**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**«ПОЛТАВСЬКА ПОЛІТЕХНІКА ІМЕНІ ЮРІЯ КОНДРАТЮКА»**

**НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ**

**ТЕХНОЛОГІЙ ТА РОБОТОТЕХНІКИ**

**КАФЕДРА КОМП’ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І**

**СИСТЕМ**

**Розрахунково-графічна робота**

**з дисципліни**

**«Об’єктно-орієнтоване програмування»**

**спеціальність 122 «Комп’ютерна інженерія»**

**Студента групи 203-ТК Нонки Івана Андрійовича**

Керівник роботи

старший викладач кафедри комп’ютерних та інформаційних технологій і систем  
Демиденко М.І.

Полтава 2024

**ЗМІСТ**

[**ВСТУП**](#_Toc135037888) 3

[**РОЗДІЛ 1 ПОСТАНОВКА ЗАДАЧІ**](#_Toc135037889) 5

[**1.1** **Мова програмування C#**](#_Toc135037890) 5

[**1.2 Вступ до роботи**](#_Toc135037891) 6

[**1.3** **Реалізація**](#_Toc135037892) 7

[**РОЗДІЛ 2 ОПИС ФУНКЦІОНАЛУ КЛАСІВ**](#_Toc135037898) 8

[**2.1 Класи основного проекту Windows Forms**](#_Toc135037899) 8

[**РОЗДІЛ 3 ДИЗАЙН КОРИСТУВАЦЬКОГО ІНТЕРФЕЙСУ**](#_Toc135037901) 10

[**3.1 Основна форма**](#_Toc135037902) 10

[**РОЗДІЛ 4 ВИХІДНИЙ КОД ПРОЕКТУ**](#_Toc135037908) 13

[**4.1 Код основної форми**](#_Toc135037909) 13

[**4.2 Код класу Program.cs**](#_Toc135037910) 14

[**4.3 Код Class1.cs**](#_Toc135037911) 15

[**ВИСНОВКИ** 16](#_Toc135037918)

[**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ** 18](#_Toc135037919)

# **ВСТУП**

**Об'є́ктно-орієнто́ване програмува́ння**[[1]](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F#cite_note-кочерга-мейнарович-1) (**ООП**, іноді *об'єктно-зорієнто́ване програмува́ння*[[2]](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F#cite_note-2)[[1]](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F#cite_note-кочерга-мейнарович-1)[[3]](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F#cite_note-3); [англ.](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) *Object-oriented programming, OOP*) — одна з [парадигм](https://uk.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%B4%D0%B8%D0%B3%D0%BC%D0%B8_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) [програмування](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F), яка розглядає програму як множину [«об'єктів»](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)), що взаємодіють між собою.[[4]](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F#cite_note-4) Основу ООП складають чотири основні концепції: [інкапсуляція](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D0%BA%D0%B0%D0%BF%D1%81%D1%83%D0%BB%D1%8F%D1%86%D1%96%D1%8F), [успадкування](https://uk.wikipedia.org/wiki/%D0%A3%D1%81%D0%BF%D0%B0%D0%B4%D0%BA%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)), [поліморфізм](https://uk.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D1%96%D0%BC%D0%BE%D1%80%D1%84%D1%96%D0%B7%D0%BC_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)) та абстракція[[5]](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F#cite_note-5) Однією з переваг ООП є краща [модульність](https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D1%8C_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)) програмного забезпечення (тисячу функцій процедурної мови, в ООП можна замінити кількома десятками класів із своїми методами). Попри те, що ця парадигма з'явилась в [1960-х](https://uk.wikipedia.org/wiki/1960-%D1%82%D1%96) роках, вона не мала широкого застосування до [1990-х](https://uk.wikipedia.org/wiki/1990-%D1%82%D1%96), коли розвиток комп'ютерів та комп'ютерних мереж дав змогу писати надзвичайно об'ємне і складне програмне забезпечення, що змусило переглянути підходи до написання програм. На теперешній час багато [мов програмування](https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) або підтримують ООП ([PHP](https://uk.wikipedia.org/wiki/PHP), [Lua](https://uk.wikipedia.org/wiki/Lua)) або ж є цілком [об'єкто-орієнтованими](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) (зокрема, [Java](https://uk.wikipedia.org/wiki/Java_(%D0%BC%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)), [C#](https://uk.wikipedia.org/wiki/C_Sharp_(%D0%BC%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)), [C++](https://uk.wikipedia.org/wiki/C%2B%2B), [Python](https://uk.wikipedia.org/wiki/Python), [Ruby](https://uk.wikipedia.org/wiki/Ruby) і [Objective-C](https://uk.wikipedia.org/wiki/Objective-C), [ActionScript 3](https://uk.wikipedia.org/wiki/ActionScript), [Swift](https://uk.wikipedia.org/wiki/Swift_(%D0%BC%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)), [Vala](https://uk.wikipedia.org/wiki/Vala)).

Об'єктно-орієнтоване програмування сягає своїм корінням до створення мови програмування [Симула](https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BC%D1%83%D0%BB%D0%B0) в 1960-х роках, одночасно з посиленням дискусій про [кризу програмного забезпечення](https://uk.wikipedia.org/w/index.php?title=%D0%9A%D1%80%D0%B8%D0%B7%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F&action=edit&redlink=1)[[en]](https://en.wikipedia.org/wiki/Software_crisis). Через ускладнення апаратного та програмного забезпечення було дуже важко зберегти якість програм. Об'єкто-орієнтоване програмування частково розв'язує цю проблему шляхом наголошення на модульності програми[[6]](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F#cite_note-MeyerCh3-6).

На відміну від традиційних поглядів, коли програму розглядали як набір [підпрограм](https://uk.wikipedia.org/wiki/%D0%9F%D1%96%D0%B4%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B0), або як перелік [інструкцій](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%86%D1%96%D1%8F) комп'ютеру, ООП-програми можна вважати сукупністю об'єктів. Відповідно до парадигми об'єктно-орієнтованого програмування, кожен об'єкт здатний отримувати [повідомлення](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BC%D1%96%D0%BD_%D0%BF%D0%BE%D0%B2%D1%96%D0%B4%D0%BE%D0%BC%D0%BB%D0%B5%D0%BD%D0%BD%D1%8F%D0%BC%D0%B8), обробляти дані, та надсилати повідомлення іншим об'єктам. Кожен об'єкт — своєрідний незалежний автомат з окремим призначенням та відповідальністю[[7]](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F#cite_note-BoochCh2-7).   
На думку [Алана Кея](https://uk.wikipedia.org/wiki/%D0%90%D0%BB%D0%B0%D0%BD_%D0%9A%D0%B5%D0%B9), розробника мови [Smalltalk](https://uk.wikipedia.org/wiki/Smalltalk), якого вважають одним з «батьків-засновників» ООП, об'єкто-орієнтований підхід полягає в наступному наборі основних принципів:

* Все є об'єктами.
* Всі дії та розрахунки виконуються шляхом взаємодії (обміну даними) між об'єктами, під час якої один об'єкт потребує, щоб інший об'єкт виконав деяку дію. Об'єкти взаємодіють, надсилаючи й отримуючи повідомлення. Повідомлення — це запит на виконання дії, доповнений набором аргументів, які можуть знадобитися під час виконання дії.
* Кожен об'єкт має незалежну пам'ять, яка складається з інших об'єктів.
* Кожен об'єкт є представником (екземпляром, примірником) [класу](https://uk.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)), який виражає загальні властивості об'єктів.
* У класі задається поведінка (функціональність) об'єкта. Таким чином усі об'єкти, які є екземплярами одного класу, можуть виконувати одні й ті ж самі дії.
* Класи організовані у єдину деревоподібну структуру з загальним корінням, яка називається ієрархією [успадкування](https://uk.wikipedia.org/wiki/%D0%A3%D1%81%D0%BF%D0%B0%D0%B4%D0%BA%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)). Пам'ять та поведінка, зв'язані з екземплярами деякого класу, автоматично доступні будь-якому класу, розташованому нижче в ієрархічному дереві.

Таким чином, програма є набором об'єктів, що мають стан та поведінку. Об'єкти взаємодіють використовуючи повідомлення. Будується ієрархія об'єктів: програма в цілому — це об'єкт, для виконання своїх функцій вона звертається до об'єктів що містяться у ньому, які своєю чергою виконують запит шляхом звернення до інших об'єктів програми. Звісно, щоб уникнути нескінченної рекурсії у зверненнях, на якомусь етапі об'єкт трансформує запит у повідомлення до стандартних системних об'єктів, що даються мовою та середовищем програмування. Стійкість та керованість системи забезпечуються шляхом чіткого розподілу відповідальності об'єктів (за кожну дію відповідає певний об'єкт), однозначного означення інтерфейсів міжоб'єктної взаємодії та повної ізольованості внутрішньої структури об'єкта від зовнішнього середовища (інкапсуляції).

# **РОЗДІЛ 1 ПОСТАНОВКА ЗАДАЧІ**

Ціле довільної довжини в зовнішній формі представлення у вигляді

рядка символів-цифр. Розробити клас, об'єкт якого реалізує «користувацький»

тип даних. Забезпечити його довільну розмірність за рахунок використання в

об'єкті динамічних структур даних. Розробити необхідні конструктори,

деструктор, конструктор копіювання, а також методи, що забезпечують зміну

окремих складових частин об'єкта й виведення його вмісту

## **Мова програмування C#**

C# (вимовляється C-sharp) — це об’єктно-орієнтована мова програмування з безпечною системою введення тексту для платформи .NET.

Розроблено Андерсом Гейлсбергом, Скоттом Вілтамутом і Пітером Голде під егідою Microsoft Research (належить Microsoft).

Синтаксис C# подібний до C++ і Java.

Мова має сильну статичну типізацію та підтримує поліморфізм, перевантаження операторів, покажчики на функції-члени класу, атрибути, події, властивості, винятки та коментарі у форматі XML.

Слідуючи за мовами-попередниками C++, Object Pascal, Modula та Smalltalk, C# виключає деякі моделі, які виявилися проблематичними при розробці програмних систем, таких як C#.

На відміну від C++, ця мова не допускає багатокласового успадкування.

Станом на 2024 рік поточною стабільною версією C# є 17.10.3

випущена в 2024 році як частина платформи .NET 7.0

Символ # у назві мови можна інтерпретувати як пару з двох плюсів ++.

Це знаменує новий крок у розвитку мови порівняно з C++ (подібно до кроку від C до C++) і також може бути інтерпретовано як музичний символ хреста.

, разом з англійською літерою C, називають звуком C-дієз.

Останній дав назву мові.

# (Oktothorp) насправді є символом числа на більшості клавіатур і відрізняється від знака фунта ♯ (Unicode U+266F), але як творець мови Microsoft неодноразово просив клієнтів прийняти цей стиль.

## **Вступ до роботи**

Метою цієї роботи є розробка класу BigInt, який дозволяє працювати з великими цілими числами у середовищі Windows Forms на базі .NET Framework 4.8.1. Програма дозволяє додавати цифри до числа, змінювати окремі цифри та виводити число у вигляді рядка.

## **Реалізація**

#### **1.3.1 Вступ**

Проект спрямований на створення класу для роботи з великими цілими числами (BigInt) та інтеграцію його в Windows Forms додаток. Нижче наведені основні кроки, які були виконані для реалізації цього проекту.

**Етапи реалізації**

**1. Створення проекту у Visual Studio**

Проект був створений у Visual Studio як Windows Forms додаток на базі .NET Framework 4.8.1.

**2. Створення класу BigInt**

Створено клас BigInt, який дозволяє працювати з великими цілими числами, зберігаючи їх у вигляді динамічного списку символів. Цей клас реалізує необхідні конструктори, деструктор, методи додавання та зміни цифр, а також метод для виведення числа у вигляді рядка.

**3. Інтерфейс користувача**

Головне вікно програми було створено у Windows Forms. Інтерфейс містить:

* Поле для введення цифри для додавання до числа.
* Поля для введення індексу та нової цифри для зміни конкретної цифри числа.
* Кнопки для додавання та зміни цифр.
* Мітку для відображення поточного значення числа.

**4. Обробка подій**

Для кожної кнопки були створені обробники подій:

* btnAddDigit\_Click: Додає введену цифру до числа.
* btnSetDigit\_Click: Змінює цифру на вказаній позиції.
* UpdateDisplay: Оновлює відображення поточного значення числа.

# **РОЗДІЛ 2 ОПИС ФУНКЦІОНАЛУ КЛАСІВ**

Дана розрахунково-графічна робота включає в себе один проект в середовищі розробки Visual Studio 2022: проект Windows Forms (.NET Framework 4.8) під назвою “BigIntApp ”.

## **2.1 Класи основного проекту Windows Forms**

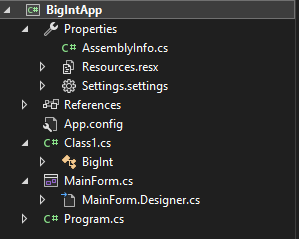


Рисунок 2.1 – Структура проекту Windows Forms

 **Properties**

* **AssemblyInfo.cs**: Цей файл містить метадані про збірку, такі як назва, версія, авторські права тощо. Він використовується для налаштування збірки додатку.

 **Resources.resx**: Файл ресурсів, який використовується для зберігання локалізованих рядків, зображень та інших ресурсів, які можуть бути використані в додатку.

 **Settings.settings**: Файл налаштувань проекту. Він дозволяє зберігати параметри конфігурації, які можуть бути доступні під час виконання програми.

 **References**: Папка, яка містить посилання на бібліотеки та збірки, які використовуються в проекті. Це можуть бути як стандартні бібліотеки .NET, так і сторонні бібліотеки.

**App.config**: Файл конфігурації додатку, який може містити налаштування для різних компонентів додатку, таких як підключення до баз даних, налаштування логування тощо.

**Class1.cs**: Цей файл, ймовірно, створений за замовчуванням і може містити початковий код для класу, який ви створили. Його можна видалити або перейменувати для конкретного використання в вашому проекті.

**BigInt**

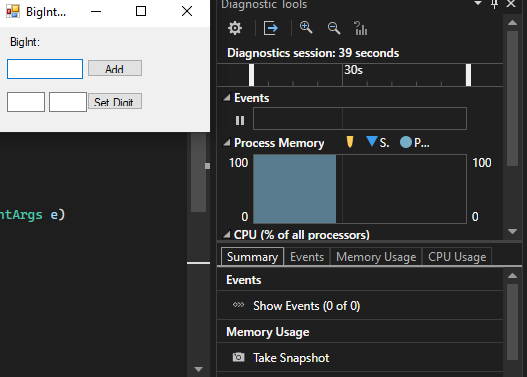
* **BigInt.cs**: Це файл класу, в якому реалізований клас BigInt, призначений для роботи з великими цілими числами. В ньому містяться методи та властивості для маніпулювання такими числами.

**MainForm.cs**: Файл, який містить код головної форми Windows Forms. Він включає в себе логіку для взаємодії користувача з додатком.

**MainForm.Designer.cs**: Файл, автоматично згенерований дизайнером форми у Visual Studio. Він містить код для ініціалізації компонентів форми та їх розташування на формі.

**Program.cs**: Головний файл проекту, який містить точку входу в додаток. Він містить метод Main, який запускає додаток і відкриває головну форму.

# **РОЗДІЛ 3 ДИЗАЙН КОРИСТУВАЦЬКОГО ІНТЕРФЕЙСУ**



## **3.1 Основна форма**

Рисунок 3.1 – Дизайн основної форми програми

Для коректної роботи проекту з великими числами в інтерфейсі користувача (Windows Forms) необхідно вводити валідні цифри та індекси, з якими можна працювати. Ось деякі поради щодо коректного введення даних:

**1.Основні правила введення**

1. Введення цифр:

- В текстове поле для введення цифри (наприклад, `txtDigit`) слід вводити лише одну цифру від `0` до `9`.

- Програма перевіряє, чи введений символ є цифрою, інакше видає помилку.

2. Введення індексів та нових цифр:

- В текстові поля для зміни цифр на конкретних позиціях (наприклад, `txtIndex` для індексу і `txtNewDigit` для нової цифри) слід вводити відповідно:

- Індекс — цілочислове значення, яке знаходиться в межах довжини поточного числа.

- Нова цифра — одинична цифра від `0` до `9`.

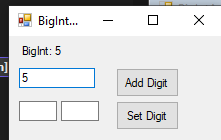
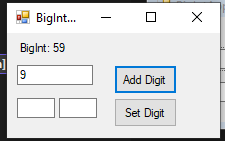
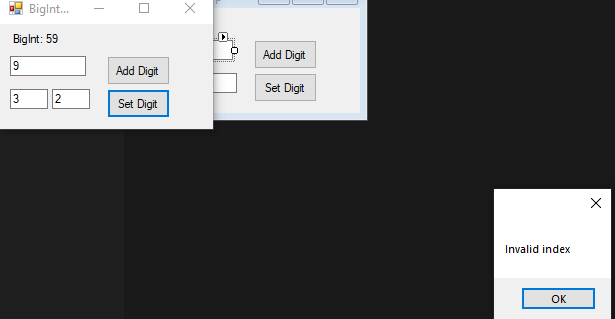
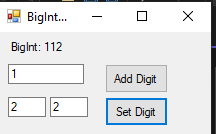


Рисунок 3.2 -Введення цифри 5

  
Рисунок 3.3-Введення цифри 9

  
Рисунок 3.4 – Помилка при введенні неправильних значень

  
Рисунок 3.5- Зміна числа через функцію Set Digit

3. Використання кнопок:

- \*\*Кнопка для додавання цифри\*\* (`btnAddDigit`): Додає введену цифру до кінця числа.

- \*\*Кнопка для зміни цифри\*\* (`btnSetDigit`): Змінює цифру на заданому індексі новою цифрою.

**2. Приклад роботи**

Розглянемо приклад роботи з додатком:

1. \*\*Додавання цифр\*\*:

- Введіть цифру `5` в текстове поле і натисніть кнопку "Додати цифру". Число стане `5`.

- Введіть цифру `3` і натисніть кнопку "Додати цифру". Число стане `53`.

2. \*\*Зміна цифри\*\*:

- Введіть індекс `1` (друга позиція) в текстове поле для індексу і введіть цифру `7` в текстове поле для нової цифри. Натисніть кнопку "Змінити цифру". Число зміниться з `53` на `57`.

Обробка помилок

Програма обробляє помилки введення, наприклад:

- Якщо ввести некоректну цифру або символ, програма видасть повідомлення про помилку.

- Якщо індекс виходить за межі допустимих значень, програма також видасть повідомлення про помилку.

3. **Рекомендації**

- Переконайтеся, що вводите лише цифри та коректні індекси для коректної роботи додатку.

- Перевірте правильність введення перед натисканням кнопок, щоб уникнути помилок.

# **РОЗДІЛ 4 ВИХІДНИЙ КОД ПРОЕКТУ**

## **4.1 Код основної форми**

using System;

using System.Windows.Forms;

namespace BigIntApp

{

public partial class MainForm : Form

{

private BigInt bigInt;

public MainForm()

{

InitializeComponent();

bigInt = new BigInt();

}

private void btnAddDigit\_Click(object sender, EventArgs e)

{

try

{

char digit = txtDigit.Text[0];

bigInt.AddDigit(digit);

UpdateDisplay();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void btnSetDigit\_Click(object sender, EventArgs e)

{

try

{

int index = int.Parse(txtIndex.Text);

char digit = txtNewDigit.Text[0];

bigInt.SetDigit(index, digit);

UpdateDisplay();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void UpdateDisplay()

{

lblBigInt.Text = "BigInt: " + bigInt.ToString();

}

}

}

## **4.2 Код класу Program.cs**

using System;

using System.Windows.Forms;

namespace BigIntApp

{

static class Program

{

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new MainForm());

}

}

}

## **4.3 Код Class1.cs**

using System;

using System.Collections.Generic;

public class BigInt

{

private List<char> digits;

public BigInt()

{

digits = new List<char>();

}

public BigInt(string number)

{

digits = new List<char>(number);

}

public BigInt(BigInt other)

{

digits = new List<char>(other.digits);

}

~BigInt()

{

digits.Clear();

}

public void AddDigit(char digit)

{

if (char.IsDigit(digit))

{

digits.Add(digit);

}

else

{

throw new ArgumentException("Invalid digit");

}

}

public override string ToString()

{

return new string(digits.ToArray());

}

public int Length()

{

return digits.Count;

}

public void SetDigit(int index, char digit)

{

if (index < 0 || index >= digits.Count)

{

throw new IndexOutOfRangeException("Invalid index");

}

if (!char.IsDigit(digit))

{

throw new ArgumentException("Invalid digit");

}

digits[index] = digit;

}

}

}

# **ВИСНОВКИ**

Метою цієї курсової роботи є розробка спеціального типу даних для обробки великих цілих чисел (BigInt) за допомогою динамічних структур даних у середовищі Windows Forms.

Основною метою було створити клас BigInt, який може обробляти числа довільної довжини, і інтегрувати його в графічний інтерфейс користувача для зручного маніпулювання.

Етапи розробки Аналіз завдання та дизайну класу BigInt: На першому етапі було проведено аналіз завдання, що дозволило визначити основні вимоги до класу BigInt.

Було вирішено використовувати список символів для зберігання цифр числа.

Це дозволяє динамічно змінювати розмір чисел під час операцій.

Реалізація класу BigInt: Клас BigInt реалізовано для забезпечення базових операцій над великими числами, таких як додавання нових цифр, зміна існуючих цифр у певних позиціях і виведення числа у вигляді рядка.

Для цього були розроблені необхідні методи та конструктори, у тому числі конструктори копіювання та деструктори для очищення ресурсів.

Розробка інтерфейсу користувача: У середовищі Windows Forms створено інтерфейс користувача, що дозволяє взаємодіяти з об’єктами класу BigInt.

Цей інтерфейс містить текстові поля для введення чисел та індексів, кнопки для додавання та зміни чисел, а також мітку для відображення поточного значення числа.

Тестування та налагодження: Після впровадження основних функцій проект був ретельно протестований для перевірки правильності роботи.

Тестування включає перевірку додавання та зміни чисел, обробку недійсних введених даних і перевірку граничних умов.

Досягнення Створення класу BigInt: Клас BigInt успішно реалізовано, і тепер він може зберігати та обробляти великі числа, містить усі методи, необхідні для роботи з такими числами.

Інтеграція з інтерфейсом користувача: Програми Windows Forms містять корисні функції, які дозволяють користувачам додавати числа до чисел, змінювати числа в певних позиціях і відображати поточне значення чисел у реальному часі.

Обробка помилок: Цей проект забезпечує обробку помилок для забезпечення стабільної роботи програми, якщо користувач вводить дані неправильно.

Це покращує надійність програми та зручність використання.

Висновок і наступні кроки Цей проект продемонстрував успішну реалізацію спеціального типу даних для роботи з великими цілими числами та його інтеграцію в графічний інтерфейс користувача.

Це допомогло мені ознайомитися з основними принципами роботи з динамічними структурами даних, реалізацією класів на C# та розробкою додатків у середовищі Windows Forms.

Можливі вдосконалення та подальший розвиток проекту: Розширення функціональності: Додавання математичних операцій, таких як віднімання, множення, ділення, порівняння великих чисел.

Оптимізація продуктивності: Оптимізація методів класу BigInt для підвищення продуктивності під час роботи з дуже великими числами.

Покращення інтерфейсу користувача: Ми покращили дизайн і функціональність інтерфейсу користувача, щоб зробити взаємодію з програмою більш інтуїтивно зрозумілою та зручною.

Додати локалізацію: Додати багатомовну підтримку для інтерфейсу користувача за допомогою файлів ресурсів.

Цей проект надав поглиблені знання та практичні навички з об’єктно-орієнтованого програмування, динамічних структур даних та розробки додатків Windows Forms.

# **СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. **Документація по мові програмування C#** [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/dotnet/csharp/
2. **Dou - український форум розробників програмного забезпечення** [Електронний ресурс]. – Режим доступу: https://dou.ua/forums/
3. **Курс українською мовою по вивченню C# з нуля для початківців.** [Електронний ресурс]. – Режим доступу: https://www.youtube.com/playlist?list=PL7vq4D0vOpQZI9\_AZIPGh4nxuePewUR5X  
    4. **Інструменти розробки проекту: Microsoft Visual Studio 2022 (версія 17.10.3), Replit-** Режим доступу: <https://replit.com/> (використувалось для тестування окремих елементів коду)  
   5. **Основні відомості про мову програмування C#** [Електронний ресурс]. – Режим доступу: https://armedsoft.com/ua/blog/osnovni-vidomosti-pro-movu-programuvannya-c
4. **Файли проекту курсової роботи** [Електронний ресурс]. – Режим доступу: https://github.com/Dead4light/oop-rgr