів, реалізованих у програмі

Додавання.

Крок 1. Отримання числа, яке потрібно додати до існуючого.

Крок 2. В циклі від *і=0* до *і=n*, де *n* – довжина більшого числа:

А. Отримання значення *і-*го розряду першого числа.

Б. Отримання значення *і-*го розряду другого числа.

В. Сумування отриманих значень та значення переносу.

Г. Додавання значення суми в список.

Д. Якщо результат більше 9, то:

І. Відняти від доданого в список елемента 10.

ІІ. Присвоїти значення переносу одиницю.

Е. Інакше значення переносу нуль.

Крок 3. Якщо значення переносу рівне 1, то додати його в список.

Крок 4. Формуємо зі списку число типу LongInteger і повертаємо його.

Віднімання.

Крок 1. Отримання число, яке потрібно додати до існуючого.

Крок 2. В циклі від *і=0* до *і=n*, де *n* – довжина більшого числа:

А. Отримання значення *і*-го розряду першого числа.

Б. Отримання значення *і*-го розряду другого числа.

В. Віднімання отриманих значень та значення переносу.

Г. Додавання значення різниці в список.

Д. Якщо результат віднімання менше 0, то:

І. Додати до доданого в список елемента 10.

ІІ. Присвоїти значення переносу одиницю.

Е. Інакше значення переносу нуль.

Крок 4. Формуємо зі списку число типу LongInteger і повертаємо його.

Множення.

Крок 1: Розбиття довгого числа на int-масив менших чисел.

Крок 2: Проходження циклом по масиву першого числа:

А. Взяти елемент масиву першого числа і в циклі проходження елементів іншого числа:

І. Множення елемента першого числа на елемент іншого числа і додавання змінної значення переносу *k*.

II. Додавання остачі від ділення отриманого результату на 108 (тому що велике число ділиться на масив менших 8-значних чисел) в список (List).

ІІІ. Присвоєння змінній *k* значення результату ділення націло результату множення з кроку І на 108.

Б. Додавання *k* в список, та видаляємо 0, що йдуть на початку списку (якщо такі є). Формування тимчасову змінну число типу LongInteger.

В. Додавання на початок числа-відповіді кількість нулів, рівну числу ітерацій циклу.

Г. Сумування числа-відповіді і тимчасову змінну з Б.

Крок 3: Повертання числа-відповіді типу LongInteger.

Ділення.

*u = u[m+n-1]..u[0]* – ділене за основою *b*

*v = v[n-1]..v[0]* — дільник за основою *b*

*[u/v] = q[m]..q[0]* — частка

*u mod v = r[n-1]..r[0]* — остача

Крок 1: Нормалізація. Нам необхідно перетворити ділене і дільник, помноживши на коефіцієнт d:

А. Множимо *d = [b / (v [n-1] +1)].*

Б. Якщо d = 1, то додаємо нульовий розряд *u [m + n] = 0.*

Крок 2: Початкова ініціалізація *j = m.*

Крок 3: Обчислити тимчасове q. *tempq = [(u [j + n] \* b + u [j + n-1]) / v [n-1]],* *tempr* - залишок від такого поділу.

Крок 4: Поки *tempr = b*, то виконуємо перевірку:

А. Якщо *tempq=b*, або ж *q\*v[n-2]>b\*tempr + u[j+n-2]*, то:

І. *tempq = tempq - 1*

ІІ. *tempr = tempr + v[n-1]*

Крок 5: Помножити і відняти. *u [j + n] .. u [j] = u [j + n] .. u [j] - tempq \* v [n-1] .. v [0]*

Крок 6: Перевірка залишку. *q [j] = tempq*.

А. Якщо результат кроку 5 від’ємний, то переходимо до кроку 7.

Б. Інакше, переходимо до кроку 8.

Крок 7: Компенсуємо додавання. *q[j] = q[j] — 1. u[j+n]..u[j] = u[j+n]..u[j] + v[n-1]..v[0]*

Крок 8: Декрементуємо j. Поки *j>=3*, то повертаємося до кроку 3.

Крок 9: Повертаємо відповідь *q[m]..q[0]*

Алгоритми операцій у вигляді блок-схем наведено у додатку А:

Блок-схема А.1 - алгоритм додавання.

Блок-схема А.2 - алгоритм віднімання.

Блок-схема А.3 – алгоритм ділення.

Блок-схема А.4 – алгоритм множення.

4 ОПИС ПРОГРАМИ

4.1 Опис методів та подій

В програмному коді програми для додавання великих чисел реалізовано такі методи та події:

* Подія, що відбувається при натисканні кнопки «+» (додати).
* Подія, що відбувається при натисканні кнопки «-» (відняти).
* Подія, що відбувається при натисканні кнопки «х» (помножити).
* Подія, що відбувається при натисканні кнопки «/» (поділити).
* Подія, що відбувається при натисканні кнопки «Скопіювати».
* Подія, що відбувається при натисканні кнопки «About».
* Подія, що відбувається при натисканні кнопки «Інструкції».
* Подія, що відбувається при додаванні символа до першого текстового поля
* Подія, що відбувається при додаванні символа до другого текстового поля
* Метод, що додає два числа.
* Метод, що визначає знаки чисел при додаванні.
* Метод, що віднімає два числа.
* Метод, що визначає знаки двох чисел.
* Метод, що визначає знак результату при множенні числа типу int та довгого числа.
* Метод, що перемножує довге число та число типу int.
* Метод, що визначає знак результату при множенні двох довгих чисел.
* Метод, що перемножує два великих числа.
* Метод, що визначає знак результату при діленні великого числа на число типу int.
* Метод, що ділить велике число на число типу int.
* Метод, що визначає знак результату при діленні великих чисел.
* Метод, що ділить великі числа.
* Метод, що визначає чи є дільник типу int. Якщо так, то викликається метод, що ділить велике число на число типу int, а якщо ні, то викликається метод, що ділить великі числа.
* Метод, що видаляє 0, які йдуть на початку списку, який передали (так званий метод нормалізації)
* Метод, що формує рядок типу string з великого числа.
* Метод, який порівнює два великі числа за їхніми знаками. Якщо числа одного знаку, то викликається метод, що порівнює ці числа, взятих по модулю.
* Метод, який порівнює два великі числа, взятих по модулю.

4.2 Опис методів додатку

Метод Add(LongInteger value) отримує число яке потрібно додати до існуючого, перевіряє знаки доданків, визначає знак результату їх додавання та передає число value методу AddNumbers(LongInteger value).

Метод AddNumbers(LongInteger value) додає два довгих числа, взятих по модулю та повертає результат типу LongInteger.

Метод Substract(LongInteger value) отримує число яке потрібно відняти від існуючого, перевіряє знаки аргументів, визначає знак результату їх різниці та передає число value методу SubstractNumbers(LongInteger value).

Метод MultiplyNumbers(LongInteger value) отримує число на яке потрібно помножити та перевіряє чи value не може бути представлене типом integer. Якщо так то value передається в метод Multiply(int value). Інакше value передається в метод Multiply(LongInteger value).

Метод Multiply(int value) отримує число типу int на яке потрібно помножити, перевіряє знаки аргументів, визначає знак результату їх добутку, та передає число value методу MultiplyNumbers(int value). Метод повертає результат виконання методу MultiplyNumbers(int value) та присвоює необхідний знак.

Метод MultiplyNumbers(int value) отримує число типу int та множить його на число класу за алгоритмом, описаним в розділі №3. Результат повертає типу LongInteger в метод Multiply(int value).

Метод Multiply(LongInteger value) отримує число типу LongInteger на яке потрібно помножити, перевіряє знаки аргументів, визначає знак результату їх добутку, та передає число value методу MultiplyNumbers(LongInteger value). Метод повертає результат виконання методу MultiplyNumbers(LongInteger value) та присвоює необхідний знак.

Метод MultiplyNumbers(LongInteger value) отримує число типу LongInteger та множить його на число класу за алгоритмом, описаним в розділі №3. Результат повертає типу LongInteger в метод Multiply(LongInteger value).

Метод Div(LongInteger v) отримує число-дільник типу LongInteger та перевіряє чи не може v бути представлене числом типу int. Як так то *v* передається в метод Divide(int v, out int r), де *r* – це залишок від ділення. Інакше *value* передається в метод Divide(out LongInteger q, out LongInteger r, LongInteger u, LongInteger v).

Метод Divide(int v, out int r) отримує числа типу int, де *v* - це дільник, а *r* - це остача. Метод перевіряє знаки діленого та дільника та визначає знак результату. В методі викликається інший метод DivideNumbers(int v, out int r), який повертає результат ділення за модулем.

Метод DivideNumbers(int v, out int r) ділить модуль числа класу на модуль числа *v* типу int та повертає число типу LongInteger.

Метод Divide(out LongInteger q, out LongInteger r, LongInteger u, LongInteger v) отримує як параметри *u* та *v*, де *u* – ділене, а *v* – дільник. Повертає цей метод *q* (частку) та *r* (остачу). Цей метод перевіряє знаки діленого та дільника та визначає знак результату. В методі викликається інший метод DivideNumbers(out LongInteger q, out LongInteger r, LongInteger u, LongInteger v), який повертає результат ділення за модулем двох великих чисел.

Метод DivideNumbers(out LongInteger q, out LongInteger r, LongInteger u, LongInteger v) отримує як параметри *u* та *v*, де *u* – ділене, а *v* – дільник. Метод ділить *u* на *v* (*u* та *v* взяті по модулю) та повертає *q* (частку) та *r* (остачу) в метод Divide(out LongInteger q, out LongInteger r, LongInteger u, LongInteger v).

Метод ZeroNormalize(List<int> arr) отримує число, представлене списком та, якщо на початку є нулі, то вони видаляються. Метод повертає нормалізований List.

Метод ToString() перевизначений від класу SystemObject та представляє число типу LongInteger як рядок типу string.

Метод CompareTo(LongInteger other) порівнює великі числа за їхніми знаками. Якщо ж знаки співпадають то викликається метод NumbersCompareTo(LongInteger other), який порівнює числа за їхніми значеннями. Може повертати 1, -1 та 0 (якщо числа рівні).

Метод NumbersCompareTo(LongInteger other) порівнює великі числа, взяті по модулю, за їхніми значеннями. Може повертати 1, -1 та 0 (якщо числа по модулю рівні).

4.3 Події у програмі

При запуску програми можна ввести два великих числа та здійснити їх додавання, віднімання, множення або ділення за допомогою відповідних кнопок та подій, що пов’язаних із цими кнопками. Далі наведений список методів, що є пов’язаними з подіями в програмі:

* Plus\_Click(object sender, RoutedEventArgs e) – відбувається при натисканні кнопки «+». Викликається метод Add(LongInteger value) класу LongInteger який додає два числа, що записані в textBox1 та textBox2. Результат приводиться до типу string за допомогою методу ToString та записується в ResulTextBox.
* Minus\_Click(object sender, RoutedEventArgs e) – відбувається при натисканні кнопки «-». Викликається метод Substract(LongInteger value) класу LongInteger який знаходить різницю чисел, що записані в textBox1 та textBox2. Результат приводиться до типу string за допомогою методу ToString та записується в ResulTextBox.
* Multiple\_Click(object sender, RoutedEventArgs e) – відбувається при натисканні кнопки «X». Викликається метод Multiply(LongInteger value) класу LongInteger який знаходить добуток чисел, що записані в textBox1 та textBox2. Результат приводиться до типу string за допомогою методу ToString та записується в ResulTextBox.
* Divide\_Click(object sender, RoutedEventArgs e) – відбувається при натисканні кнопки «X». Викликається метод Div(LongInteger value) класу LongInteger який націло ділить число з textBox1 на число з textBox2. Результат приводиться до типу string за допомогою методу ToString та записується в ResulTextBox.
* CopyResultButton\_OnClick(object sender, RoutedEventArgs e) – натисканні кнопки «Скопіювати» відбувається копіювання результату з ResulTextBox в буфер обміну Windows для подальшого використання результату в сторонніх програмах та додатках.
* NumberValidationTextBox1(object sender, TextCompositionEventArgs e) – відбувається при вводі символа в поле вводу textBox1. При цьому виконується перевірка на правильність вводу, тобто якщо користувач введе недопустимий символ, то цей символ не додається в textBox1.
* NumberValidationTextBox2(object sender, TextCompositionEventArgs e) – відбувається при вводі символа в поле вводу textBox2. При цьому виконується перевірка на правильність вводу, тобто якщо користувач введе недопустимий символ, то цей символ не додається в textBox2.
* AboutButton\_Click(object sender, RoutedEventArgs e) – відбувається при натисканні на кнопку «About». В результаті відкриваться нове вікно класу About.
* InstrButton\_Click(object sender, RoutedEventArgs e) – відбувається при натисканні на кнопку «Інструкції». В результаті відкриваться нове вікно класу Instructions.

4.4 Опис класів у програмі

Клас MainWindow – клас, що пов’язаний з MainWindow.xaml, та є головним вікном програми.

Містить методи для кнопок операцій з великими числами.

Метод для кнопки «Скопіювати» – CopyResultButton\_OnClick(object sender, RoutedEventArgs e).

Метод для кнопки «About» – AboutButton\_Click(object sender, RoutedEventArgs e).

Метод для кнопки «Інструкції» – InstrButton\_Click(object sender, RoutedEventArgs e).

Клас About – клас, що виводить інформацію про програму, її версію та інформацію про розробника.

Клас Instructions – клас, що виводить коротку інформацію про те, як потрібно користуватись програмою для операцій з довгими числами.

Клас LongInteger – клас, в якому зберігається довге число, яке вводить користувач. Саме в цьому класі реалізовані методи для роботи з великими числами (див 3.4).

5 ОПИС ІНТЕРФЕЙСУ ПРОГРАМИ ТА КЕРІВНИЦТВО ДЛЯ КОРИСТУВАЧА

5.1 Опис інтерфейсу

Інтерфейс програми складається з одного головного вікна та двох допоміжних:

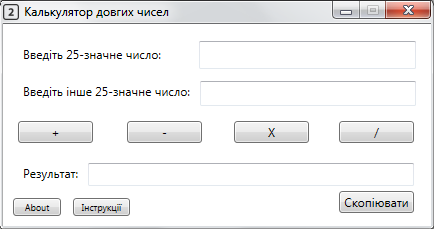


Рисунок 5.1 – Інтерфейс головного вікна програми

Головне вікно включає в себе такі елементи управління:

1. Поле вводу першого числа.
2. Поле вводу другого числа.
3. Кнопка для додавання, при натисненні на яку числа, що введені в поля вводу додаються.
4. Кнопка для віднімання, при натисненні на яку знаходиться різниця чисел, що введені в поля вводу.
5. Кнопка для знаходження добутку введених чисел.
6. Кнопка, що ділить число з першого поля вводу на число з другого поля вводу.
7. Поле для виводу результату операції над великими числами. Дане поле є ReadOnly, тому запис в нього є неможливим.
8. Кнопка для копіювання результатів з поля виводу результату в буфер обміну Windows для подальшого використання в сторонніх програмах.
9. Кнопка для відкриття вікна About.
10. Кнопка для відкриття вікна інструкції по користуванню програмою.

Окрім головного вікна в програмі ще є вікно About. В ньому відображається вся основна інформація про програму, а саме: її назва, версія, ім’я розробника та його контакти.

Також, в програмі є допоміжне вікно Instructions, в якому викладений короткий HELP по користуванню програмою.

5.2 Інструкція по користуванню програмою для користувача

Після запуску програми, відкривається головне вікно MainWindow (Рисунок 5.1). Воно включає в себе весь необхідний функціонал для роботи з великими числами, а саме можливість їх додавання, віднімання, множення та ділення.

Поля для вводу призначені для того, щоб користувач міг ввести числа, які потрібно порахувати. Алгоритм введення символів з клавіатури в програмі реалізований таким чином, що користувач може вводити тільки цифри. Знак «-» можна ввести тільки на початку числа.

Далі потрібно натиснути ту кнопку, яка позначає бажану операцію над введеними числами. У випадку, якщо поля для вводу пусті, то програма продовжує стабільно працювати і не припиняє свою роботу.

Після натискання на потрібну кнопку, в результаті роботи алгоритму, у поле виводу інформації записується число-відповідь. Користувач не може внести зміни у це число оскільки у поля вказаний атрибут ReadOnly і запис в нього не є можливим.

Для того, щоб відповідь можна було використовувати в інших програмах або додатках операційної системи Windows, було реалізовано функцію збереження результату в буфер обміну. Таким чином, скопіювавши результат, можна з легкістю користуватися ним будь-де.

У випадку, якщо користувач хоче швидко отримати якусь інформацію про функціонал програми, то він може переглянути HELP-вікно, натиснувши в головному вікні на кнопку «Інструкції».

У вікні About користувач може переглянути версію програми та дані про розробника. Якщо виникла непередбачувана помилка, або додаток має певні недоліки, то в цьому вікні користувач може дізнатися дані для зв’язку з розробником та повідомити йому про знайдені помилки. Також, можна запропонувати розробнику новий функціонал для майбутніх версій програми.

5.3 Інструкція для програміста

Створена програма називається «Калькулятор довгих чисел»

Програма написана на мові програмування С# з використанням WPF для створення графічного інтерфейсу.

Для її роботи необхідні такі бібліотеки як:

* PresentationCore
* PresentationFramework
* System.Core
* System.Data
* System.Xaml
* WindowBase

Також, обов’язково повинен бути встановлений .NET Framework версії не нижче 4.

Створювалася програма в IDE MS Visual Studio 2013.

Конфігурація ПК, на якій проектувалася програма:

* Процесор: Intel Core i3 3110m
* Графічний процесор: Nvidia GeForce 740m
* ОЗП: 4 Гб
* Операційна система: Windows 7 Ultimate x64

Для додавання операцій, що будуть виконуватись з довгими числами, необхідно редагувати та доповнювати клас LongInteger, тому що саме він містить в собі число та методи, що до нього відносяться. Для того, щоб змінити головне вікно потрібно редагувати файли MainWindow.xaml.cs та MainWindow.xaml (містить в собі код розмітки головного вікна).

Математичні алгоритми описані в даній курсовій роботі в розділі 3.2, а їх реалізацію можна знайти в додатку Б (вихідний код програми).

Компілювалася програма в режимі Debug з цільовою платформою Any CPU.

5.4 Системні вимоги програми

* Операційна система: (64-bit) Windows XP \ Windows Vista \ Windows 7 \ Windows 8 \ Windows 8.1
* ОЗП: 64 Mb
* Процесор: 1 Gh
* Вільний простір на вінчестері : 2 Мбайт
* Маніпулятор, клавіатура

ВИСНОВОК

В ході виконання курсової роботи були набуті навики розроблення та реалізації алгоритму для здійснення операцій з великими (двадцятип’ятизначними) числами.

При розробці програми були розглянуті різноманітні алгоритми для додавання, віднімання, множення та ділення великих чисел. Серед них було вибрано ті, які є оптимальними для вирішення поставленої задачі. Алгоритми було закодовано за допомогою мови програмування C# з використанням технології WPF.

У додатку було реалізовано алгоритм додавання, віднімання та множення «в стовпчик» з деякими змінами для зменшення складності алгоритму та збільшення швидкості виконання операцій. Ділення двох чисел реалізовано за допомогою алгоритму Дональда Кнута.

Окремо проводилося тестування програми на екстремальні ситуації, такі як неправильне значення введених змінних, або ж невідповідність типу введеної змінної.

Серед переваг розробленої програми можна відмітити наступні:

* Зручність.
* Інтуїтивна зрозумілість інтерфейсу.
* Функціонал.
* Легкість та оптимізація програми.

Додаток, створений у ході даної курсової роботи відповідає всім вимогам, поставленим перед ним та вирішує необхідні задачі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. «Длинная арифметика» [Електронний ресурс] Режим доступу:­ http://ru.wikipedia.org/wiki/Длинная\_арифметика
2. «Встроенные типы данных» [Електронний ресурс] Режим доступу:­ <https://msdn.microsoft.com/ru-ru/library/cs7y5x0x(v=vs.90).aspx>
3. «Умножение Карацубы» [Електронний ресурс] Режим доступу:­ <https://ru.wikipedia.org/wiki/Умножение_Карацубы>
4. «Алгоритм Тоома – Кука. Арифметические вычисления с высокой точностью» [Електронний ресурс] Режим доступу:­ <http://www.termist.com/bibliot/stud/wetherell/127.htm>
5. Герберт Ш. C# 4.0. Полное руководство [Текст] / Ш. Герберт, — М. : Издательский дом «Вильямс», 2011. — 1056 с.
6. Кнут Дональд Эрвин. Искусство программирования, том 2. Получисленные алгоритмы, 3-е издание.: Перевод с английского: Учебное пособие. - М.: Издательский дом «Вильямс», 2001. - 832 с.

ДОДАТКИ

Додаток А

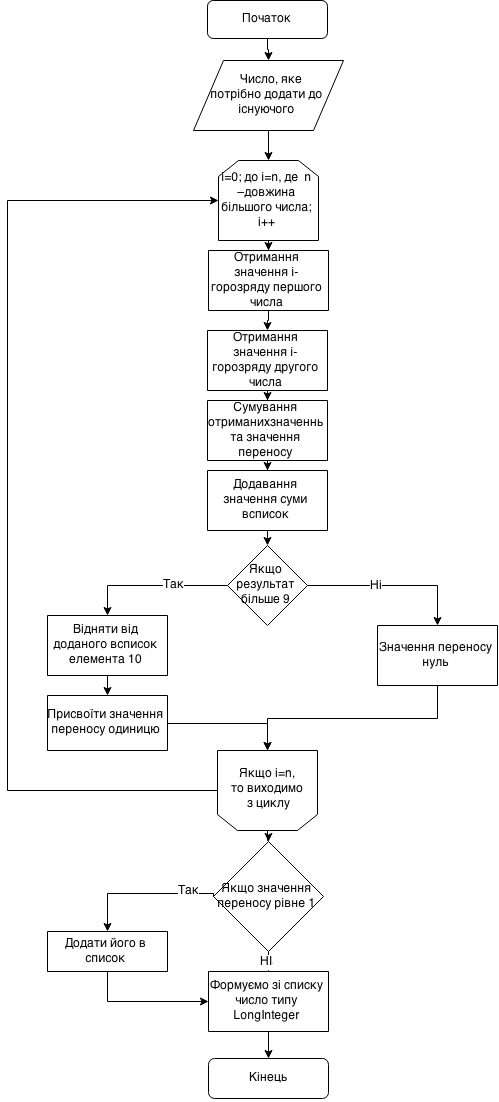


Рисунок А.1 – Блок-схема алгортиму додавання великих чисел

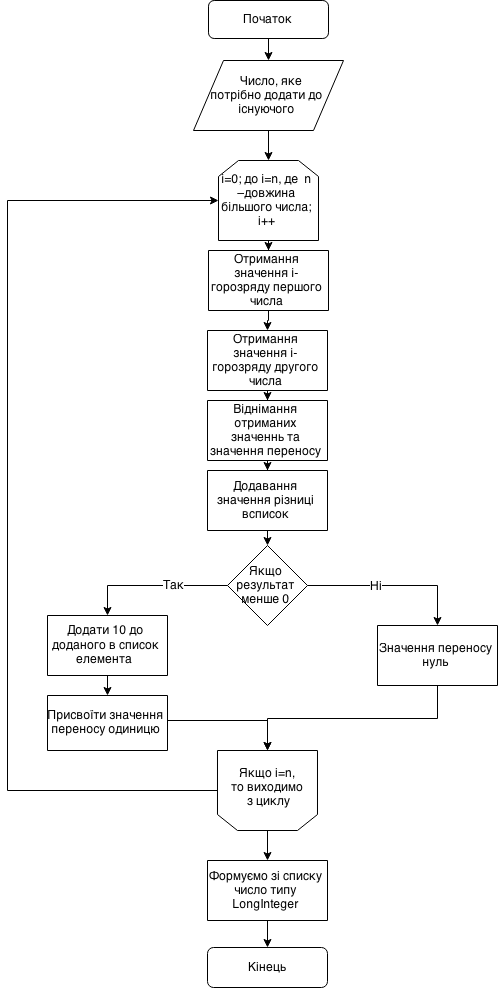
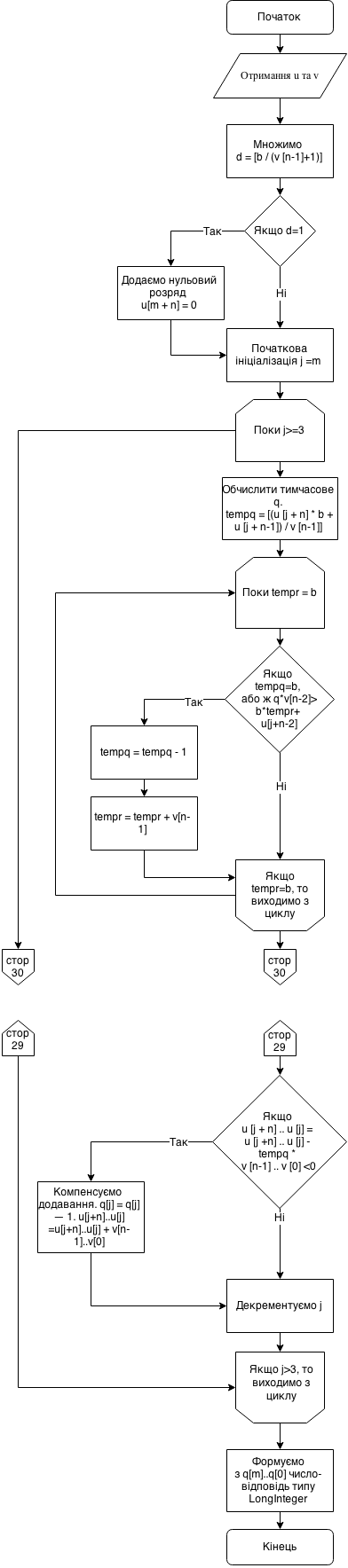


Рисунок А.2 – Блок-схема алгоритму віднімання



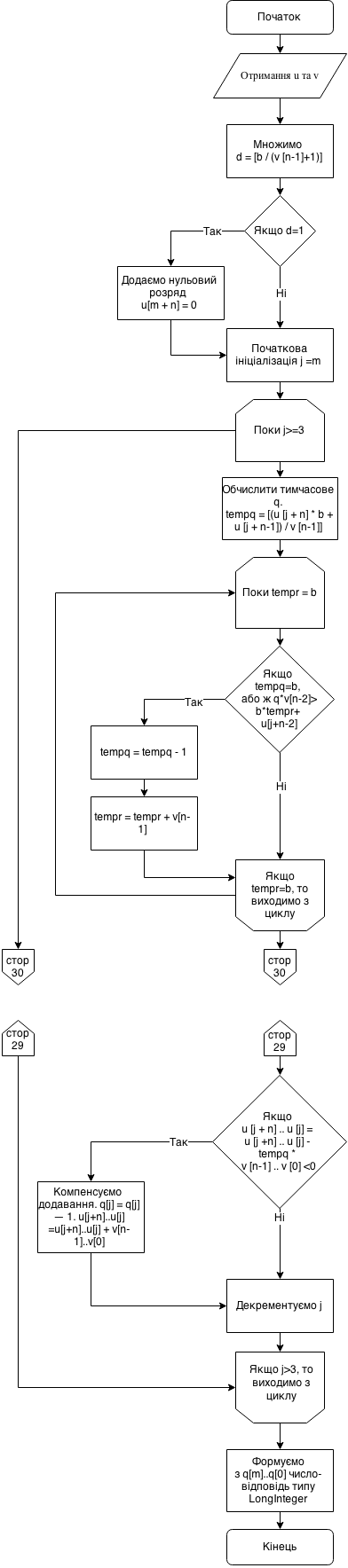


Рисунок А.3 – Блок-схема алгоритму ділення

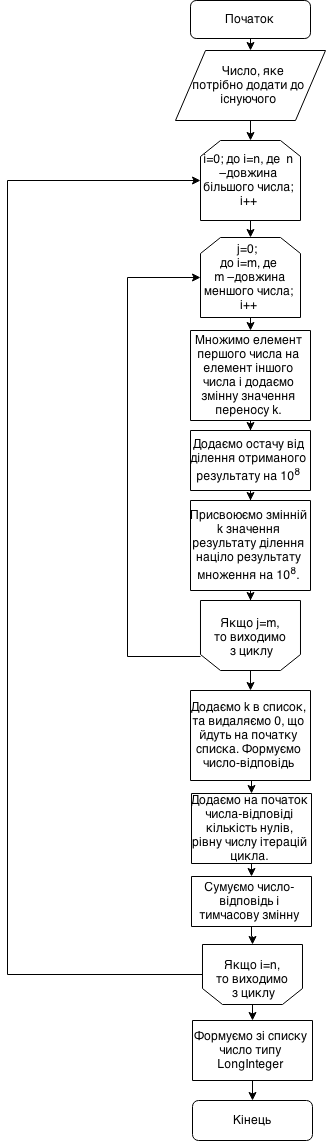


Рисунок А.4 – Блок-схема алгоритма множеня

Додаток Б

Вихідний код програми

Вихідний код вікна запуску:

using System;

using System.Collections.Generic;

using System.Data.Odbc;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using BigInteger;

namespace kursova

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private LongInteger \_result;

private void Plus\_Click(object sender, RoutedEventArgs e)//відбувається при натисканні кнопки «+»

{

if (string.IsNullOrEmpty(textBox1.Text) || string.IsNullOrEmpty(textBox2.Text)) return;

\_result = new LongInteger(textBox1.Text).Add(new LongInteger(textBox2.Text));

ResulTextBox.Text = \_result.ToString();

}

private void Minus\_Click(object sender, RoutedEventArgs e)//відбувається при натисканні кнопки «-»

{

if (string.IsNullOrEmpty(textBox1.Text) || string.IsNullOrEmpty(textBox2.Text)) return;

\_result = new LongInteger(textBox1.Text).Substract(new LongInteger(textBox2.Text));

ResulTextBox.Text = \_result.ToString();

}

private void Multiple\_Click(object sender, RoutedEventArgs e)//відбувається при натисканні кнопки «x»

{

if (string.IsNullOrEmpty(textBox1.Text) || string.IsNullOrEmpty(textBox2.Text)) return;

\_result = new LongInteger(textBox1.Text).Multiply(new LongInteger(textBox2.Text));

ResulTextBox.Text = \_result.ToString();

}

private void Divide\_Click(object sender, RoutedEventArgs e)//відбувається при натисканні кнопки «/»

{

if (string.IsNullOrEmpty(textBox1.Text) || string.IsNullOrEmpty(textBox2.Text)) return;

\_result = new LongInteger(textBox1.Text).Div(new LongInteger(textBox2.Text));

ResulTextBox.Text = \_result.ToString();

}

private void CopyResultButton\_OnClick(object sender, RoutedEventArgs e)//відбувається при натисканні кнопки «Скопіювати»

{

if (!string.IsNullOrEmpty(ResulTextBox.Text))

{

Clipboard.SetText(ResulTextBox.Text);

}

else

{

MessageBox.Show("Поле відповіді порожнє!");

}

}

private void NumberValidationTextBox1(object sender, TextCompositionEventArgs e)//відбувається при додаванні символа до textBox1

{

var regex = new Regex("[^0-9]+");

if (textBox1.Text == string.Empty && e.Text == "-")

{

e.Handled = false;

return;

}

e.Handled = regex.IsMatch(e.Text);

}

private void NumberValidationTextBox2(object sender, TextCompositionEventArgs e)//відбувається при додаванні символа до textBox2

{

var regex = new Regex("[^0-9]+");

if (textBox2.Text == string.Empty && e.Text == "-")

{

e.Handled = false;

return;

}

e.Handled = regex.IsMatch(e.Text);

}

private void AboutButton\_Click(object sender, RoutedEventArgs e)//відбувається при натисканні кнопки «About».

{

About about = new About();

about.Show();

}

private void InstrButton\_Click(object sender, RoutedEventArgs e)//відбувається при натисканні кнопки «Інструкції».

{

Instructions instructions = new Instructions();

instructions.Show();

}

}

Вихідний код класу LongInteger:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace BigInteger

{

class LongInteger

{

private List<int> arr = new List<int>(); //Лист, що зберігає поцифрово велике число

private const int order = 8; // Основа; по скільки розрядів ділиться велике число

private static readonly int ownBase = (int)Math.Pow(10, 8);//10 в степені основи

public static LongInteger Zero = new LongInteger("0");//Нуль

public static LongInteger One = new LongInteger("1");//Одиниця

private bool sign = true; //Знак числа (true - додатнє)

#region Конструктори

private void ChangeSign(bool flag)

{

sign = flag; //Зміна знака числа на вказаний

}

public LongInteger(string s)

{

int tempValue = 0;

int tempOrder = 0;

sign = true;

for (int i = s.Length - 1; i >= 0; i--)

{

if (i == 0 && s[0] == '-')

{

sign = false;

}

else

{

if (tempOrder == order)

{

arr.Add(tempValue);

tempValue = int.Parse(s[i].ToString());

tempOrder = 1;

}

else

{

tempValue = tempValue + (int)Math.Pow(10, tempOrder) \* int.Parse(s[i].ToString());

tempOrder++;

}

}

}

if (tempOrder != 0)

{

arr.Add(tempValue);

}

}

private LongInteger(int[] arr, bool sign)

{

for (int i = 0; i < arr.Length; i++)

{

this.arr.Add(arr[i]);

}

this.sign = sign;

}

private LongInteger(List<int> arr, bool sign)

{

this.arr = arr;

this.sign = sign;

}

#endregion

#region Додавання

private LongInteger AddNumbers(LongInteger value)//Метод для додавання великих чисел

{

int k = 0; //значення переноса

int n = Math.Max(arr.Count, value.arr.Count); //максимальная довжина числа

var ans = new List<int>(); //масив, в який записується результат

for (int i = 0; i < n; i++)

{

int tempA = (arr.Count > i) ? arr[i] : 0; //значення і-го розряда першого числа

int tempB = (value.arr.Count > i) ? value.arr[i] : 0; //значення і-го розряда другого числа

ans.Add(tempA + tempB + k);

if (ans[i] >= ownBase) //якщо результат більше ніж основа

{

ans[i] -= ownBase;

k = 1;

}

else

{

k = 0;

}

}

if (k == 1)

{

ans.Add(k); //додаємо в список якщо значення переносу залишилось 1

}

return new LongInteger(ans, true); //Повертаємо число типу LongInteger

}

public LongInteger Add(LongInteger value) //Перевіряємо знаки в числах, що будемо додавати

{

if (sign && value.sign) //обидва додатні

{

return AddNumbers(value);

}

if (!sign && value.sign) //різні знаки

{

return value.SubstractNumbers(this);

}

if (sign && !value.sign) //різні знаки

{

return SubstractNumbers(value);

}

LongInteger ans = AddNumbers(value); //додаємо по модулю

ans.ChangeSign(false); //змінюємо знак на від'ємний

return ans;

}

#endregion

#region Віднімання

private LongInteger SubstractNumbers(LongInteger value)//Метод для віднімання великих чисел

{

if (this.NumbersCompareTo(value) == -1) //Порівнюємо два числа

{

LongInteger temp = value.SubstractNumbers(this);

temp.ChangeSign(false);

return temp;

}

int k = 0;//змінна для переносу одиниці

int n = Math.Max(arr.Count, value.arr.Count);

List<int> ans = new List<int>();

for (int i = 0; i < n; i++) //Проходимося в циклі по числу

{

int tempA = (arr.Count > i) ? arr[i] : 0; //значення і-го розряду першого числа

int tempB = (value.arr.Count > i) ? value.arr[i] : 0; //значення і-го розряду другого числа

ans.Add(tempA - tempB - k); //віднімаємо

if (ans[i] < 0)

{

ans[i] += ownBase;//Додаємо основу

k = 1;//Вказуємо значення переносу

}

else

{

k = 0;

}

}

return new LongInteger(ZeroNormalize(ans), true);

}

public LongInteger Substract(LongInteger value)//Перевіряємо знаки в числах, що будемо віднімати

{

if (sign && value.sign) //обидва додатні

{

return SubstractNumbers(value);

}

if (!sign && value.sign) //різні знаки

{

LongInteger ans = AddNumbers(value);//віднімаємо по модулю

ans.ChangeSign(false); //змінюємо знак

return ans;

}

if (sign && !value.sign) //різні знаки

{

return AddNumbers(value);

}

return value.SubstractNumbers(this);

}

#endregion

#region Множення

private LongInteger Multiply(int value)//Метод, що визначає знак результату при множенні числа типу int та довгого числа.

{

LongInteger ans = MultiplyNumbers(Math.Abs(value));

if (sign && value >= 0)//Якщо обидва додатні

{

return ans;

}

if (!sign && value < 0)//Якщо обидва від'ємні

{

return ans;

}

ans.ChangeSign(false);//Якщо різних знаків то повертаємо від'ємний результат

return ans;

}

private LongInteger MultiplyNumbers(int value)//Метод, що перемножує довге число та число типу int

{

if (value >= ownBase)

{

throw new Exception("This value is bigger than base");

}

int k = 0;

List<int> ans = new List<int>();

for (int i = 0; i < arr.Count; i++)//проходимося по масиву довгого числа

{

long temp = (long)arr[i] \* (long)value + k;//перемножаємо і-й елемент на value і додаємо значення переносу k

ans.Add((int)(temp % ownBase));//додаємо результат в список

k = (int)(temp / ownBase);

}

ans.Add(k);//після проходження циклу додаємо значення переносу

return new LongInteger(ZeroNormalize(ans), true);//...та формуємо число типу LongInteger, попередньо видаливши нулі, що йдуть на початку

}

public LongInteger Multiply(LongInteger value)//Метод, що визначає знак результату при множенні двох довгих чисел

{

LongInteger ans = MultiplyNumbers(value);

if ((sign && value.sign) || (!sign && !value.sign))

{

return ans;

}

ans.ChangeSign(false);

return ans;

}

private LongInteger MultiplyNumbers(LongInteger value)//Метод, що перемножує два великих числа

{

LongInteger ans = new LongInteger("0");

for (int i = 0; i < arr.Count; i++)//проходимося по масиву більшого числа

{

LongInteger temp = value.MultiplyNumbers(arr[i]);//перемножаємо

for (int j = 0; j < i; j++)

{

temp.arr.Insert(0, 0);//додаємо нулі на поч. списку, рівні к-сті ітерацій

}

ans = ans.AddNumbers(temp);//додаємо результати перемножувань

}

return ans;

}

#endregion

#region Ділення

private LongInteger Divide(int v, out int r)//Метод, що визначає знак результату при діленні великого числа на чис-ло типу int.

{

if (v != 0)//якщо дільник не 0

{

LongInteger ans = this.DivideNumbers(Math.Abs(v), out r); //ділимо по модулю

if ((this.sign && v > 0) || (!this.sign && v < 0)) //якщо знаки однакові

{

return ans;

}

ans.ChangeSign(false); //інакше міняємо знаки

r = -r;

return ans;

}

throw new Exception("divide by zero");

}

private LongInteger DivideNumbers(int v, out int r)//Метод, що ділить велике число на число типу int.

{

int[] ans = new int[arr.Count];

r = 0;

int j = arr.Count - 1;//індекс останнього елемента масиву числа

while (j >= 0)//з кінця проходимось по числу

{

long cur = (long)(r) \* (long)(ownBase) + arr[j];

ans[j] = (int)(cur / v);

r = (int)(cur % v);

j--;

}

return new LongInteger(ans, true);//повертаємо результат дівлення

}

private void Divide(out LongInteger q, out LongInteger r, LongInteger u, LongInteger v)//Метод, що визначає знак результату при діленні великих чисел

{

LongInteger first = new LongInteger(u.ToString());//ділене

first.ChangeSign(true);

LongInteger second = new LongInteger(v.ToString());//дільник

second.ChangeSign(true);

DivideNumbers(out q, out r, first, second);//ділимо два числа по модулю

if ((u.sign && v.sign) || (!u.sign && !v.sign))//перевірка на знаки

{

return;

}

q.ChangeSign(!q.sign);//змінюємо знаки

r.ChangeSign(!r.sign);

}

private int DivideNumbers(out LongInteger q, out LongInteger r, LongInteger u, LongInteger v)//Метод, що ділить великі числа

{

//початкова ініціалізація

int n = v.arr.Count;

int m = u.arr.Count - v.arr.Count;

int[] tempArray = new int[m + 1];

tempArray[m] = 1;

q = new LongInteger(tempArray, true);

//Нормалізація

if (n == 1)

{

int tempr = 0;

q = u.Divide(v.arr[0], out tempr);

r = new LongInteger(tempr.ToString());

return 0;

}

int d = (ownBase / (v.arr[n - 1] + 1));

u = u.Multiply(d);

v = v.Multiply(d);

if (u.arr.Count == n + m)//перевірка на d==1

{

u.arr.Add(0);

}

int j = m;//Початкове присвоєння j

while (j >= 0)//Цикл по j

{

//Обчислюємо тимчасове q

long cur = (long)(u.arr[j + n]) \* (long)(ownBase) + u.arr[j + n - 1];

int tempq = (int)(cur / v.arr[n - 1]);

int tempr = (int)(cur % v.arr[n - 1]);

do

{

if (tempq == ownBase || (long)tempq \* (long)v.arr[n - 2] > (long)ownBase \* (long)tempr + u.arr[j + n - 2])

{

tempq--;

tempr += v.arr[n - 1];

}

else

{

break;

}

}

while (tempr < ownBase);

//Помножити і відняти

LongInteger u2 = new LongInteger(u.arr.GetRange(j, n + 1), true);

u2 = u2.Substract(v.Multiply(tempq));

bool flag = false;

if (!u2.sign)//якщо від'ємні

{

flag = true;

List<int> bn = new List<int>();

for (int i = 0; i <= n; i++)

{

bn.Add(0);

}

bn.Add(1);

u2.ChangeSign(true);

u2 = new LongInteger(bn, true).Substract(u2);

}

//Перевірка остачі

q.arr[j] = tempq;

if (flag)//компенсування множення

{

q.arr[j]--;

u2 = u2.Add(v);

if (u2.arr.Count > n + j)

u2.arr.RemoveAt(n + j);

}

//змінюємо u, так як всі операції проходять з його розрядами

for (int h = j; h < j + n; h++)

{

if (h - j >= u2.arr.Count)

{

u.arr[h] = 0;

}

else

{

u.arr[h] = u2.arr[h - j];

}

}

j--;

}

q.arr = ZeroNormalize(q.arr);

//Денормалізація

int unusedR = 0;

r = new LongInteger(u.arr.GetRange(0, n), true).Divide(d, out unusedR);

return 0;

}

public LongInteger Div(LongInteger v)//Метод, що визначає чи є дільник типу int

{

LongInteger q;

LongInteger r;

if (NumbersCompareTo(v) > -1)//порівнюємо число з отриманим v

{

if (v.arr.Count == 1)//Якщо довжина масиву v 1...

{

int tempr;

return Divide(v.arr[0], out tempr);//...то v є типу int

}

Divide(out q, out r, this, v);//...інакше ділимо як два великі числа

}

else

{

return Zero;//Інакше повертаємо 0

}

return q;

}

#endregion

#region Додаткові методи

private List<int> ZeroNormalize(List<int> arr)//Видаляємо 0 що йдуть спочатку (Нормалізація)

{

while (arr.Count > 0 && arr[arr.Count - 1] == 0)

{

arr.RemoveAt(arr.Count - 1);

}

return arr;

}

public override string ToString() //Метод для виводу числа типу LongInteger. Представляється у вигляді string

{

StringBuilder ans = new StringBuilder();

if (!sign)

{

ans.Append('-');//Якщо число від'ємне то на початку "-"

}

ans.Append(arr[arr.Count - 1].ToString());

for (int i = arr.Count - 2; i >= 0; i--)//проходимося по масиву числа та формуємо string відповіді

{

string temp = arr[i].ToString();

int zeroCount = order - temp.Length;

for (int j = 0; j < zeroCount; j++)//додаємо 0 на кінець числа

{

ans.Append('0');

}

ans.Append(temp);

}

return ans.ToString();

}

public int CompareTo(LongInteger other)

{

if (sign && other.sign)//Обидва числа є додатніми

{

return NumbersCompareTo(other);

}

if (sign && !other.sign)//Перше число додатнє

{

return 1;

}

if (!sign && other.sign)//Друге додатнє

{

return -1;

}

return -1 \* NumbersCompareTo(other);//Обидва числа є від'ємними

}

private int NumbersCompareTo(LongInteger other)

{

int ans = 0;

if (arr.Count == other.arr.Count) //Якщо однакова к-сть елементів у масиві чисел

{

for (int i = 0; i < arr.Count; i++)//проходимося по числу

{

if (arr[i] > other.arr[i])

{

ans = 1;

}

if (arr[i] < other.arr[i])

{

ans = -1;

}

}

}

else

{

if (arr.Count > other.arr.Count)

{

ans = 1;

}

if (arr.Count < other.arr.Count)

{

ans = -1;

}

}

return ans;

}

#endregion

}

}

Вихідний код вікна About:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace kursach

{

public partial class About : Window

{

public About()

{

InitializeComponent();

TextBlock.Text = "Версія програми: 1.0.0.1\n" +

"Розробник: Говоровський Сергій\n" +

"Контакти розробника:\n\t" +

"e-mail - [longinteger@program.com](mailto:longinteger@program.com)"; //Текст вікна About

}

}

}

Вихідний код вікна Інструкції:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace kursach

{

public partial class Instructions : Window

{

public Instructions()

{

InitializeComponent();

TextBlock.Text = "Програма призначена для виконання операцій над довгими числами\n" +

"Спочатку потрібно ввести числа в текстові поля. \n" +

"Далі необхідно натиснути ту кнопку, що позначає бажану операцію над числами. У полі для виводу з'явиться число-відповідь.\n" +

"Ви можете скопіювати відповідь у буфер обміну, натисеувши на кнопку 'Скопіювати'."; //Текст вікна Instructions

}

}

}

Додаток В

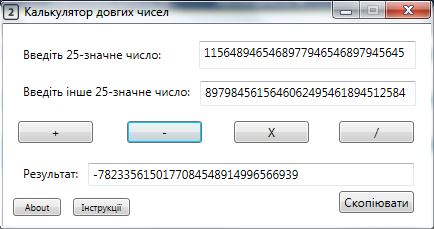


Рисунок В.1 – Приклад використання програми