ЗМІСТ

[ВСТУП 2](#_Toc65599721)

[РОЗДІЛ 1. ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ 3](#_Toc65599722)

[1.1. Математика 3](#_Toc65599723)

[1.2. Калькулятори 20](#_Toc65599724)

[РОЗДІЛ 2. ВИБІР ЗАСОБІВ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ 27](#_Toc65599726)

[2.1. Мова програмування C# 27](#_Toc65599727)

[2.2. Фреймворк для розробки додатків на Windows (Windows Forms) 37](#_Toc65599728)

[РОЗДІЛ 3. РЕЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ 43](#_Toc65599729)

[3.1. Діаграма класів 43](#_Toc65599730)

[3.2. Діаграма варіантів використання 44](#_Toc65599731)

[3.3. Розробка інтерфейсу користувача 45](#_Toc65599732)

[3.4. Розробка багатокористувацьких функцій 47](#_Toc65599733)

[ВИСНОВКИ 51](#_Toc65599734)

# ВСТУП

В теперішньому часі людству потрібно все більше і більше сучасних, а також оптимальних рішень в вирішенні сучасних проблем.

Для цього вони почали розробляти комп’ютерної технології, які в майбутньому часі почали використовуватись щоденно для вирішення різноманітних проблем, а також користуватись великим попитом.

Зробивши з цього факту висновки, люди також користувались настільними калькуляторами, які були окремим пристроєм для вирішення математичних задач.

На теперішній час доступ до калькулятора зможе отримати кожна людина, яка має телефонний пристрій, а також ПК (Персональний Комп’ютер) та може з мінімальними зусиллями виконати свої потреби в виконанні математичного прикладу.

Виходячи з такої великої актуальності теми технологічних програмних рішень саме її було обрано як тему даної роботи, а саме розроблення статистичного калькулятора.

Для виконання поставленої задачі слід виконати наступні завдання:

* Провести огляд предметної області
  + Визначити, які дії має використовувати технологія
  + Визначити поняття створення даного функціоналу
* Обрати засоби реалізації програмного забезпечення
* Розробити діаграму класів
* Розробити інтерфейс користувача
* Розробити основний функціонал програми за допомогою коду

# РОЗДІЛ 1. ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1. Математика

Історія [математики](https://uk.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0) — галузь знань, що займається дослідженням походження та розвитку математичних відкриттів і методів, а також математичних праць минулого.

Математика первісно виникла як один із напрямків пошуку [істини](https://uk.wikipedia.org/wiki/%D0%86%D1%81%D1%82%D0%B8%D0%BD%D0%B0) (у [грецькій філософії](https://uk.wikipedia.org/wiki/%D0%94%D0%B0%D0%B2%D0%BD%D1%8C%D0%BE%D0%B3%D1%80%D0%B5%D1%86%D1%8C%D0%BA%D0%B0_%D1%84%D1%96%D0%BB%D0%BE%D1%81%D0%BE%D1%84%D1%96%D1%8F)) у сфері просторових відношень (землеміряння — [геометрії](https://uk.wikipedia.org/wiki/%D0%93%D0%B5%D0%BE%D0%BC%D0%B5%D1%82%D1%80%D1%96%D1%8F)) та обчислень ([арифметики](https://uk.wikipedia.org/wiki/%D0%90%D1%80%D0%B8%D1%84%D0%BC%D0%B5%D1%82%D0%B8%D0%BA%D0%B0)), для практичних потреб людини рахувати, обчислювати, вимірювати, досліджувати форми і [рух фізичних тіл](https://uk.wikipedia.org/wiki/%D0%9C%D0%B5%D1%85%D0%B0%D0%BD%D1%96%D0%BA%D0%B0). Нині цей термін позначає цілком визначену галузь знань, пов'язану з дослідженням задач про кількість, просторові форми, процеси розвитку та [формальні структури](https://uk.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0), в основі якого лежать точні

## Періодизація

Історію математики можна поділити на чотири періоди.

1. У перший період (приблизно 6-5 ст. до н. е.) сформувалося поняття цілого числа, раціонального дробу, відстані, площі, об'єму, створені правила дій з числами, найпростіші правила визначення площ фігур і об'ємів тіл. Так накопичився матеріал, що склався в арифметику. Вимірювання площ і об'ємів сприяло розвиткові геометрії. На базі створення методів арифметичних обчислень виникла алгебра, а в зв'язку із запитами астрономії — тригонометрія. Однак у цей період математика ще не була дедуктивною наукою, вона складалася переважно з прикладів на розв'язування окремих задач, іноді становила збірку правил для їхнього розв'язування.
2. У другий період (до середини XVII ст.) математика стає самостійною наукою зі своєрідним, чітко вираженим методом і системою основних понять. В Індії було створено десяткову систему числення, в Китаї — метод розв'язування лінійних рівнянь з двома і трьома невідомими; створена стародавніми греками система викладу елементарної геометрії стала зразком дедуктивної побудови математичної теорії на багато століть вперед. У цей період з арифметики поступово виділяється теорія чисел. Велике значення мали праці [Піфагора Самоського](https://uk.wikipedia.org/wiki/%D0%9F%D1%96%D1%84%D0%B0%D0%B3%D0%BE%D1%80), Гіппократа Хіоського, Евдокса Кнідського, Евкліда, Архімеда, Діофанта. У Київській Русі математична освіта була на рівні найкультурніших країн Європи того часу.
3. Третій період (до початку ХХ ст.), в який було створено математику змінних величин, — суттєво новий період у розвитку математики.
4. Четвертий — сучасний період — характеризується систематичним вивченням можливих типів кількісних відношень і просторових форм.

## Математика у первісному суспільстві

Уже в найперших писемних знахідках є докази, які свідчать про математичні знання їхніх авторів, що використовувались для вимірювання [часу](https://uk.wikipedia.org/wiki/%D0%A7%D0%B0%D1%81) на основі спостереження за небесними світилами. Доісторичні артефакти, виявлені в [Африці](https://uk.wikipedia.org/wiki/%D0%90%D1%84%D1%80%D0%B8%D0%BA%D0%B0) та [Франції](https://uk.wikipedia.org/wiki/%D0%A4%D1%80%D0%B0%D0%BD%D1%86%D1%96%D1%8F), вказують на здійснення перших спроб квантифікації часу. Існує припущення, що відліком часу займалися жінки, які реєстрували місячні цикли або фази місяця. Паралельно розвивалися уявлення про [число](https://uk.wikipedia.org/wiki/%D0%A7%D0%B8%D1%81%D0%BB%D0%BE): вірогідно, спостерігаючи за групами (стадами) тварин, люди почали розрізняти поняття «один», «два» та «багато». Саме такі кількісні уявлення донині збереглися у [зулусів](https://uk.wikipedia.org/wiki/%D0%97%D1%83%D0%BB%D1%83%D1%81%D0%B8), африканських [пігмеїв](https://uk.wikipedia.org/wiki/%D0%9F%D1%96%D0%B3%D0%BC%D0%B5%D1%97_(%D1%80%D0%B0%D1%81%D0%B0)) та ще ряду племен — австралійських, бразильських тощо. Згодом числа об'єднували у групи, утворюючи більші одиниці лічби; зазвичай використовували пальці однієї чи обох рук або ж рук і ніг, що давало лічбу з основою 5, 10 або 20. Записи вели позначенням одиниць, зарубками, камінцями тощо.

## Математика найдавніших цивілізацій

Найдавніші відомості про використання математики — господарські задачі в [Стародавньому Єгипті](https://uk.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D1%80%D0%BE%D0%B4%D0%B0%D0%B2%D0%BD%D1%96%D0%B9_%D0%84%D0%B3%D0%B8%D0%BF%D0%B5%D1%82) ([Папірус Рінда](https://uk.wikipedia.org/wiki/%D0%9F%D0%B0%D0%BF%D1%96%D1%80%D1%83%D1%81_%D0%A0%D1%96%D0%BD%D0%B4%D0%B0), [Московський папірус](https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D1%81%D0%BA%D0%BE%D0%B2%D1%81%D1%8C%D0%BA%D0%B8%D0%B9_%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%BD%D0%B8%D0%B9_%D0%BF%D0%B0%D0%BF%D1%96%D1%80%D1%83%D1%81), Шкіряний сувій єгипетської математики) та [Вавилонії](https://uk.wikipedia.org/wiki/%D0%92%D0%B0%D0%B2%D0%B8%D0%BB%D0%BE%D0%BD%D1%96%D1%8F) (Математичні тексти Суз). Вона використовувалася для календарних обрахунків, розподілу врожаю, організації суспільних робіт, збирання податків.

### Вавилонське царство

Про [вавилонську цивілізацію](https://uk.wikipedia.org/wiki/%D0%92%D0%B0%D0%B2%D0%B8%D0%BB%D0%BE%D0%BD%D1%81%D1%8C%D0%BA%D0%B5_%D1%86%D0%B0%D1%80%D1%81%D1%82%D0%B2%D0%BE), на щастя, нам відомо доволі багато. Все це завдяки глиняним табличкам, на яких були [клинописні тексти](https://uk.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B8%D0%BD%D0%BE%D0%BF%D0%B8%D1%81), вік яких датується приблизно від 2000 років до н. е. та аж до III століття до н. е. Математика знайдених [клинописних табличок](https://uk.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B8%D0%BD%D0%BE%D0%BF%D0%B8%D1%81) в основному стосувалася тільки моментів, пов'язаних із веденням господарства. Також проста [арифметика](https://uk.wikipedia.org/wiki/%D0%90%D1%80%D0%B8%D1%84%D0%BC%D0%B5%D1%82%D0%B8%D0%BA%D0%B0) і [алгебра](https://uk.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%B5%D0%B1%D1%80%D0%B0) застосовувалися для оплати товарів, обчислення простих або складних [відсотків](https://uk.wikipedia.org/wiki/%D0%92%D1%96%D0%B4%D1%81%D0%BE%D1%82%D0%BE%D0%BA). З часом, коли почали будувати канали, зерносховища та інші складні споруди, арифметичні та геометричні задачі стали складнішими. Математика знадобилася також для ведення обліку громадських робіт, яких у той час було чимало. Вкрай важливу роль математика зіграла у розрахунках [календаря](https://uk.wikipedia.org/wiki/%D0%9A%D0%B0%D0%BB%D0%B5%D0%BD%D0%B4%D0%B0%D1%80). Адже саме за календарем визначали час сівби та збору врожаю, а також усі релігійні свята. Саме [вавилонська астрономія](https://uk.wikipedia.org/wiki/%D0%92%D0%B0%D0%B2%D0%B8%D0%BB%D0%BE%D0%BD%D1%81%D1%8C%D0%BA%D0%B0_%D0%B0%D1%81%D1%82%D1%80%D0%BE%D0%BD%D0%BE%D0%BC%D1%96%D1%8F) поклала початок поділу кола на 360 [градусів](https://uk.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D0%B4%D1%83%D1%81_(%D0%B3%D0%B5%D0%BE%D0%BC%D0%B5%D1%82%D1%80%D1%96%D1%8F)), а градуса — на хвилини та секунди. Вавилонянам належить одна з перших [систем числення](https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%BD%D1%8F). Для цього вони використовували числа від 1 до 59, основою яких була 10-ка. Символ, що позначав одиницю, вавилоняни повторювали необхідну кількість разів для чисел від 1 до 9. Подальші позначення, тобто, від 11 до 59, позначалися комбінацією символу числа 10, а також символу одиниці. Для чисел, починаючи з 60 і більше, була введена [позиційна система числення](https://uk.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B7%D0%B8%D1%86%D1%96%D0%B9%D0%BD%D0%B0_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%BD%D1%8F), основою якої стало число 60. Суттєвим проривом у вавилонській математиці став позиційний принцип. Тобто, один і той же числовий знак або символ мав різні значення залежно від місця його розташування. Прикладом може слугувати значення 6 у нинішньому записі числа 606. Однак у вавилонян нуль був відсутній, саме тому і набір символів міг означати таке: 65 — це 60+5, і 3605 — це 602+0+5. Виникала неоднозначність зі сприйняттям [дробів](https://uk.wikipedia.org/wiki/%D0%94%D1%80%D1%96%D0%B1), оскільки ті ж самі символи могли трактуватися і як [число](https://uk.wikipedia.org/wiki/%D0%A7%D0%B8%D1%81%D0%BB%D0%BE), і як [дріб](https://uk.wikipedia.org/wiki/%D0%94%D1%80%D1%96%D0%B1). Однак ця проблема вирішувалася досить просто — все залежало від конкретного контексту.

### Єгипет

Наше розуміння староєгипетської математики ґрунтується в основному на двох [папірусах](https://uk.wikipedia.org/wiki/%D0%9F%D0%B0%D0%BF%D1%96%D1%80%D1%83%D1%81), що датуються приблизно 1700 роком до н. е. Однак ті математичні відомості, які містять ці папіруси, належать до зовсім раннього періоду, приблизно 3500 років до н. е. Єгиптяни відмінно орієнтувалися на той момент в математиці. Вони використовували її для обчислення [маси тіл](https://uk.wikipedia.org/wiki/%D0%9C%D0%B0%D1%81%D0%B0), [площ](https://uk.wikipedia.org/wiki/%D0%9F%D0%BB%D0%BE%D1%89%D0%B0) посівів, [об'ємів](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BC) зерносховищ, розмірів податків, кількості каменів, які призначалися для будівництва різних споруд. У папірусах знайшлася і згадка про завдання з визначенням кількості зерна для приготування необхідного числа кухлів [пива](https://uk.wikipedia.org/wiki/%D0%9F%D0%B8%D0%B2%D0%BE) і навіть більш складних, де для приготування [пива](https://uk.wikipedia.org/wiki/%D0%9F%D0%B8%D0%B2%D0%BE) використовувалися одночасно кілька сортів зерна. Однак, незважаючи на всі ці факти, рівень [астрономії](https://uk.wikipedia.org/wiki/%D0%90%D1%81%D1%82%D1%80%D0%BE%D0%BD%D0%BE%D0%BC%D1%96%D1%8F) в [Стародавньому Єгипті](https://uk.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D1%80%D0%BE%D0%B4%D0%B0%D0%B2%D0%BD%D1%96%D0%B9_%D0%84%D0%B3%D0%B8%D0%BF%D0%B5%D1%82) все ж істотно поступався ступеню її розвитку у [Вавилонському царстві](https://uk.wikipedia.org/wiki/%D0%92%D0%B0%D0%B2%D0%B8%D0%BB%D0%BE%D0%BD%D1%81%D1%8C%D0%BA%D0%B5_%D1%86%D0%B0%D1%80%D1%81%D1%82%D0%B2%D0%BE). Вся давньоєгипетська писемність була заснована на [ієрогліфах](https://uk.wikipedia.org/wiki/%D0%86%D1%94%D1%80%D0%BE%D0%B3%D0%BB%D1%96%D1%84%D0%B8). Причому система числення, як і астрономія, сильно поступалася вавилонській системі. Єгиптяни використовували тільки непозиційну десяткову систему, де числа від одного до дев'яти позначалися за допомогою вертикальних паличок відповідним числом. Що стосується послідовних ступенів числа десять, то тут вже використовувалися індивідуальні символи. У єгиптян [геометрія](https://uk.wikipedia.org/wiki/%D0%93%D0%B5%D0%BE%D0%BC%D0%B5%D1%82%D1%80%D1%96%D1%8F) здебільшого зводилася до обчислень площ [круга](https://uk.wikipedia.org/wiki/%D0%9A%D1%80%D1%83%D0%B3), [трикутників](https://uk.wikipedia.org/wiki/%D0%A2%D1%80%D0%B8%D0%BA%D1%83%D1%82%D0%BD%D0%B8%D0%BA), [прямокутників](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D1%8F%D0%BC%D0%BE%D0%BA%D1%83%D1%82%D0%BD%D0%B8%D0%BA), [трапецій](https://uk.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BF%D0%B5%D1%86%D1%96%D1%8F) і до формул об'ємів певних тіл. Варто також відзначити, що, незважаючи на всю велич [єгипетських пірамід](https://uk.wikipedia.org/wiki/%D0%84%D0%B3%D0%B8%D0%BF%D0%B5%D1%82%D1%81%D1%8C%D0%BA%D1%96_%D0%BF%D1%96%D1%80%D0%B0%D0%BC%D1%96%D0%B4%D0%B8), для їхнього будівництва єгиптяни використовували вкрай просту і примітивну математику.

### Піфагорійська школа

Математика як теорія отримала розвиток у школі [Піфагора](https://uk.wikipedia.org/wiki/%D0%9F%D1%96%D1%84%D0%B0%D0%B3%D0%BE%D1%80) (571—479 рр. до н. е.). Головним досягненням піфагорійців в області науки є істотний розвиток [математики](https://uk.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0) як за змістом, так і за формою. За змістом — відкриття нових математичних фактів. За формою — побудова геометрії і арифметики як теоретичних доказових наук, що вивчають властивості абстрактних понять про числа і геометричні форми. Дедуктивна побудова геометрії стала потужним стимулом її подальшого розвитку. Піфагорійці розвинули і обґрунтували [планіметрію](https://uk.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D0%BD%D1%96%D0%BC%D0%B5%D1%82%D1%80%D1%96%D1%8F) прямолінійних фігур — вчення про [паралельні лінії](https://uk.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%BB%D0%B5%D0%BB%D1%8C%D0%BD%D1%96_%D0%BF%D1%80%D1%8F%D0%BC%D1%96), [трикутники](https://uk.wikipedia.org/wiki/%D0%A2%D1%80%D0%B8%D0%BA%D1%83%D1%82%D0%BD%D0%B8%D0%BA), [чотирикутники](https://uk.wikipedia.org/wiki/%D0%A7%D0%BE%D1%82%D0%B8%D1%80%D0%B8%D0%BA%D1%83%D1%82%D0%BD%D0%B8%D0%BA), [правильні багатокутники](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%B0%D0%B2%D0%B8%D0%BB%D1%8C%D0%BD%D0%B8%D0%B9_%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE%D0%BA%D1%83%D1%82%D0%BD%D0%B8%D0%BA). Отримала розвиток елементарна теорія кола. Наявність у піфагорійців вчення про паралельні лінії вказує на те, що вони володіли методом [доведення від супротивного](https://uk.wikipedia.org/wiki/%D0%94%D0%BE%D0%B2%D0%B5%D0%B4%D0%B5%D0%BD%D0%BD%D1%8F_%D0%B2%D1%96%D0%B4_%D1%81%D1%83%D0%BF%D1%80%D0%BE%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B3%D0%BE) і вперше довели теорему про суму кутів трикутника. Вершиною досягнень піфагорійців у планіметрії є доказ [теореми Піфагора](https://uk.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D0%B5%D0%BC%D0%B0_%D0%9F%D1%96%D1%84%D0%B0%D0%B3%D0%BE%D1%80%D0%B0). Остання на багато століть раніше була сформульована вавилонськими, китайськими й індійськими вченими, однак її доказ їм не був відомий. Успіхи піфагорійців у [стереометрії](https://uk.wikipedia.org/wiki/%D0%A1%D1%82%D0%B5%D1%80%D0%B5%D0%BE%D0%BC%D0%B5%D1%82%D1%80%D1%96%D1%8F) були значними. Вони займалися вивченням властивостей [кулі](https://uk.wikipedia.org/wiki/%D0%9A%D1%83%D0%BB%D1%8F), відкрили побудову чотирьох правильних многокутників — [тетраедра](https://uk.wikipedia.org/wiki/%D0%A7%D0%BE%D1%82%D0%B8%D1%80%D0%B8%D0%B3%D1%80%D0%B0%D0%BD%D0%BD%D0%B8%D0%BA), [куба](https://uk.wikipedia.org/wiki/%D0%9A%D1%83%D0%B1), [октаедра](https://uk.wikipedia.org/wiki/%D0%9E%D0%BA%D1%82%D0%B0%D0%B5%D0%B4%D1%80) і [додекаедра](https://uk.wikipedia.org/wiki/%D0%94%D0%BE%D0%B4%D0%B5%D0%BA%D0%B0%D0%B5%D0%B4%D1%80) ([ікосаедр](https://uk.wikipedia.org/wiki/%D0%86%D0%BA%D0%BE%D1%81%D0%B0%D0%B5%D0%B4%D1%80) досліджував згодом Геетет). Однак вони не змогли обґрунтувати твердження, які стосуються об'ємів тіл (піраміди, конуса, циліндра і кулі), хоча, звичайно, ці твердження були встановлені емпірично на багато століть раніше. В галузі арифметики піфагорійці вивчали властивості парних і непарних, простих і складених натуральних чисел, шукали [досконалі числа](https://uk.wikipedia.org/wiki/%D0%94%D0%BE%D1%81%D0%BA%D0%BE%D0%BD%D0%B0%D0%BB%D1%96_%D1%87%D0%B8%D1%81%D0%BB%D0%B0), тобто такі, які дорівнюють сумі всіх своїх дільників (наприклад, 6=1+2+3; 28=1+2+4+7+14). Піфагорійці знали також дробові числа і в зв'язку з цим розробили теорію арифметичної та геометричної пропорцій. Вони володіли поняттями [середнього арифметичного](https://uk.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B5%D0%B4%D0%BD%D1%94_%D0%B0%D1%80%D0%B8%D1%84%D0%BC%D0%B5%D1%82%D0%B8%D1%87%D0%BD%D0%B5), [середнього геометричного](https://uk.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B5%D0%B4%D0%BD%D1%94_%D0%B3%D0%B5%D0%BE%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%87%D0%BD%D0%B5) і [середнього гармонійного](https://uk.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B5%D0%B4%D0%BD%D1%94_%D0%B3%D0%B0%D1%80%D0%BC%D0%BE%D0%BD%D1%96%D0%B9%D0%BD%D0%B5).

### Поворотний пункт в історії античної математики

Якими б великими не були досягнення піфагорійців у розвитку змісту та систематизації геометрії і арифметики, однак всі вони не можуть зрівнятися зі зробленим ними ж відкриттям несумірних величин. Це відкриття стало поворотним пунктом в історії античної математики. З приводу цього відкриття [Аристотель](https://uk.wikipedia.org/wiki/%D0%90%D1%80%D1%96%D1%81%D1%82%D0%BE%D1%82%D0%B5%D0%BB%D1%8C) говорив, що [Піфагор](https://uk.wikipedia.org/wiki/%D0%9F%D1%96%D1%84%D0%B0%D0%B3%D0%BE%D1%80) показав, що якщо б діагональ квадрата була б порівнянна з його стороною, то парне дорівнювало б непарному. Це зауваження Арістотеля показує, що при доведенні несумірності діагоналі квадрата з його стороною Піфагор використовував метод від супротивного. Наприкінці V століття до н. е. Феодор із Кирени встановив, що несумірність діагоналі квадрата з його стороною не є винятком. Він показав, що сторони квадратів, площі яких дорівнюють 3, 5, 6, …, 17 несумірні зі стороною одиничного квадрата. Піфагор вчив, що сутність усіх речей є число; число — самі речі; гармонія чисел — гармонія самих речей. Це природно: до відкриття Піфагора давньогрецькі математики вважали, що будь-які два відрізки мають спільну міру, хоча, може бути, і дуже малу. Піфагорійці знали тільки додатні цілі і дробові числа. Дотримуючись своєї філософської установки, вони, по суті справи, вважали, що кожна річ може бути охарактеризована позитивним цілим або дробовим числом, яке «виражає сутність» цієї речі. Насправді це означало, що геометрія будувалася на базі арифметики. Після виявлення існування несумірних величин перед піфагорійцями відкрилися дві можливості. Можна було спробувати розширити поняття числа за рахунок приєднання до раціональних чисел ірраціональних чисел, охарактеризувати несумірні величини числами іншої природи і таким чином відновити силу філософського принципу «все є число». Однак цей шлях, настільки природний і простий з сучасної точки зору, для піфагорійців був закритий. В цьому випадку треба було побудувати досить точну арифметичну теорію дійсних чисел, що при рівні піфагорійської математики було справою нереальною. Тому треба було йти іншим шляхом — шляхом певного перегляду вихідних принципів, наприклад, прийняти, що геометричні об'єкти є величинами більш загальної природи, ніж дробові і цілі числа, і намагатися будувати всю математику не на арифметичній, а на геометричній основі. Саме цей другий шлях і обрали піфагорійці, а слідом за ними більшість давньогрецьких математиків, аж до [Архімеда](https://uk.wikipedia.org/wiki/%D0%90%D1%80%D1%85%D1%96%D0%BC%D0%B5%D0%B4) і [Аполлонія](https://uk.wikipedia.org/wiki/%D0%90%D0%BF%D0%BE%D0%BB%D0%BB%D0%BE%D0%BD%D1%96%D0%B9_%D0%9F%D0%B5%D1%80%D0%B7%D1%8C%D0%BA%D0%B8%D0%B9).

### Період Академії

Період цілком самостійної діяльності греків в області математики починається з діяльності Платона і заснованої ним у 389 р. Філософської школи, відомої під ім'ям Академія. З цього часу подальший розвиток, якщо не всієї математики взагалі, то, безсумнівно, геометрії, зосереджується виключно в руках однієї грецької нації, яка й веде його, поки знаходить у своєму розпорядженні необхідні засоби. Головним результатом математичної діяльності самого Платона було створення філософії математики і, зокрема, її методології. Як відомо, його власні роботи дуже мало стосувалися збільшення математичних знань у кількісному відношенні і були спрямовані на встановлення точних визначень основних понять геометрії, на виявлення і відведення справжнього місця її основних положень, на приведення надбаних раніше математичних знань в суворий логічний зв'язок як між собою, так і з основними поняттями та положеннями, і нарешті, на приведення в повну ясність і вивчення методів відкриття та докази нових істин, методів. Методів, розроблених Платоном, за свідченням Прокла, було три: аналітичний, синтетичний та апологічний. Викладені, на підставі пізніших досліджень предмета, більш повним і головне більш певним чином, ці визначення представляються в наступному вигляді. Аналітичний метод полягає в утворенні ланцюга пропозицій, з яких кожне випливає з наступного за ним, як безпосередній наслідок. Синтетичний метод є частиною аналітичного і тому складається з ланцюгів пропозицій, з яких перше є доведена істина, а кожне з наступних є наслідком того, що йому передує. Про апологічний метод, або метод приведення до безглуздості, Евклід не говорить, але досить чітке його визначення поряд з нечіткими визначеннями аналізу і синтезу дає Прокл, при своєму приписуванні їх Платону; «Третій (апологічний) метод, — говорить він, — є приведення до неможливого, яке не доводить прямо того, що шукається, а спростовує те, що йому суперечить, і таким чином через зв'язок того й іншого знаходить істину». Вчені математики, що належали до Академії розпадалися на дві групи: на вчених, які отримали свою математичну освіту незалежно від Академії і перебували тільки в більш або менш тісних зносинах з нею, і на колишніх учнів Академії. До перших належали Теэтет Афінський, Леодам Фасосский, Архіт Тарентським і пізніше Евдокс Кнідський; до числа других — Неоклид, Леон, Амикл з Гераклеї, брати [Менехм](https://uk.wikipedia.org/wiki/%D0%9C%D0%B5%D0%BD%D0%B5%D1%85%D0%BC) і Динострат, і під час старості Платона — Теюдий з Магнезії, Кизикен Афінський, Гермотим Колофонский, Філіп Мендейский і Філіп Опунтский.

## Китай та Індія

Додатні і від'ємні кількості вперше в історії науки розрізняли в Китаї ще понад 2000 років тому. Уже у 8-й книзі збірника «Математика в дев'яти книгах» автори вільно користувалися від'ємними кількостями. У цій книжці є рівняння з від'ємними першими коефіцієнтами і вільними членами; тут же сформульовано правила додавання і віднімання від'ємних кількостей.

Додатні кількості в китайській математиці назвали «чен», від'ємні — «фу»; їх зображали різними кольорами: «чен» — червоним, «фу» — чорним. Такий спосіб зображення використовувався в Китаї до середини XIII ст., поки Лі Є не запропонував зручніше позначення від'ємних чисел — цифри, що зображали від'ємні числа, перекреслювали рискою навскіс справа наліво.

У V—VI ст. від'ємні числа поширюються в індійській математиці. В Індії від'ємні числа систематично застосовували і тлумачили в основному так само, як це ми робимо тепер.

Уже в творі Брамагупти «Перегляд системи Брами» (628 р.) ми читаємо: «Майно» і «майно» є «майно», сума двох «боргів» є «борг»; сума «майна» і нуля є «майно»; сума двох нулів є нуль… Борг, який віднімають від нуля, стає «майном», а «майно» — «боргом». Якщо треба відняти «майно» від «боргу», а «борг» від «майна», то беруть їх суму…".

Від'ємними числами індійські математики користувалися під час розв'язування рівнянь, причому віднімання замінювали додаванням до рівного й протилежного числа. Про те, як індійські вчені відкрили від'ємні числа, достовірно ми нічого не знаємо.

Слід зазначити, що основною особливістю індійської математики є переважання обчислювальних прийомів, які давалися в догматичній формі.

Розв'язуючи задачі на рух, виграш і. програш та інші, індійці, очевидно, на досвіді переконалися в зручності від'ємних чисел. Так, у творі видатного індійського математика й астронома Аріабхати І (476—бл. 550) подано розв'язування задачі, в якій йдеться про «момент зустрічі в минулому і майбутньому».

Проте, запровадивши від'ємні числа, індійські математики вважали їх не рівноправними елементами математики, а чимось подібним до логічних можливостей, бо, за висловом індійського математика Бхаскари, люди з ними не згодні.

Таким чином, при розв'язуванні алгебраїчних рівнянь математики зустрілися з від'ємними величинами, але почали вважати їх об'єктивними поняттями тільки тоді, коли реально розтлумачили.

Введення від'ємних чисел було зумовлене, в першу чергу, розвитком алгебри як науки, що дає загальні способи розв'язування арифметичних задач незалежно від вихідних числових даних. Від'ємні числа були необхідні вже при розв'язуванні задач, які зводяться до рівнянь першого степеня з однією змінною. Можливий від'ємний розв'язок у таких задачах можна пояснити прикладами протилежних величин (протилежно напрямлені вектори, температура, вища і нижча від нуля, майно — борг і т. д.).

## Середньовічна Європа

Незважаючи на всю свою велич, [Римська цивілізація](https://uk.wikipedia.org/wiki/%D0%A0%D0%B8%D0%BC%D1%81%D1%8C%D0%BA%D0%B0_%D1%96%D0%BC%D0%BF%D0%B5%D1%80%D1%96%D1%8F) не змогла залишити жодного істотного сліду в математиці, бо вона була вже надто стурбована рішенням своїх практичних проблем. А ось цивілізація, яка склалася в Європі часів раннього [Середньовіччя](https://uk.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B5%D0%B4%D0%BD%D1%8C%D0%BE%D0%B2%D1%96%D1%87%D1%87%D1%8F) (приблизно 400—1100 рр. н. е.) не була настільки продуктивною через ряд причин. По-перше, усе інтелектуальне життя було сконцентровано тільки на [теології](https://uk.wikipedia.org/wiki/%D0%91%D0%BE%D0%B3%D0%BE%D1%81%D0%BB%D1%96%D0%B2%27%D1%8F). Тому рівень математичних знань не піднімався вище простої [арифметики](https://uk.wikipedia.org/wiki/%D0%90%D1%80%D0%B8%D1%84%D0%BC%D0%B5%D1%82%D0%B8%D0%BA%D0%B0), а також найелементарніших розділів [«Начала» Евкліда](https://uk.wikipedia.org/wiki/%D0%9D%D0%B0%D1%87%D0%B0%D0%BB%D0%B0_%D0%95%D0%B2%D0%BA%D0%BB%D1%96%D0%B4%D0%B0). Мабуть, найголовнішим розділом математики в Середньовіччя залишалася [астрологія](https://uk.wikipedia.org/wiki/%D0%90%D1%81%D1%82%D1%80%D0%BE%D0%BB%D0%BE%D0%B3%D1%96%D1%8F). В той час будь-якого астролога називали математиком. А оскільки вся медицина на той момент ґрунтувалася переважно на астрологічних показаннях і протипоказаннях, всім медикам довелося терміново стати математиками. Приблизно в 1100 році західноєвропейська математика приступила до освоєння збережених візантійськими греками і арабами спадщини [Стародавнього світу Сходу](https://uk.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D1%80%D0%BE%D0%B4%D0%B0%D0%B2%D0%BD%D1%96%D0%B9_%D0%A1%D1%85%D1%96%D0%B4). Це тривало близько трьох століть. А оскільки араби практично повністю володіли усіма працями давніх греків, Європа змогла отримати в своє розпорядження просто величезну кількість математичної літератури. Всі праці перекладалися на латину, що сприяло істотному зростанню знань і підйому математичних досліджень в досить короткі терміни. Практично всі вчені Європи визнавали, що своє натхнення вони черпали саме з праць греків. Одним з найперших європейських математиків, якого варто загадати, став [Леонардо Пізанський](https://uk.wikipedia.org/wiki/%D0%A4%D1%96%D0%B1%D0%BE%D0%BD%D0%B0%D1%87%D1%87%D1%96) або [Фібоначчі](https://uk.wikipedia.org/wiki/%D0%A4%D1%96%D0%B1%D0%BE%D0%BD%D0%B0%D1%87%D1%87%D1%96). Завдяки його праці [«Книга абака»](https://uk.wikipedia.org/wiki/%D0%9A%D0%BD%D0%B8%D0%B3%D0%B0_%D0%B0%D0%B1%D0%B0%D0%BA%D0%B0), виданій у 1202 році, європейці змогли познайомитися з індо-арабськими цифрами, а також методами обчислень. З неї вони дізналися і про алгебру. Однак протягом наступних кількох століть математична активність пішла на спад. Весь звід математичних досліджень і знань тієї епохи відбив [Лука Пачолі](https://uk.wikipedia.org/wiki/%D0%9B%D1%83%D0%BA%D0%B0_%D0%9F%D0%B0%D1%87%D0%BE%D0%BB%D1%96) у 1494 році. В його працях написано, що ніяких алгебраїчних нововведень відкрито або придумано не було, все це вже є у [Леонардо](https://uk.wikipedia.org/wiki/%D0%9B%D0%B5%D0%BE%D0%BD%D0%B0%D1%80%D0%B4%D0%BE_%D0%B4%D0%B0_%D0%92%D1%96%D0%BD%D1%87%D1%96).

## Відродження

Одними з найвидатніших геометрів епохи [Відродження](https://uk.wikipedia.org/wiki/%D0%92%D1%96%D0%B4%D1%80%D0%BE%D0%B4%D0%B6%D0%B5%D0%BD%D0%BD%D1%8F), як не дивно, стали художники. Саме вони розвинули ідею [перспективи.](https://uk.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D1%81%D0%BF%D0%B5%D0%BA%D1%82%D0%B8%D0%B2%D0%B0) Поняття [проекції](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B5%D0%BA%D1%86%D1%96%D1%97) і [перерізу](https://uk.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D1%80%D1%96%D0%B7_(%D0%BA%D1%80%D0%B5%D1%81%D0%BB%D0%B5%D0%BD%D0%BD%D1%8F)) ввів художник [Леон Баттіста Альберті](https://uk.wikipedia.org/wiki/%D0%9B%D0%B5%D0%BE%D0%BD-%D0%91%D0%B0%D1%82%D1%82%D1%96%D1%81%D1%82%D0%B0_%D0%90%D0%BB%D1%8C%D0%B1%D0%B5%D1%80%D1%82%D1%96) (1404—1472 рр.). Всі прямі промені світла, що виходять від очей спостерігача до різних точок представленої сцени, утворюють [проекцію](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B5%D0%BA%D1%86%D1%96%D1%8F). А перетин виходить шляхом проходження площини через проекцію. Тому для того, щоб картина, яку малює художник, в кінцевому результаті була максимально реалістичною, вона повинна виконувати закони проекції і бути саме таким перетином. Судження про проекції і перерізи одразу викликали ряд математичних питань. Завдяки ним і народилася проективна геометрія, а заснував її [Ж. Дезарг](https://uk.wikipedia.org/wiki/%D0%96%D0%B5%D1%80%D0%B0%D1%80_%D0%94%D0%B5%D0%B7%D0%B0%D1%80%D0%B3) (1593—1662 рр.). Він створив її за допомогою доказів, які ґрунтувалися на проекції, а також перерізі. Він уніфікував підхід до різних типів конічних перерізів, які видатний геометр з Греції — [Аполлоній](https://uk.wikipedia.org/wiki/%D0%90%D0%BF%D0%BE%D0%BB%D0%BB%D0%BE%D0%BD%D1%96%D0%B9_%D0%9F%D0%B5%D1%80%D0%B7%D1%8C%D0%BA%D0%B8%D0%B9), розглядав завжди окремо.

Розвиток сучасної математики XVI століття в Західній Європі став визначним у досягненнях алгебри та арифметики. [1582](https://uk.wikipedia.org/wiki/1582) року іспанський король [Філіп II](https://uk.wikipedia.org/wiki/%D0%A4%D1%96%D0%BB%D1%96%D0%BF_II) заснував першу в Європі [математичну академію](https://uk.wikipedia.org/wiki/%D0%86%D1%81%D0%BF%D0%B0%D0%BD%D1%81%D1%8C%D0%BA%D0%B0_%D0%BA%D0%BE%D1%80%D0%BE%D0%BB%D1%96%D0%B2%D1%81%D1%8C%D0%BA%D0%B0_%D0%B0%D0%BA%D0%B0%D0%B4%D0%B5%D0%BC%D1%96%D1%8F_%D1%82%D0%BE%D1%87%D0%BD%D0%B8%D1%85,_%D1%84%D1%96%D0%B7%D0%B8%D1%87%D0%BD%D0%B8%D1%85_%D1%96_%D0%BF%D1%80%D0%B8%D1%80%D0%BE%D0%B4%D0%BD%D0%B8%D1%87%D0%B8%D1%85_%D0%BD%D0%B0%D1%83%D0%BA).

Математики ввели в ужиток десяткові [дроби](https://uk.wikipedia.org/wiki/%D0%94%D1%80%D1%96%D0%B1), а також правила арифметичних дій з ними. Справжній фурор викликав [Дж. Непер](https://uk.wikipedia.org/wiki/%D0%94%D0%B6%D0%BE%D0%BD_%D0%9D%D0%B5%D0%BF%D0%B5%D1%80), який у 1614 році винайшов [логарифми](https://uk.wikipedia.org/wiki/%D0%9B%D0%BE%D0%B3%D0%B0%D1%80%D0%B8%D1%84%D0%BC). Вже в кінці XVII століття склалося чітке розуміння логарифмів як показників ступенів з абсолютно будь-яким позитивним числом, але тільки не одиницею. У XVI столітті стали активно використовувати ірраціональні числа. [Б. Паскаль](https://uk.wikipedia.org/wiki/%D0%91%D0%BB%D0%B5%D0%B7_%D0%9F%D0%B0%D1%81%D0%BA%D0%B0%D0%BB%D1%8C) (1623—1662 рр..), а також [І. Барроу](https://uk.wikipedia.org/wiki/%D0%86%D1%81%D0%B0%D0%B0%D0%BA_%D0%91%D0%B0%D1%80%D1%80%D0%BE%D1%83) (1630—1677 рр.), який був вчителем [І. Ньютона](https://uk.wikipedia.org/wiki/%D0%86%D1%81%D0%B0%D0%B0%D0%BA_%D0%9D%D1%8C%D1%8E%D1%82%D0%BE%D0%BD) (1643—1727 рр.) і викладав у [Кембриджському університеті](https://uk.wikipedia.org/wiki/%D0%9A%D0%B5%D0%BC%D0%B1%D1%80%D0%B8%D0%B4%D0%B6%D1%81%D1%8C%D0%BA%D0%B8%D0%B9_%D1%83%D0%BD%D1%96%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%82%D0%B5%D1%82), заявив, що число корінь з двох, можна трактувати виключно як геометричну величину і більше ніяк. Але в той же час [Р. Декарт](https://uk.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BD%D0%B5_%D0%94%D0%B5%D0%BA%D0%B0%D1%80%D1%82) (1596—1650 рр.) і [Дж. Валліс](https://uk.wikipedia.org/wiki/%D0%94%D0%B6%D0%BE%D0%BD_%D0%92%D0%B0%D0%BB%D0%BB%D1%96%D1%81) (1616—1703 рр.) стверджували наступне: ірраціональні числа припустимі і без посилань на геометрію, тобто самі по собі. Однак у XVI столітті відновилися суперечки з приводу законності від'ємних чисел, а також комплексних чисел (Декарт їх назвав «уявними»), які виникали при розв'язуванні квадратних рівнянь. Незважаючи на доказову базу, ці числа були під підозрою аж до XVIII століття, незважаючи на те, що [Л. Ейлер](https://uk.wikipedia.org/wiki/%D0%9B%D0%B5%D0%BE%D0%BD%D0%B0%D1%80%D0%B4_%D0%95%D0%B9%D0%BB%D0%B5%D1%80) (1707—1783) ними користувався. Комплексні числа остаточно були визнані тільки в XIX столітті, після того, як математики того часу повністю ознайомилися з їх геометричним представленням.

## 16-19 століття

У XVI столітті італійські математики С. Даль Ферро (1465—1526 рр.), [Н. Тарталья](https://uk.wikipedia.org/wiki/%D0%9D%D1%96%D0%BA%D0%BA%D0%BE%D0%BB%D0%BE_%D0%A2%D0%B0%D1%80%D1%82%D0%B0%D0%BB%D1%8C%D1%8F) (1499—1577 рр.) і [Д. Кардано](https://uk.wikipedia.org/wiki/%D0%94%D0%B6%D0%B8%D1%80%D0%BE%D0%BB%D0%B0%D0%BC%D0%BE_%D0%9A%D0%B0%D1%80%D0%B4%D0%B0%D0%BD%D0%BE) (1501—1576 рр.) змогли знайти спільні рішення рівнянь третього, а також четвертого ступеня. Щоб їх алгебраїчні міркування були зрозумілими, а записи стали більш точними, було прийнято рішення ввести багато відомих сьогодні символів, таких як: «+», «–», "=", «>», «<» та інших. Одним з найбільш яскравих нововведень стало систематичне застосування французьким математиком [Ф. Вієтом](https://uk.wikipedia.org/wiki/%D0%A4%D1%80%D0%B0%D0%BD%D1%81%D1%83%D0%B0_%D0%92%D1%96%D1%94%D1%82) (1540—1603 рр.) букв, які позначали невідомі, а також постійні величини. Це нововведення дозволило знайти Вієту єдиний метод рішення рівнянь другого, третього і четвертого ступенів. Після того, як усе було знайдено, математики пішли далі, тобто до рівнянь вище четвертого ступеня. Над цим наполегливо працювали [Кардано](https://uk.wikipedia.org/wiki/%D0%9A%D0%B0%D1%80%D0%B4%D0%B0%D0%BD%D0%BE), [Ньютон](https://uk.wikipedia.org/wiki/%D0%86%D1%81%D0%B0%D0%B0%D0%BA_%D0%9D%D1%8C%D1%8E%D1%82%D0%BE%D0%BD) і [Декарт](https://uk.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BD%D0%B5_%D0%94%D0%B5%D0%BA%D0%B0%D1%80%D1%82). Вони опублікували, щоправда, без будь-яких доказів, цілий ряд своїх результатів, що стосуються числа і виду коренів рівняння. І. Ньютон відкрив співвідношення між коренем і [дискримінантом](https://uk.wikipedia.org/wiki/%D0%94%D0%B8%D1%81%D0%BA%D1%80%D0%B8%D0%BC%D1%96%D0%BD%D0%B0%D0%BD%D1%82_%D0%BA%D0%B2%D0%B0%D0%B4%D1%80%D0%B0%D1%82%D0%BD%D0%BE%D0%B3%D0%BE_%D1%80%D1%96%D0%B2%D0%BD%D1%8F%D0%BD%D0%BD%D1%8F) квадратного рівняння. [Фрідріх Гаусс](https://uk.wikipedia.org/wiki/%D0%9A%D0%B0%D1%80%D0%BB_%D0%A4%D1%80%D1%96%D0%B4%D1%80%D1%96%D1%85_%D0%93%D0%B0%D1%83%D1%81) (1777—1855 рр.) в 1779 році довів так звану [основну теорему алгебри](https://uk.wikipedia.org/wiki/%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D0%BD%D0%B0_%D1%82%D0%B5%D0%BE%D1%80%D0%B5%D0%BC%D0%B0_%D0%B0%D0%BB%D0%B3%D0%B5%D0%B1%D1%80%D0%B8), згідно з якою многочлен *n*-го ступеню має рівно *n* коренів. В алгебрі основне завдання полягає в наступному: знайти спільне рішення алгебраїчного рівняння. Це завдання продовжувало хвилювати математиків на початку XIX століття. Коли мова йде про спільне вирішення рівняння другого ступеня, мається на увазі наступне: кожен з двох коренів може бути виражений за допомогою кінцевого числа операцій додавання, віднімання, а також множення, ділення і добування коренів, здійснюваних над коефіцієнтами рівняння. [Нільс Абель](https://uk.wikipedia.org/wiki/%D0%9D%D1%96%D0%BB%D1%8C%D1%81_%D0%93%D0%B5%D0%BD%D1%80%D1%96%D0%BA_%D0%90%D0%B1%D0%B5%D0%BB%D1%8C) (молодий норвезький математик, 1802—1829 рр.) довів, що немає ніякої можливості отримати спільне рішення рівнянь вище четвертого ступеня за допомогою кінцевого числа алгебраїчних рішень. Але є багато рівнянь спеціального виду вище четвертого ступеня, які, в принципі, можуть допускати подібне рішення. Зовсім юний французький математик [Е. Галуа](https://uk.wikipedia.org/wiki/%D0%95%D0%B2%D0%B0%D1%80%D0%B8%D1%81%D1%82_%D0%93%D0%B0%D0%BB%D1%83%D0%B0) (1811—1832 рр.) буквально напередодні своєї дуелі на якій і загинув, зміг дати заключну відповідь на питання: які саме рівняння можна відобразити через коефіцієнти за допомогою кінцевого числа алгебраїчних операцій. У його теорії застосовувалися підстановки коренів. Розвиток теорії груп — це хороший приклад того, що в математиці все ж присутні і творчі процеси. Галуа створив свою теорію на основі робіт Абеля. Сам же Абель брав за основу роботи [Ж. Лагранжа](https://uk.wikipedia.org/wiki/%D0%96%D0%BE%D0%B7%D0%B5%D1%84-%D0%9B%D1%83%D1%97_%D0%9B%D0%B0%D0%B3%D1%80%D0%B0%D0%BD%D0%B6) (1736—1813 рр.). Насправді, дуже багато відомих математиків, включаючи і Гауса, і А. Лежандра (1752—1833 рр.) використовували в своїх працях поняття груп. У свій час Ньютон і заявив: «Якщо я і міг бачити набагато далі, ніж інші, все тільки тому, що я стояв на плечах гігантів».

## Аналітична геометрія

[Аналітична (координатна) геометрія](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B0%D0%BB%D1%96%D1%82%D0%B8%D1%87%D0%BD%D0%B0_%D0%B3%D0%B5%D0%BE%D0%BC%D0%B5%D1%82%D1%80%D1%96%D1%8F) створювалася незалежно математиками П. Ферма (1601—1655 рр.) і [Р. Декартом.](https://uk.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BD%D0%B5_%D0%94%D0%B5%D0%BA%D0%B0%D1%80%D1%82) Це було зроблено спеціально для розширення можливостей евклідової геометрії в задачах на побудову. Але Ферма оцінював свої роботи тільки як переформулювання творів [Аполлонія](https://uk.wikipedia.org/wiki/%D0%90%D0%BF%D0%BE%D0%BB%D0%BB%D0%BE%D0%BD%D1%96%D0%B9_%D0%9F%D0%B5%D1%80%D0%B7%D1%8C%D0%BA%D0%B8%D0%B9). Справжнє відкриття — це усвідомлення всієї могутності алгебраїчних методів, яке належить все ж Декарту. Аналітична геометрія виникла саме тоді, коли Декарт приступив до розгляду невизначених задач на побудову шляхом рішень, де є не одна, а відразу безліч різних довжин. Аналітична геометрія застосовує алгебраїчні рівняння, щоб представити дослідження поверхонь і кривих. Декарт вважав, що певну криву можна записати за допомогою єдиного алгебраїчного рівняння відносно *x* і *y*. Даний підхід став важливим кроком вперед. Таким чином, у XVII—XVIII ст. ст. більшість головних відкриттів, приміром, циклоїда або [ланцюгова лінія](https://uk.wikipedia.org/wiki/%D0%9B%D0%B0%D0%BD%D1%86%D1%8E%D0%B3%D0%BE%D0%B2%D0%B0_%D0%BB%D1%96%D0%BD%D1%96%D1%8F), швидко змогли увійти в побут вчених. Швидше за все, першим математиком, який використовував рівняння для доказів властивостей конічних перерізів, став [Дж. Валліс](https://uk.wikipedia.org/wiki/%D0%94%D0%B6%D0%BE%D0%BD_%D0%92%D0%B0%D0%BB%D0%BB%D1%96%D1%81) (1616—1703 рр.). У 1685 році він алгебраїчним методом отримав всі необхідні йому результати, які були представлені в книзі [«Начала»](https://uk.wikipedia.org/wiki/%D0%9D%D0%B0%D1%87%D0%B0%D0%BB%D0%B0_%D0%95%D0%B2%D0%BA%D0%BB%D1%96%D0%B4%D0%B0) [Евкліда](https://uk.wikipedia.org/wiki/%D0%95%D0%B2%D0%BA%D0%BB%D1%96%D0%B4). Саме аналітична геометрія змогла повністю обміняти ролями геометрію і алгебру. Видатний французький математик [Лагранж](https://uk.wikipedia.org/wiki/%D0%96%D0%BE%D0%B7%D0%B5%D1%84-%D0%9B%D1%83%D1%97_%D0%9B%D0%B0%D0%B3%D1%80%D0%B0%D0%BD%D0%B6) сказав: «Алгебра і геометрія, рухаючись своїми шляхами, лише уповільнюють свій прогрес. Однак, як тільки ці науки об'єднуються, вони починають позичати одна у одної життєві сили і можливості, які змушують їх обох рухатися величезними кроками вперед до досконалості».

## [Математичний аналіз](https://uk.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%BD%D0%B8%D0%B9_%D0%B0%D0%BD%D0%B0%D0%BB%D1%96%D0%B7)

Такі засновники сучасної науки, як [Ньютон](https://uk.wikipedia.org/wiki/%D0%86%D1%81%D0%B0%D0%B0%D0%BA_%D0%9D%D1%8C%D1%8E%D1%82%D0%BE%D0%BD), [Коперник](https://uk.wikipedia.org/wiki/%D0%9C%D0%B8%D0%BA%D0%BE%D0%BB%D0%B0%D0%B9_%D0%9A%D0%BE%D0%BF%D0%B5%D1%80%D0%BD%D0%B8%D0%BA), [Галілей](https://uk.wikipedia.org/wiki/%D0%93%D0%B0%D0%BB%D1%96%D0%BB%D0%B5%D0%BE_%D0%93%D0%B0%D0%BB%D1%96%D0%BB%D0%B5%D0%B9) і [Кеплер](https://uk.wikipedia.org/wiki/%D0%99%D0%BE%D0%B3%D0%B0%D0%BD%D0%BD_%D0%9A%D0%B5%D0%BF%D0%BB%D0%B5%D1%80), підходили до вивчення природи так само, як і до математики. Досліджуючи, таким чином, рух, ці великі математики змогли виробити таке фундаментальне поняття, як відношення між змінними і [функція](https://uk.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D1%96%D1%8F_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)). Таке завдання, як визначення миттєвих швидкостей зміни різних величин, цікавило практично всіх математиків XVII століття, у тому числі і [Барроу](https://uk.wikipedia.org/wiki/%D0%91%D0%B0%D1%80%D1%80%D0%BE%D1%83), [Декарта](https://uk.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BD%D0%B5_%D0%94%D0%B5%D0%BA%D0%B0%D1%80%D1%82), Валліса, і Ферма. Вони запропонували різні ідеї і методи, які були об'єднані в систематичний універсальний формальний спосіб, що використовувався [Ньютоном](https://uk.wikipedia.org/wiki/%D0%86%D1%81%D0%B0%D0%B0%D0%BA_%D0%9D%D1%8C%D1%8E%D1%82%D0%BE%D0%BD), а також [Г. Лейбніцем](https://uk.wikipedia.org/wiki/%D2%90%D0%BE%D1%82%D1%84%D1%80%D1%96%D0%B4_%D0%92%D1%96%D0%BB%D1%8C%D0%B3%D0%B5%D0%BB%D1%8C%D0%BC_%D0%9B%D0%B5%D0%B9%D0%B1%D0%BD%D1%96%D1%86) (1646—1716 рр.), які, до речі, були творцями [диференціального числення](https://uk.wikipedia.org/wiki/%D0%94%D0%B8%D1%84%D0%B5%D1%80%D0%B5%D0%BD%D1%86%D1%96%D0%B0%D0%BB%D1%8C%D0%BD%D0%B5_%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%BD%D1%8F). Але в розробці даного обчислення, математики постійно вели гарячі суперечки, з'ясовуючи, кому ж все-таки належить головна заслуга, і Ньютон постійно звинувачував Лейбніца в чистому плагіаті. З плином часу дослідження підтвердили, що Лейбніц не займався плагіатом, а навпаки, створив незалежно від Ньютона математичний аналіз. Через незгоду сторін миритися з ситуацією, обмін знаннями між математиками Англії і континентальної Європи «заморозився» на довгі роки. Хочеться відзначити, що в цій ситуації найбільше постраждала англійська сторона. Математики з Англії так і продовжували аналізувати в геометричному напрямку, тоді як математики з континентальної Європи, включаючи таких гігантів думки, як [Я. Бернуллі](https://uk.wikipedia.org/wiki/%D0%AF%D0%BA%D0%BE%D0%B1_%D0%91%D0%B5%D1%80%D0%BD%D1%83%D0%BB%D0%BB%D1%96) (1667—1748 рр.), Лагранжа і Ейлера, змогли досягти неймовірно високих результатів, дотримуючись аналітичного або алгебраїчного підходу.

## Сучасна математика

Створення диференціального й інтегрального числень ознаменувало початок «вищої математики». Методи математичного аналізу, на відміну від поняття межі, що лежить в його основі, виглядали чіткими і зрозумілими. Багато років математики, у тому числі Ньютон і Лейбніц, марно намагалися дати точне визначення поняттю межі. І все ж, незважаючи на численні сумніви в обґрунтованості математичного аналізу, він знаходив все більш широке застосування.  Диференціальне і інтегральне числення стали наріжними каменями математичного аналізу, який з часом включив в себе і такі предмети, як теорія диференціальних рівнянь, звичайних і з частковими похідними, нескінченні ряди, варіаційне числення, диференціальна геометрія і багато іншого. Строге визначення межі вдалося отримати лише в 19-му столітті.

Неевклідова геометрія. До 1800 років, математика лежала на двох «китах», а саме  — на числовій системі і евклідовій геометрії. Так як багато властивостей числової системи доводили геометрично, евклідова геометрія була найбільш надійною частиною будівлі математики. Тим не менш, аксіома про паралельні прямі містила твердження про прямі, що тягнуться у нескінченність, яке не могло бути підтверджено досвідом. Навіть версія цієї аксіоми, що належить самому Евкліду, зовсім не стверджувала, що якісь прямі не перетнуться. У ній швидше формулювалася умова, при якій вони перетнуться в деякій кінцевій точці. Математики століттями намагалися знайти аксіомі про паралельні прямі відповідну заміну. Але в кожному варіанті неодмінно, у когось, був пробіл. Честь створення неевклідової геометрії випала [Лобачевському М. І.](https://uk.wikipedia.org/wiki/%D0%9B%D0%BE%D0%B1%D0%B0%D1%87%D0%B5%D0%B2%D1%81%D1%8C%D0%BA%D0%B8%D0%B9_%D0%9C%D0%B8%D0%BA%D0%BE%D0%BB%D0%B0_%D0%86%D0%B2%D0%B0%D0%BD%D0%BE%D0%B2%D0%B8%D1%87) та [Я. Бояї](https://uk.wikipedia.org/wiki/%D0%AF%D0%BD%D0%BE%D1%88_%D0%91%D0%BE%D1%8F%D1%97), кожен з яких незалежно опублікував свій власний оригінальний виклад неевклідової геометрії. У їх геометріях, через дану точку можна було провести нескінченно багато паралельних прямих. В геометрії [Б. Рімана](https://uk.wikipedia.org/wiki/%D0%91%D0%B5%D1%80%D0%BD%D0%B3%D0%B0%D1%80%D0%B4_%D0%A0%D1%96%D0%BC%D0%B0%D0%BD) через точку поза прямою не можна провести ні однієї паралельної.

Про фізичні додатки неевклідової геометрії ніхто серйозно не думав. Але, створення в 1915 році [А. Ейнштейном](https://uk.wikipedia.org/wiki/%D0%90%D0%BB%D1%8C%D0%B1%D0%B5%D1%80%D1%82_%D0%95%D0%B9%D0%BD%D1%88%D1%82%D0%B5%D0%B9%D0%BD) загальної теорії відносності, пробудило науковий світ до усвідомлення реальності неевклідової геометрії.

Неевклідова геометрія стала найбільш вражаючим інтелектуальним звершенням 19-го століття. Вона ясно продемонструвала, що математику не можна більше розглядати, як звід незаперечних істин. У кращому випадку, математика могла гарантувати достовірність доказів на основі не достовірних аксіом. Проте, математики надалі здобули свободу досліджувати будь-які ідеї, які могли здаватися їм привабливими. Кожен математик окремо був тепер вільний вводити свої власні нові поняття і встановлювати аксіоми на свій розсуд, стежачи лише за тим, щоб теореми, які виникають з аксіом, не суперечили одна одній. Грандіозне розширення кола математичних досліджень в кінці 19-го століття, по суті, стало наслідком цієї нової свободи.

Математична строгість. Приблизно до 1870 року, математики перебували в переконанні, що діють за визначенням древніх греків, застосовуючи дедуктивні міркування до математичних аксіом, тим самим забезпечуючи своїми висновками не меншу надійність, аніж та, якою володіли аксіоми. Неевклідова геометрія і кватерніони (алгебра, в якій не виконується властивість комутативності) змусили математиків усвідомити, що те, що вони брали за абстрактні і логічно несуперечливі затвердження, в дійсності ґрунтується на емпіричному та прагматичному базисі.

Створення неевклідової геометрії супроводжувалося також усвідомленням існування в евклідової геометрії логічних прогалин. Одним з недоліків евклідової геометрії було використання припущень, не сформульованих в явному вигляді. Мабуть, Евклід не піддавав сумніву ті властивості, якими володіли його геометричні фігури, але ці властивості не були включені в його аксіоми. Окрім того, доводячи подобу двох трикутників, Евклід скористався накладенням одного трикутника на інший, не явно припускаючи, що при русі, властивості фігур не змінюються. Але окрім таких логічних прогалин, виявилося і кілька помилкових доказів.

Створення нових алгебр, що почалося з квартерніонов, породило аналогічні сумніви і стосовно логічної обґрунтованості арифметики і алгебри звичайної числової системи. Всі раніше відомі математикам числа мали властивістю комутативності, тобто ab = ba. Кватерніони, вчинили переворот у традиційних уявленнях про числа, були відкриті в 1843 році [В. Гамільтоном](https://uk.wikipedia.org/wiki/%D0%92%D1%96%D0%BB%D1%8C%D1%8F%D0%BC_%D0%A0%D0%BE%D0%B2%D0%B5%D0%BD_%D0%93%D0%B0%D0%BC%D1%96%D0%BB%D1%8C%D1%82%D0%BE%D0%BD). Вони виявилися корисними для вирішення цілого ряду фізичних і геометричних проблем, хоча для кватерніонів не виконувалося властивість комутативності. Квартерніони змусили математиків усвідомити, що якщо не вважати присвяченій цілим числам і далекою від досконалості частини евклідових, арифметика і алгебра не мають власної аксіоматичної основи. Математики вільно поводилися з негативними та комплексними числами й виконували алгебраїчні операції, керуючись лише тим, що вони успішно працюють. Логічна строгість поступилася місцем демонстрації практичної користі введення сумнівних понять і процедур.

Майже з самого зародження математичного аналізу, неодноразово робилися спроби підвести під нього суворі підстави. Математичний аналіз ввів два нових складних поняття — похідна і визначений інтеграл. Над цими поняттями билися Ньютон і Лейбніц, а також математики наступних поколінь, які перетворили диференціальне та інтегральне числення в математичний аналіз. Однак, незважаючи на всі зусилля, в поняттях межі, безперервності і диференційовності залишалося багато неясного. Крім того, з'ясувалося, що властивості алгебраїчних функцій не можна перенести на всі інші функції. Майже всі математики 18-го століття — початку 19-го століття, робили зусилля, щоб знайти сувору основу для математичного аналізу, і всі вони зазнали невдачі. Нарешті, в 1821 році, [О. Коші](https://uk.wikipedia.org/wiki/%D0%9E%D2%91%D1%8E%D1%81%D1%82%D0%B5%D0%BD-%D0%9B%D1%83%D1%97_%D0%9A%D0%BE%D1%88%D1%96), використовуючи поняття числа, підвів сувору базу під весь математичний аналіз. Однак пізніше, математики виявили у О. Коші логічні пробіли. Бажана строгість була нарешті досягнута в 1859 році [К. Вейерштрасом](https://uk.wikipedia.org/wiki/%D0%9A%D0%B0%D1%80%D0%BB_%D0%92%D0%B5%D1%94%D1%80%D1%88%D1%82%D1%80%D0%B0%D1%81).

К. Вейерштрасс, спочатку вважав властивості дійсних і комплексних чисел самоочевидними. Пізніше він, як і [Г. Кантор](https://uk.wikipedia.org/wiki/%D0%93%D0%B5%D0%BE%D1%80%D0%B3_%D0%9A%D0%B0%D0%BD%D1%82%D0%BE%D1%80) і [Р. Дедекінд](https://uk.wikipedia.org/wiki/%D0%A0%D1%96%D1%85%D0%B0%D1%80%D0%B4_%D0%94%D0%B5%D0%B4%D0%B5%D0%BA%D1%96%D0%BD%D0%B4), усвідомив необхідність побудови теорії ірраціональних чисел. Вони дали коректне визначення ірраціональних чисел і встановили їх властивості, однак властивості раціональних чисел, як і раніше вважали самоочевидними. Нарешті, логічна структура теорії дійсних і комплексних чисел придбала свій закінчений вигляд в роботах Р. Дедекінда і [Дж. Пеано](https://uk.wikipedia.org/wiki/%D0%94%D0%B6%D1%83%D0%B7%D0%B5%D0%BF%D0%BF%D0%B5_%D0%9F%D0%B5%D0%B0%D0%BD%D0%BE). Створення підстав числової системи дало змогу вирішити також проблеми обґрунтування алгебри.

Завдання посилення строгості формулювань евклідової геометрії було порівняно простим і зводилося до перерахування визначених термінів, уточнення визначень, введення відсутніх аксіом і заповнення прогалин у доказах. Це завдання виконав у 1899 році [Д. Гільберт](https://uk.wikipedia.org/wiki/%D0%94%D0%B0%D0%B2%D0%B8%D0%B4_%D0%93%D1%96%D0%BB%D1%8C%D0%B1%D0%B5%D1%80%D1%82). Майже в той же час були закладені і основи інших геометрій. Д. Гільберт сформулював концепцію формальної аксіоматики. Одна з особливостей запропонованого ним підходу — трактування невизначених термінів: під ними можна мати на увазі будь-які об'єкти, що задовольняють аксіоми. Наслідком цієї особливості стала зростаюча абстрактність сучасної математики. Евклідова і неевклідова геометрії описують фізичний простір. Але в топології, що є узагальненням геометрії, невизначений термін «точка» може бути вільний від геометричних асоціацій. Для топології «точкою» може бути функція або послідовність чисел так само, як і що-небудь інше. Абстрактний простір являє собою багато таких «точок».

Аксіоматичний метод Д. Гільберта увійшов майже в усі розділи математики 20-го століття. Однак, незабаром стало ясно, що цьому методу притаманні певні обмеження. У 1880-х роках, Г. Кантор спробував систематично класифікувати нескінченні множини (наприклад, множина всіх раціональних чисел, множина дійсних чисел і т. д.) шляхом їх порівняльної кількісної оцінки, приписуючи їм т. зв. [трансфінітні числа](https://uk.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D1%81%D1%84%D1%96%D0%BD%D1%96%D1%82%D0%BD%D0%B5_%D1%87%D0%B8%D1%81%D0%BB%D0%BE). При цьому, він виявив в теорії множин протиріччя. Таким чином, до початку 20-го століття, математикам довелося мати справу з проблемою їх дозволу, а також з іншими проблемами підстав їх науки, такими, як неявне використання т. зв. аксіоми вибору.

І все ж ніщо не могло зрівнятися з руйнівним впливом теореми неповноти [К. Геделя](https://uk.wikipedia.org/wiki/%D0%9A%D1%83%D1%80%D1%82_%D0%93%D0%B5%D0%B4%D0%B5%D0%BB%D1%8C). Ця теорема стверджує, що будь-яка несуперечлива формальна система, що досить багата, щоб утримувати теорію чисел, обов'язково містить нерозв'язну пропозицію, тобто твердження, яке неможливо ні довести, ні спростувати в її рамках. Тепер загальновизнано, що абсолютного доказу в математиці не існує. Щодо того, що ж таке доказ, думки розходяться. Однак більшість математиків схильні вважати, що проблеми основ математики є філософськими. І справді, жодна теорема не змінилася внаслідок знову знайдених логічно строгих структур. Це засвідчує, що в основі математики лежить не логіка, а здорова інтуїція.

Якщо математику, відому до 1600, можна охарактеризувати як елементарну, то порівняно з тим, що було створено пізніше, ця елементарна математика нескінченно мала. Розширилися старі галузі та з'явилися нові, як чисті, так і прикладні галузі математичних знань. Виходять близько 500 математичних журналів. Величезна кількість публікованих результатів не дає змоги навіть фахівцеві ознайомитися з усім, що відбувається в тій галузі, в якій він працює, не кажучи вже про те, що багато результатів доступні для розуміння лише фахівцям вузького профілю. Ні один математик сьогодні не може сподіватися знати більше від того, що відбувається в дуже маленькому куточку науки.

## 1.2.Калькулятори

**Калькулятор**

1. Електронно-обчислювальний пристрій для виконання операцій над числами або алгебраїчними формулами;
2. [Комп'ютерна програма](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%27%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B0), яка емулює функції калькулятора.
3. Спеціалізована програма, яка автоматично проводить деякий вид розрахунків
4. Застаріла [професія](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%84%D0%B5%D1%81%D1%96%D1%8F) (людина, що здійснювала [калькуляцію](https://uk.wikipedia.org/wiki/%D0%9A%D0%B0%D0%BB%D1%8C%D0%BA%D1%83%D0%BB%D1%8F%D1%86%D1%96%D1%8F)).

В [СРСР](https://uk.wikipedia.org/wiki/%D0%A1%D0%A0%D0%A1%D0%A0) для позначення малогабаритного обчислювального пристрою використовувався термін «мікрокалькулятор», вперше застосований у 1973-му році для мікрокалькулятора «Електроніка Б3-04». Просто «калькуляторами» називали великі за розміром настільні обчислювальні пристрої. І настільні, і мікрокалькулятори офіційно називалися *електронними клавішними обчислювальними машинами* ([рос.](https://uk.wikipedia.org/wiki/%D0%A0%D0%BE%D1%81%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) *«ЭКВМ — электронные клавишные вычислительные машины»*).

Сьогодні, у зв'язку з тим, що в англійській мові використовується тільки термін «калькулятор» ([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) *Calculator*), термін «мікрокалькулятор» поступово виходить з обігу.

Сучасні калькулятори є [електронними пристроями](https://uk.wikipedia.org/wiki/%D0%95%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%B8%D0%B9_%D0%BF%D1%80%D0%B8%D1%81%D1%82%D1%80%D1%96%D0%B9). У минулому для математичних обчислень використовувалися [абак](https://uk.wikipedia.org/wiki/%D0%90%D0%B1%D0%B0%D0%BA_(%D1%80%D0%B0%D1%85%D1%96%D0%B2%D0%BD%D0%B8%D1%86%D1%8F)), [рахівниця](https://uk.wikipedia.org/wiki/%D0%A0%D0%B0%D1%85%D1%96%D0%B2%D0%BD%D0%B8%D1%86%D1%8F), [математичні таблиці](https://uk.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%BD%D1%96_%D1%82%D0%B0%D0%B1%D0%BB%D0%B8%D1%86%D1%96) (особливо таблиці логарифмів), [логарифмічні лінійки](https://uk.wikipedia.org/wiki/%D0%9B%D0%BE%D0%B3%D0%B0%D1%80%D0%B8%D1%84%D0%BC%D1%96%D1%87%D0%BD%D0%B0_%D0%BB%D1%96%D0%BD%D1%96%D0%B9%D0%BA%D0%B0) і механічні або електромеханічні [арифмометри](https://uk.wikipedia.org/wiki/%D0%90%D1%80%D0%B8%D1%84%D0%BC%D0%BE%D0%BC%D0%B5%D1%82%D1%80).

Популярними сьогодні є калькулятори, вбудовані в [персональні комп'ютери](https://uk.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D1%81%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%96_%D0%BA%D0%BE%D0%BC%D0%BF%27%D1%8E%D1%82%D0%B5%D1%80%D0%B8), [стільникові телефони](https://uk.wikipedia.org/wiki/%D0%A1%D1%82%D1%96%D0%BB%D1%8C%D0%BD%D0%B8%D0%BA%D0%BE%D0%B2%D0%B8%D0%B9_%D1%82%D0%B5%D0%BB%D0%B5%D1%84%D0%BE%D0%BD), [КПК](https://uk.wikipedia.org/wiki/%D0%9A%D0%B8%D1%88%D0%B5%D0%BD%D1%8C%D0%BA%D0%BE%D0%B2%D0%B8%D0%B9_%D0%BF%D0%B5%D1%80%D1%81%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%B8%D0%B9_%D0%BA%D0%BE%D0%BC%D0%BF%27%D1%8E%D1%82%D0%B5%D1%80) і навіть [наручний годинник](https://uk.wikipedia.org/wiki/%D0%9D%D0%B0%D1%80%D1%83%D1%87%D0%BD%D0%B8%D0%B9_%D0%B3%D0%BE%D0%B4%D0%B8%D0%BD%D0%BD%D0%B8%D0%BA).

## Типи калькуляторів

### Калькулятор — універсальний обчислювальний пристрій

* **Найпростіші** калькулятори мають невеликі розміри і вагу, один регістр пам'яті і невелике число функцій. Призначені для широкого кола споживачів.
* **Бухгалтерські** калькулятори мають додаткові засоби для роботи з грошовими сумами (кнопки «00» і «000», фіксована кількість розрядів дробової частини, автоматичне округлення). Як правило, мають настільні габарити. Призначені для всіх, хто змушений рахувати гроші: [бухгалтерів](https://uk.wikipedia.org/wiki/%D0%91%D1%83%D1%85%D0%B3%D0%B0%D0%BB%D1%82%D0%B5%D1%80), [касирів](https://uk.wikipedia.org/wiki/%D0%9A%D0%B0%D1%81%D0%B8%D1%80) і т. д.
* **Інженерні** ([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) *Scientific*): призначені для складних наукових та інженерних розрахунків. Мають велику кількість функцій, включаючи обчислення всіх елементарних функцій, статистичні розрахунки, задання [кутів](https://uk.wikipedia.org/wiki/%D0%9A%D1%83%D1%82) в [градусах](https://uk.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D0%B4%D1%83%D1%81_(%D0%B3%D0%B5%D0%BE%D0%BC%D0%B5%D1%82%D1%80%D1%96%D1%8F)), [хвилинах](https://uk.wikipedia.org/wiki/%D0%A5%D0%B2%D0%B8%D0%BB%D0%B8%D0%BD%D0%B0) і [секундах](https://uk.wikipedia.org/wiki/%D0%A1%D0%B5%D0%BA%D1%83%D0%BD%D0%B4%D0%B0_(%D0%B3%D0%B5%D0%BE%D0%BC%D0%B5%D1%82%D1%80%D1%96%D1%8F)). Для складних обчислень застосовуються дужки або [зворотний польський запис](https://uk.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D1%8C%D1%81%D1%8C%D0%BA%D0%B8%D0%B9_%D1%96%D0%BD%D0%B2%D0%B5%D1%80%D1%81%D0%BD%D0%B8%D0%B9_%D0%B7%D0%B0%D0%BF%D0%B8%D1%81).
* **Візуальні** калькулятори дозволяють вводити довгий вираз і редагувати його. При натисканні кнопки «=» відбувається обчислення значення цього виразу. Такі калькулятори дорогі і трохи незручні для простих розрахунків, проте зручні, коли потрібно провести велику кількість однотипних розрахунків з різними аргументами.
* **Програмовані** калькулятори мають можливість вводити і виконувати програми користувача. Мають велику кількість регістрів пам'яті (10 і більше). По функціональності наближаються до найпростіших комп'ютерів. В залежності від моделі програмовані калькулятори мають кілька типів вбудованих мов програмування: найпростіша (запам'ятовує тільки натискання клавіш без виведення на екран); така, що відображає коди команд (зазвичай використовуючи зворотний польський запис), [Бейсік](https://uk.wikipedia.org/wiki/%D0%91%D0%B5%D0%B9%D1%81%D1%96%D0%BA) або власна [мова 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), наближена до Бейсік.
* **Графічні** калькулятори мають графічний екран, що дозволяє будувати [графіки функцій](https://uk.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D1%96%D0%BA%D0%B8_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D1%96%D0%B9) або навіть виводити на екран довільні малюнки.

## Історія

[](https://uk.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB:Electronila_mk-59_1.JPG)

[Електроніка МК-59](https://uk.wikipedia.org/wiki/%D0%95%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D1%96%D0%BA%D0%B0_%D0%9C%D0%9A-59), радянський настільний калькулятор 1980-х років, що вироблявся в [Івано-Франківську](https://uk.wikipedia.org/wiki/%D0%86%D0%B2%D0%B0%D0%BD%D0%BE-%D0%A4%D1%80%D0%B0%D0%BD%D0%BA%D1%96%D0%B2%D1%81%D1%8C%D0%BA).

* [1954](https://uk.wikipedia.org/wiki/1954): Фірма [IBM](https://uk.wikipedia.org/wiki/IBM) продемонструвала перший повністю [транзисторний](https://uk.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D0%B7%D0%B8%D1%81%D1%82%D0%BE%D1%80) калькулятор.
* [1957](https://uk.wikipedia.org/wiki/1957): IBM почала випуск перших комерційних калькуляторів на транзисторах (IBM 608).
* [1963](https://uk.wikipedia.org/wiki/1963): Розпочато випуск першого масового калькулятора — ANITA MK VIII (Англія, на газорозрядних лампах, повна клавіатура для введення числа + десять клавіш для введення множника).
* [1964](https://uk.wikipedia.org/wiki/1964): Розпочато випуск першого масового повністю транзисторного калькулятора — FRIDEN 130 (США, 4 регістра, використовувалася [«зворотна польська нотація»](https://uk.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D1%8C%D1%81%D1%8C%D0%BA%D0%B8%D0%B9_%D1%96%D0%BD%D0%B2%D0%B5%D1%80%D1%81%D0%BD%D0%B8%D0%B9_%D0%B7%D0%B0%D0%BF%D0%B8%D1%81)). Розпочато випуск першого серійного вітчизняного калькулятора «Вега».
* [1965](https://uk.wikipedia.org/wiki/1965): Компанія Wang Laboratories випустила калькулятор Wang LOCI-2, який міг обчислювати [логарифми](https://uk.wikipedia.org/wiki/%D0%9B%D0%BE%D0%B3%D0%B0%D1%80%D0%B8%D1%84%D0%BC).
* [1969](https://uk.wikipedia.org/wiki/1969): Випущено перший настільний програмований калькулятор — HP 9100A (США, транзисторний, використовувалася «зворотня польська нотація»).
* [1970](https://uk.wikipedia.org/wiki/1970): Почався продаж калькуляторів, які можна тримати в руці (фірми [Sharp](https://uk.wikipedia.org/wiki/Sharp) і [Canon](https://uk.wikipedia.org/wiki/Canon), вага калькуляторів близько 800 г). Перший вітчизняний калькулятор, виконаний з використанням [інтегральних мікросхем](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0_%D0%BC%D1%96%D0%BA%D1%80%D0%BE%D1%81%D1%85%D0%B5%D0%BC%D0%B0) — Іскра 110.
* [1971](https://uk.wikipedia.org/wiki/1971): З'явився перший кишеньковий калькулятор — модель 901B фірми Bomwar розміром 131 × 77 × 37 мм, з 4 операціями та 8-розрядним «червоним» індикатором (на світлодіодах); ціна $ 240.
* [1972](https://uk.wikipedia.org/wiki/1972): З'явився перший інженерний калькулятор — HP-35 фірми [Hewlett-Packard](https://uk.wikipedia.org/wiki/Hewlett-Packard).
* [1974](https://uk.wikipedia.org/wiki/1974): Перший вітчизняний мікрокалькулятор — «Електроніка Б3-04» (вперше використаний термін «мікрокалькулятор»).
* [1975](https://uk.wikipedia.org/wiki/1975): З'явився калькулятор HP-25C, в якому програми і дані не зникали при вимиканні живлення.
* [1977](https://uk.wikipedia.org/wiki/1977): Розроблений перший радянський кишеньковий програмований мікрокалькулятор «Електроніка Б3-21».
* [1979](https://uk.wikipedia.org/wiki/1979): Hewlett Packard випустила перший калькулятор з алфавітно-цифровим індикатором — HP-41C. Він був програмованим, з можливістю підключення додаткових модулів — RAM, ROM, пристрою читання штрих-кодів, касети з магнітною стрічкою, флоппі-дисків, принтерів, роз'ємів [RS-232](https://uk.wikipedia.org/wiki/RS-232), HP-IL, [HP-IB](https://uk.wikipedia.org/wiki/HP-IB).
* [1980](https://uk.wikipedia.org/wiki/1980): З'явився [Б3-34](https://uk.wikipedia.org/wiki/%D0%95%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D1%96%D0%BA%D0%B0_%D0%913-34).
* [1985](https://uk.wikipedia.org/wiki/1985): З'явилися МК-61 і МК-52.
* [1985](https://uk.wikipedia.org/wiki/1985): З'явився перший програмований калькулятор з графічним дисплеєм Casio FX-7000G.
* [2007](https://uk.wikipedia.org/wiki/2007): З'явився останній (на сьогоднішній день) російський калькулятор МК-152.

# РОЗДІЛ 2. ВИБІР ЗАСОБІВ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

## 2.1. Мова програмування C#

**C#** (вимовляється *Сі-шарп*) — [об'єктно-орієнтована мова програмування](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) з безпечною [системою типізації](https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%82%D0%B8%D0%BF%D1%96%D0%B7%D0%B0%D1%86%D1%96%D1%97) для платформи [.NET](https://uk.wikipedia.org/wiki/.NET). Розроблена [Андерсом Гейлсбергом](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B4%D0%B5%D1%80%D1%81_%D0%93%D0%B5%D0%B9%D0%BB%D1%81%D0%B1%D0%B5%D1%80%D0%B3), [Скотом Вілтамутом](https://uk.wikipedia.org/w/index.php?title=%D0%92%D1%96%D0%BB%D1%82%D0%B0%D0%BC%D1%83%D1%82_%D0%A1%D0%BA%D0%BE%D1%82&action=edit&redlink=1) та [Пітером Гольде](https://uk.wikipedia.org/w/index.php?title=%D0%93%D0%BE%D0%BB%D1%8C%D0%B4%D0%B5_%D0%9F%D1%96%D1%82%D0%B5%D1%80&action=edit&redlink=1) під егідою [Microsoft Research](https://uk.wikipedia.org/wiki/Microsoft_Research) (належить [Microsoft](https://uk.wikipedia.org/wiki/Microsoft)).

[Синтаксис](https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%81) C# близький до [С++](https://uk.wikipedia.org/wiki/%D0%A1%2B%2B) і [Java](https://uk.wikipedia.org/wiki/Java). [Мова](https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B2%D0%B0) має строгу статичну типізацію, підтримує [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)), перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, [винятки](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%BE%D0%B1%D0%BA%D0%B0_%D0%B2%D0%B8%D0%BD%D1%8F%D1%82%D0%BA%D1%96%D0%B2), коментарі у форматі [XML](https://uk.wikipedia.org/wiki/XML). Перейнявши багато від своїх попередників — мов [С++](https://uk.wikipedia.org/wiki/%D0%A1%2B%2B), [Object Pascal](https://uk.wikipedia.org/wiki/Object_Pascal), [Модула](https://uk.wikipedia.org/w/index.php?title=%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D0%B0&action=edit&redlink=1) і [Smalltalk](https://uk.wikipedia.org/wiki/Smalltalk) — С#, спираючись на практику їхнього [використання](https://uk.wikipedia.org/wiki/%D0%92%D0%B8%D0%BA%D0%BE%D1%80%D0%B8%D1%81%D1%82%D0%B0%D0%BD%D0%BD%D1%8F), виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад, мова С#, на відміну від C++, не передбачає [множинне успадкування](https://uk.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B8%D0%BD%D0%BD%D0%B5_%D1%83%D1%81%D0%BF%D0%B0%D0%B4%D0%BA%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) класів.

## Історія виникнення

C# є дуже близьким родичем мови [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) [Java](https://uk.wikipedia.org/wiki/Java). Мова Java була створена компанією [Sun Microsystems](https://uk.wikipedia.org/wiki/Sun_Microsystems), коли глобальний розвиток інтернету поставив завдання [розподілених обчислень](https://uk.wikipedia.org/wiki/%D0%A0%D0%BE%D0%B7%D0%BF%D0%BE%D0%B4%D1%96%D0%BB%D0%B5%D0%BD%D1%96_%D0%BE%D0%B1%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%BD%D1%8F). Взявши за основу популярну мову C++, Java виключила з неї потенційно небезпечні речі (на зразок вказівників без контролю виходу за межі). Для розподілених обчислень була створена концепція віртуальної машини та машинно-незалежного байт-коду, свого роду посередника між вихідним текстом програм і апаратними інструкціями комп'ютера чи іншого інтелектуального пристрою.

Java набула чималої популярності, і була ліцензована також і компанією [Microsoft](https://uk.wikipedia.org/wiki/Microsoft). Але, з часом, Sun почала звинувачувати Microsoft, що та при створенні свого клону Java робить її сумісною виключно з платформою [Windows](https://uk.wikipedia.org/wiki/Windows), чим суперечить самій концепції машинно-незалежного середовища виконання і порушує ліцензійну угоду. Microsoft відмовилася піти назустріч вимогам Sun, і тому з'ясування стосунків набуло статусу [судового](https://uk.wikipedia.org/wiki/%D0%A1%D1%83%D0%B4) процесу. Суд визнав позицію Sun справедливою, і зобов'язав Microsoft відмовитися від позаліцензійного використання Java.

У цій ситуації в Microsoft вирішили, користуючись своїм панівним становищем на ринку, створити свій власний аналог Java — мову, для якої корпорація буде повноцінним власником. Ця новостворена мова отримала назву C#. Вона успадкувала від Java концепції віртуальної машини (середовище .NET), байт-коду (MSIL) і більшої безпеки вихідного коду програм, плюс врахувала досвід використання програм на Java.

Нововведенням C# стала можливість легшої взаємодії, порівняно з мовами-попередниками, з кодом програм, написаних на інших мовах, що є важливим при створенні великих проєктів. Якщо програми на різних мовах виконуються на платформі [.NET](https://uk.wikipedia.org/wiki/.NET), .NET бере на себе клопіт щодо сумісності програм (тобто типів даних, за кінцевим рахунком).

Станом на сьогодні[[*коли?*](https://uk.wikipedia.org/wiki/%D0%92%D1%96%D0%BA%D1%96%D0%BF%D0%B5%D0%B4%D1%96%D1%8F:%D0%A1%D0%BB%D0%BE%D0%B2%D0%B0-%D0%BF%D0%B0%D1%80%D0%B0%D0%B7%D0%B8%D1%82%D0%B8#%D0%92%D0%B8%D1%80%D0%B0%D0%B7%D0%B8,_%D0%BF%D0%BE%D0%B2'%D1%8F%D0%B7%D0%B0%D0%BD%D1%96_%D0%B7_%D1%87%D0%B0%D1%81%D0%BE%D0%BC)] C# визначено флагманською мовою корпорації Microsoft, бо вона найповніше використовує нові можливості .NET. Решта мов програмування, хоч і підтримуються, але визнані такими, що мають спадкові прогалини щодо використання .NET. Рядок в C# є посилальним типом.

## Назва мови

Символ # у назві мови можна інтерпретувати і як дві пари плюсів ++, що натякають на новий крок у розвитку мови порівняно з C++ (подібно до кроку від C до C++), і як музичний символ [дієз](https://uk.wikipedia.org/wiki/%D0%94%D1%96%D1%94%D0%B7), разом з буквою C, що становить в англійській мові назву ноти [до-дієз](https://uk.wikipedia.org/w/index.php?title=%D0%94%D0%BE-%D0%B4%D1%96%D1%94%D0%B7&action=edit&redlink=1). Останнє й дало назву мові. Попри те, що [символ #](https://uk.wikipedia.org/wiki/%D0%9E%D0%BA%D1%82%D0%BE%D1%82%D0%BE%D1%80%D0%BF) (октоторп) насправді є символом для позначення номера на більшості клавіатур і відрізняється від символу дієз ♯ (Unicode U+266F), Microsoft, як автор мови, неодноразово зверталася до своїх клієнтів з проханням прийняти таку стилізацію.

## Версії

### Версія 1.0

[Проєкт](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%94%D0%BA%D1%82) C# був початий в грудні 1998 і отримав кодову назву COOL (C-style Object Oriented Language). Версія 1.0 була анонсована разом з платформою .NET у червні 2000 року, тоді ж з'явилася і перша загальнодоступна бета-версія; C# 1.0 остаточно вийшов разом з Microsoft Visual Studio .NET у лютому 2002 року.

Перша версія C# нагадувала за своїми можливостями Java 1.4, дещо їх розширюючи: так, в C# були властивості (що виглядають у коді як поля об'єкта, але, при зверненні до них, можуть викликати пов'язані методи класу), індексатори (подібні до властивостей, але приймають параметр як індекс масиву), події, делегати, цикли *foreach*, структури, що передаються за значенням, автоматичне перетворення вбудованих типів в об'єкти при необхідності (boxing), атрибути, вбудовані засоби взаємодії з некерованим кодом (DLL, COM) тощо. Крім того, в C# вирішено було перенести деякі можливості C++, відсутні в Java: беззнакові типи, перевизначення операцій (з деякими обмеженнями, на відміну від C++), передача параметрів у метод за посиланням, методи зі змінним числом параметрів, оператор *goto*. Також у C# залишили обмежену можливість роботи з вказівниками — в місцях коду, спеціально позначених словом *unsafe* і при вказівці спеціальної опції компілятору.

### Версія 2.0

Проєкт специфікації C# 2.0 вперше був викладений Microsoft в жовтні 2003 року; у 2004 році виходили бета-версії (проєкт з кодовою назвою Whidbey), C# 2.0 остаточно вийшов 7 листопада 2005 року разом з [Visual Studio](https://uk.wikipedia.org/wiki/Visual_Studio) 2005 і .NET 2.0.

**Нові можливості у версії 2.0**:

* Часткові типи (розділення реалізації класу більш ніж на один файл).
* Узагальнені, або параметризовані типи (generics, «дженерики»). На відміну від шаблонів C++, вони підтримують деякі додаткові можливості і працюють на рівні віртуальної машини. Разом з тим, параметрами узагальненого типу не можуть бути вирази.
* Нова форма ітератора, що дозволяє створювати співпрограми за допомогою ключового слова *yield*, подібно [Python](https://uk.wikipedia.org/wiki/Python) і [Рубі](https://uk.wikipedia.org/wiki/%D0%A0%D1%83%D0%B1%D1%96).
* Анонімні методи, що забезпечують функціональність замикання.
* Оператор ??: return obj1 ?? obj2; означає (у нотації C# 1.0) return obj1!=null ? obj1 : obj2;.
* Типи-значення, що обнуляються (nullable), (що позначаються знаком питання, наприклад, int? i = null;) є тими ж самими типами-значеннями, що можуть також приймати також значення null. Такі типи дозволяють поліпшити взаємодію з базами даних через мову SQL.

### Версія 3.0

В червні 2004 року [Андерс Гейлсберг](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B4%D0%B5%D1%80%D1%81_%D0%93%D0%B5%D0%B9%D0%BB%D1%81%D0%B1%D0%B5%D1%80%D0%B3) вперше розповів на сайті Microsoft про плановані розширення мови в C#3.0.[[2]](https://uk.wikipedia.org/wiki/C_Sharp#cite_note-2). У вересні 2005 року було випущено проєкт [специфікації](https://uk.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B5%D1%86%D0%B8%D1%84%D1%96%D0%BA%D0%B0%D1%86%D1%96%D1%8F) C# 3.0 і бета-версія C# 3.0, що встановлюється у вигляді доповнення до існуючих [Visual Studio](https://uk.wikipedia.org/wiki/Visual_Studio) 2005 і [.NET](https://uk.wikipedia.org/wiki/.NET) 2.0[[3]](https://uk.wikipedia.org/wiki/C_Sharp#cite_note-3). Офіційно версія C# 3.0 побачила світ [19 листопада](https://uk.wikipedia.org/wiki/19_%D0%BB%D0%B8%D1%81%D1%82%D0%BE%D0%BF%D0%B0%D0%B4%D0%B0) [2007 року](https://uk.wikipedia.org/wiki/2007) у складі [.NET Framework 3.5](https://uk.wikipedia.org/w/index.php?title=.NET_Framework_3.5&action=edit&redlink=1).

В C# 3.0 з'явилися такі радикальні доповнення та зміни:

* Ключові слова *select*, *from*, *where*, що дозволяють робити запити з [SQL](https://uk.wikipedia.org/wiki/SQL), [XML](https://uk.wikipedia.org/wiki/XML), колекції тощо (запит, інтегрований в мову — [LINQ](https://uk.wikipedia.org/wiki/LINQ), [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) *Language Integrated Query*)
* Ініціалізація [об'єкта](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)) разом з його властивостями:

Customer з = **new** Customer(); з.Name = "James";

матиме вигляд:

Customer з = **new** Customer { Name = "James" };

* Лямбда-вирази (анонімні функції):

listOfFoo.Where(**delegate**(Foo x) { **return** x.size > 10; });

матиме вигляд:

listOfFoo.Where(x => x.size > 10);

* Автоматичне визначення типів локальних змінних:

string x = "hello";

матиме вигляд:

var x = "hello";

* Безіменні типи:

var x = **new** { Name = "James" };

* Методи-розширення — додавання методу в існуючий клас за допомогою ключового слова *this* при першому параметрі статичної функції.

C# 3.0 сумісний з C# 2.0 за генерованим [MSIL](https://uk.wikipedia.org/w/index.php?title=MSIL&action=edit&redlink=1)-кодом; поліпшення в мові — чисто синтаксичні і реалізуються на етапі [компіляції](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%96%D0%BB%D1%8F%D1%82%D0%BE%D1%80). Наприклад, багато з інтегрованих запитів LINQ можна реалізувати в поточних версіях використовуючи безіменні делегати в поєднанні з предикативними методами над контейнерами, на кшталт List.FindAll і List.RemoveAll.

### Версія 4.0

Випуск четвертої версії мови програмування C# не випадково співпав з випуском нової версії .NET Framework. Він мав на меті створення інфраструктури для реалізації мов [динамічної типізації](https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%82%D0%B8%D0%BF%D1%96%D0%B7%D0%B0%D1%86%D1%96%D1%97#%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D1%96%D1%87%D0%BD%D0%B0_%D1%82%D0%B8%D0%BF%D1%96%D0%B7%D0%B0%D1%86%D1%96%D1%8F). Що суттєво поліпшило підтримку динамічних API в C# [[4]](https://uk.wikipedia.org/wiki/C_Sharp#cite_note-4), було бажаним для початку робіт над мовою [TypeScript](https://uk.wikipedia.org/wiki/TypeScript) та було використано наявними реалізаціями, наприклад [PowerShell](https://uk.wikipedia.org/wiki/PowerShell)[[5]](https://uk.wikipedia.org/wiki/C_Sharp#cite_note-5).

Нові можливості в версії 4.0:

* [Коваріантність і контраваріантність](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B2%D0%B0%D1%80%D1%96%D0%B0%D0%BD%D1%82%D0%BD%D1%96%D1%81%D1%82%D1%8C_%D1%96_%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%B0%D0%B2%D0%B0%D1%80%D1%96%D0%B0%D0%BD%D1%82%D0%BD%D1%96%D1%81%D1%82%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))
* Динамічна диспетчеризація — ключове слово **dynamic** вказує компілятору відкласти розв'язання посилань на метод до періоду виконання.

*// Так робили виклик методів COM до C# 4.0*

object o = GetObject();

Type t = o.GetType();

object result = t.InvokeMember("MyMethod", BindingFlags.InvokeMethod, **null**, o, **new** object[] { });

int i = Convert.ToInt32(result);

*// C# 4.0*

dynamic o = GetObject();

int i = o.MyMethod();

* [DLR](https://uk.wikipedia.org/w/index.php?title=Dynamic_Language_Runtime&action=edit&redlink=1)[[en]](https://en.wikipedia.org/wiki/Dynamic_Language_Runtime) — бібліотека проміжного рівня, що уніфікує взаємодію динамічних API та CLR
* Іменовані аргументи та параметри з усталеними значеннями — роблять код використання більш лаконічним. Мова [C](https://uk.wikipedia.org/wiki/C_(%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)) не підтримує перевантаження методів, тому більшість уніфікованих API має методи з величезним списком аргументів.

**class** **Logger**{

**public** **void** Log(int severity=0, string message){}

**void** Info(string info){

Log(message:info);

}

}

* Властивості-індексатори
* Можливість пропуску ref у викликах COM

Document d = **new** Document();

*// Так робили виклик методів COM до C# 4.0*

object filename = "Foo.docx";

d.SaveAs(**ref** filename, *// ...*

*// C# 4.0 створить проміжну змінну замість літерала "Foo.docx"*

*// і підставить з модифікатором ref*

d.SaveAs(FileName: "Foo.docx");

* Вбудовування типів "COM Interop" — позбавило розробників клопоту з поставкою PIA

## [Препроцесор](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D0%BE%D1%80)

C# має «препроцесорні директиви» (хоча насправді він не має [препроцесора](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D0%BE%D1%80)) на основі препроцесора C, це дає програмісту можливість визначити символи, але не [макроси](https://uk.wikipedia.org/wiki/%D0%9C%D0%B0%D0%BA%D1%80%D0%BE%D1%81). Умовні директиви, такі як #if, #endif, чи #else також можливі. Директиви типу #region дають натяк редактору для згортання фрагментів коду.

## Бібліотеки

Специфікація C# визначає мінімальний набір бібліотек типів і класів, на який має розраховувати [компілятор](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%96%D0%BB%D1%8F%D1%82%D0%BE%D1%80). На практиці, C# найчастіше використовується з якоюсь реалізацією [Common Language Infrastructure](https://uk.wikipedia.org/w/index.php?title=Common_Language_Infrastructure&action=edit&redlink=1) (CLI), яка стандартизована як ECMA-335 Common Language Infrastructure (CLI).

## Стандартизація

C# стандартизований в [ECMA](https://uk.wikipedia.org/wiki/ECMA)[[8]](https://uk.wikipedia.org/wiki/C_Sharp#cite_note-8) та [ISO](https://uk.wikipedia.org/wiki/ISO)[[9]](https://uk.wikipedia.org/wiki/C_Sharp#cite_note-9).

У серпні 2000 [Microsoft Corporation](https://uk.wikipedia.org/wiki/Microsoft_Corporation), [Hewlett-Packard](https://uk.wikipedia.org/wiki/Hewlett-Packard) та [Intel Corporation](https://uk.wikipedia.org/wiki/Intel_Corporation) виступили спонсорами стандартизації специфікації мови C#, а також [Common Language Infrastructure](https://uk.wikipedia.org/w/index.php?title=Common_Language_Infrastructure&action=edit&redlink=1) (CLI) в організації зі стандартизації [ECMA International](https://uk.wikipedia.org/wiki/ECMA_International). У грудні 2001 ECMA випустила ECMA-334 Специфікація мови C#. C# стала стандартом ISO у 2003 (ISO/IEC 23270:2006 — Information technology—Programming languages—C#). До того ECMA ще встигла адоптувати еквівалентну специфікацію як другу редакцію C# у грудні 2002.

У червні 2005 ECMA схвалила редакцію 3 специфікації C#, і відредагувала ECMA-334. Доповнення включали часткові класи, анонімні методи, тип null, і генерики (аналоги шаблонів C++).

У липні 2005 ECMA подала стандарти і відповідні технічні умови на ISO/IEC JTC 1 через пришвидшену процедуру (Fast-Track). Цей процес звичайно займає 6-9 місяців.

## Реалізації

Титульним [компілятором](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%96%D0%BB%D1%8F%D1%82%D0%BE%D1%80) C# є [Microsoft Visual C#](https://uk.wikipedia.org/w/index.php?title=Microsoft_Visual_C&action=edit&redlink=1).

Існують інші компілятори C#, часто вони включають реалізації Common Language Infrastructure і бібліотеки класів .NET:

* Проєкт Microsoft Rotor (який тепер зветься Shared Source Common Language Infrastructure, ліцензований тільки для навчального і дослідницького використання) забезпечує реалізації [CLR](https://uk.wikipedia.org/wiki/CLR) runtime і компілятор C#, і підмножину бібліотек фреймворка Common Language Infrastructure, відповідно до специфікації ECMA (до C# 2.0, і з підтримкою тільки [Windows XP](https://uk.wikipedia.org/wiki/Windows_XP)).
* Проєкт [SharpDevelop](https://uk.wikipedia.org/wiki/SharpDevelop) від компанії [icsharpcode](http://icsharpcode.net/), який використовується як альтернатива Visual Studio. Забезпечує повну реалізацію Common Language Infrastructure. Остання[[*коли?*](https://uk.wikipedia.org/wiki/%D0%92%D1%96%D0%BA%D1%96%D0%BF%D0%B5%D0%B4%D1%96%D1%8F:%D0%A1%D0%BB%D0%BE%D0%B2%D0%B0-%D0%BF%D0%B0%D1%80%D0%B0%D0%B7%D0%B8%D1%82%D0%B8#%D0%92%D0%B8%D1%80%D0%B0%D0%B7%D0%B8,_%D0%BF%D0%BE%D0%B2'%D1%8F%D0%B7%D0%B0%D0%BD%D1%96_%D0%B7_%D1%87%D0%B0%D1%81%D0%BE%D0%BC)] стабільна версія IDE 4.4 (28 серпня 2013), тестова версія 5.0 (13 лютого 2014). Зовнішній вигляд IDE дуже нагадує Microsoft Visual C#, що робить комфортним перехід від одного середовища до іншого.
* Проєкт [Mono](https://uk.wikipedia.org/wiki/Mono), початий компанією [Xamarin](https://uk.wikipedia.org/wiki/Xamarin) і продовжений її покупцем і наступником [Novell](https://uk.wikipedia.org/wiki/Novell), забезпечує відкритий компілятор C#, повну відкриту реалізацію Common Language Infrastructure, включаючи потрібні бібліотеки фреймворка відповідно до специфікації ECMA, і близьку до повної реалізацію власницьких бібліотек класів Microsoft .NET до .NET 2.0, але не специфічних бібліотек .NET 3.0 і .NET 3.5, як для Mono 2.0.
* Проєкт [DotGNU](https://uk.wikipedia.org/w/index.php?title=DotGNU&action=edit&redlink=1) також надає відкритий компілятор C#, близьку до повної реалізацію Common Language Infrastructure, включаючи потрібні бібліотеки фреймворка відповідно до специфікації ECMA, і підмножину деяких залишених власницьких біліотек класів Microsoft .NET до .NET 2.0 (які не документовані або не включені у специфікації ECMA, але включені у стандартне визначення Microsoft .NET Framework).
* [DotNetAnywhere](http://dotnetanywhere.org/) Micro Framework Common Language Runtime націлений на вбудовані системи, і підтримує майже всі специфікації C# 2.0.

## 2.2. Фреймворк для розробки додатків на Windows (Windows Forms)

**Windows Forms** — інтерфейс програмування додатків (API), відповідальний за графічний інтерфейс користувача і є частиною [Microsoft](https://uk.wikipedia.org/wiki/Microsoft) [.NET Framework](https://uk.wikipedia.org/wiki/.NET_Framework). Даний інтерфейс спрощує доступ до елементів інтерфейсу [Microsoft Windows](https://uk.wikipedia.org/wiki/Microsoft_Windows) за допомогою створення обгортки для [Win32 API](https://uk.wikipedia.org/wiki/Windows_API) в керованому коді.

Всередині .NET Framework, Windows Forms реалізується в межах простору імен System.Windows.Forms.

**.NET Framework** (читається *дот-нет*) — програмна технологія, запропонована фірмою [Microsoft](https://uk.wikipedia.org/wiki/Microsoft) як платформа для створення як звичайних програм, так і [веб-застосунків](https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%B7%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%BD%D0%BE%D0%BA). Багато в чому є продовженням ідей та принципів, покладених в технологію [Java](https://uk.wikipedia.org/wiki/Java). Однією з ідей .NET є сумісність служб, написаних різними [мовами](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). Хоча ця можливість рекламується Microsoft як перевага .NET, платформа Java має таку саму можливість.

Кожна [бібліотека](https://uk.wikipedia.org/wiki/%D0%91%D1%96%D0%B1%D0%BB%D1%96%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC) (збірка) в .NET має свідчення про свою версію, що дозволяє усунути можливі конфлікти між різними версіями збірок.

.NET — [крос-платформова](https://uk.wikipedia.org/wiki/%D0%9A%D1%80%D0%BE%D1%81-%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%BE%D0%B2%D1%96%D1%81%D1%82%D1%8C) технологія, в цей час існує реалізація для платформи [Microsoft Windows](https://uk.wikipedia.org/wiki/Microsoft_Windows), [FreeBSD](https://uk.wikipedia.org/wiki/FreeBSD) (від Microsoft) і варіант технології для [ОС](https://uk.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D0%B0_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) [Linux](https://uk.wikipedia.org/wiki/GNU/Linux) в проєкті [Mono](https://uk.wikipedia.org/wiki/Mono) (в рамках угоди між Microsoft з [Novell](https://uk.wikipedia.org/wiki/Novell)), [DotGNU](https://uk.wikipedia.org/w/index.php?title=DotGNU&action=edit&redlink=1)[[en]](https://en.wikipedia.org/wiki/DotGNU).

Захист авторських прав відноситься до створення середовищ виконання (CLR — [Common Language Runtime](https://uk.wikipedia.org/wiki/Common_Language_Runtime)) для програм .NET. Компілятори для .NET випускаються багатьма фірмами для різних мов вільно.

.NET поділяється на дві основні частини — середовище виконання (по суті [віртуальна машина](https://uk.wikipedia.org/wiki/%D0%92%D1%96%D1%80%D1%82%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0_%D0%BC%D0%B0%D1%88%D0%B8%D0%BD%D0%B0)) та інструментарій розробки.

Середовища розробки .NET-програм: [Visual Studio .NET](https://uk.wikipedia.org/wiki/Visual_Studio_.NET) (C++, C#, J#), [SharpDevelop](https://uk.wikipedia.org/wiki/SharpDevelop), [Borland](https://uk.wikipedia.org/wiki/Borland) Developer Studio (Delphi, C#) тощо. Середовище [Eclipse](https://uk.wikipedia.org/wiki/Eclipse) має додаток для розробки .NET-програм. Застосовні програми також можна розроблювати в текстовому редакторі та використовувати консольний компілятор.

Як і технологія [Java](https://uk.wikipedia.org/wiki/Java), середовище розробки .NET створює [байт-код](https://uk.wikipedia.org/wiki/%D0%91%D0%B0%D0%B9%D1%82-%D0%BA%D0%BE%D0%B4), призначений для виконання [віртуальною машиною](https://uk.wikipedia.org/wiki/%D0%92%D1%96%D1%80%D1%82%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0_%D0%BC%D0%B0%D1%88%D0%B8%D0%BD%D0%B0). Вхідна мова цієї машини в .NET називається [CIL (Common Intermediate Language](https://uk.wikipedia.org/wiki/Common_Intermediate_Language)), також відома як MSIL (Microsoft Intermediate Language), або просто IL. Застосування [байт-коду](https://uk.wikipedia.org/wiki/%D0%91%D0%B0%D0%B9%D1%82-%D0%BA%D0%BE%D0%B4) дозволяє отримати крос-платформність на рівні скомпільованого проєкту (в термінах .NET: *збірка*), а не на рівні [сирцевого тексту](https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D1%80%D1%86%D0%B5%D0%B2%D0%B8%D0%B9_%D1%82%D0%B5%D0%BA%D1%81%D1%82), як, наприклад, в [C](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_C). Перед запуском збірки в середовищі виконання (CLR) байт-код перетворюється вбудованим в середовище [JIT](https://uk.wikipedia.org/wiki/JIT-%D0%BA%D0%BE%D0%BC%D0%BF%D1%96%D0%BB%D1%8F%D1%86%D1%96%D1%8F)-компілятором (just in time, компіляція на льоту) в [машинні коди](https://uk.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D1%96_%D0%BA%D0%BE%D0%B4%D0%B8) цільового [процесора](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D1%81%D0%BE%D1%80).

Слід зазначити, що один з перших JIT-компіляторів для Java був також розроблений фірмою Microsoft (тепер починаючи з версії J2SE 1.3 створеної [8 травня](https://uk.wikipedia.org/wiki/8_%D1%82%D1%80%D0%B0%D0%B2%D0%BD%D1%8F) [2000](https://uk.wikipedia.org/wiki/2000) в Java використовується як основна, досконаліша багаторівнева компіляція — [Sun HotSpot](https://uk.wikipedia.org/wiki/Sun_HotSpot)). Сучасна технологія динамічної компіляції дозволяє досягнути аналогічного рівня швидкодії з традиційними «статичними» компіляторами (наприклад, [C++](https://uk.wikipedia.org/wiki/C%2B%2B)) і питання швидкодії часто залежить від якості того чи іншого [компілятора](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%96%D0%BB%D1%8F%D1%82%D0%BE%D1%80).

## Історія і розвиток

Як і [Abstract Window Toolkit](https://uk.wikipedia.org/wiki/Abstract_Window_Toolkit) (AWT) (схожий API для мови [Java](https://uk.wikipedia.org/wiki/Java)), бібліотека Windows Forms була розроблена як частина [.NET Framework](https://uk.wikipedia.org/wiki/.NET_Framework) для спрощення розробки компонентів графічного інтерфейсу користувача. Windows Forms побудована на основі застарілого [Windows API](https://uk.wikipedia.org/wiki/Windows_API).

Windows Forms надає можливість розробки кросплатформного графічного користувацького інтерфейсу. Але Windows Forms є лише обгорткою Windows API-компонентів.

Разом з [.NET Framework 3.0](https://uk.wikipedia.org/wiki/.NET_Framework#.NET_Framework_3.0) Microsoft випустила новий API для користувацьких інтерфейсів: [Windows Presentation Foundation](https://uk.wikipedia.org/wiki/Windows_Presentation_Foundation), який базується на мові [XAML](https://uk.wikipedia.org/wiki/XAML).

## Альтернативні реалізації

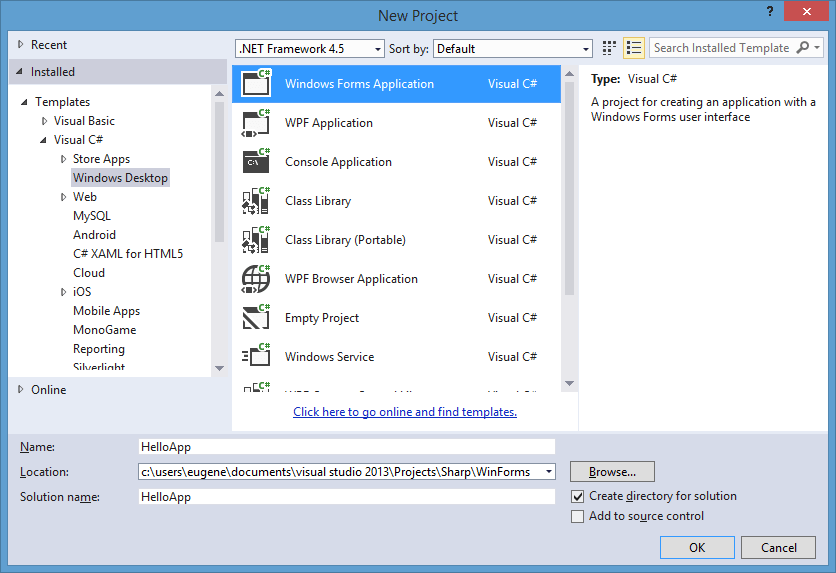
[Mono](https://uk.wikipedia.org/wiki/Mono) — проект, оплачуваний [Novell](https://uk.wikipedia.org/wiki/Novell) (раніше — [Ximian](https://uk.wikipedia.org/w/index.php?title=Ximian&action=edit&redlink=1)), одне з завдань якого — створити стандарт [Ecma](https://uk.wikipedia.org/wiki/Ecma_International), сумісний з набором інструментів [.NET](https://uk.wikipedia.org/wiki/.NET_Framework).

Для створення графічних інтерфейсів за допомогою платформи .NET застосовуються різні технології - Window Forms, WPF, додатки для магазину Windows Store (для ОС Windows 8 / 8.1 / 10). Однак найбільш простий і зручною платформою досі залишається Window Forms або форми. Даний посібник має на меті дати розуміння принципів створення графічних інтерфейсів за допомогою технології WinForms і роботи основних елементів управління.

**Створення графічного додатку**

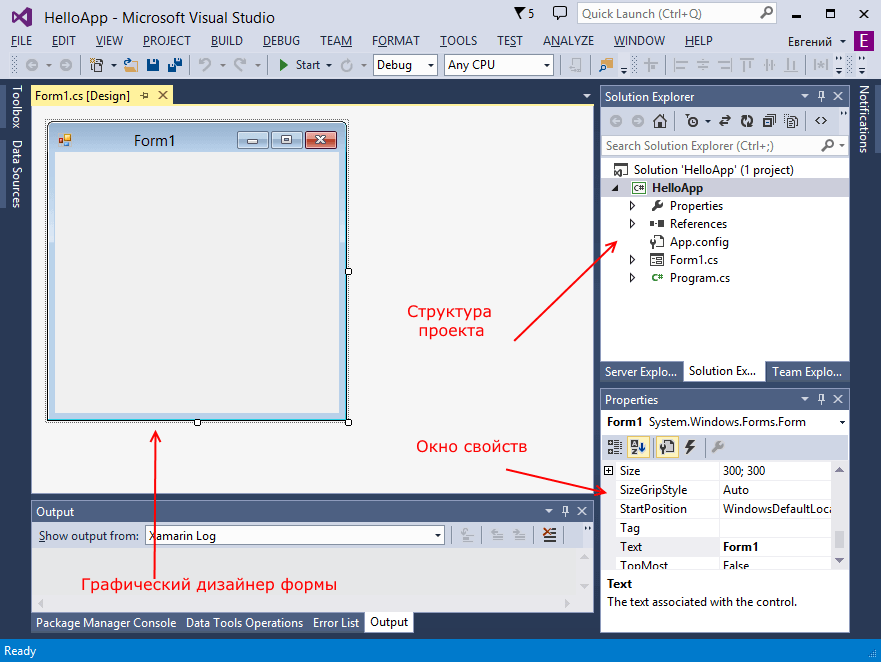
Для створення графічного проекту нам буде потрібно середовище розробки Visual Studio. Оскільки найбільш поширена версія Visual Studio 2017, то для цього посібника я буду використовувати безкоштовну версію даного середовища Visual Studio Community 2017 яку можна знайти на сторінці [https://www.visualstudio.com/en-us/products/visual-studio-community- vs.aspx](https://www.visualstudio.com/en-us/products/visual-studio-community-%20vs.aspx)

Після установки середовища та всіх її компонентів, запустимо Visual Studio і створимо проект графічного додатку. Для цього в меню виберемо пункт File (Файл) і в підміню виберемо New -> Project (Створити -> Проект). Після цього перед нами відкриється діалогове вікно створення нового проекту:



У лівій колонці виберемо Windows Desktop, а в центральній частині серед типів проектів - тип Windows Forms Application і дамо йому якесь ім'я в поле внизу. Наприклад, назвемо його HelloApp. Після цього натискаємо OK.

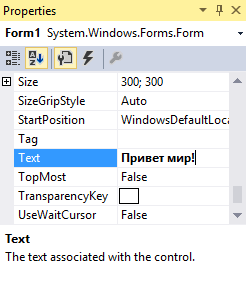
Після цього Visual Studio відкриє наш проект з створеними за замовчуванням файлами:



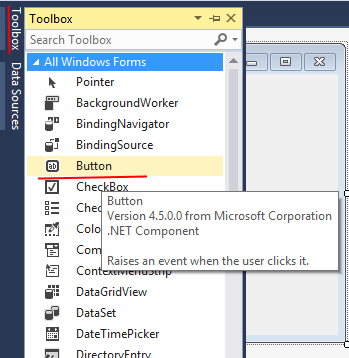
  Більшу частину простору Visual Studio займає графічний дизайнер, який містить форму майбутнього програми. Поки вона порожня і має тільки заголовок Form1. Справа знаходиться вікно файлів рішення / проекту - Solution Explorer (Оглядач рішень). Там і знаходяться всі пов'язані з нашим додатком файли, в тому числі файли форми Form1.cs.

Внизу справа знаходиться вікно властивостей - Properties. Так як у мене в даний момент обрана форма як елемент управління, то в цьому полі відображаються властивості, пов'язані з формою.

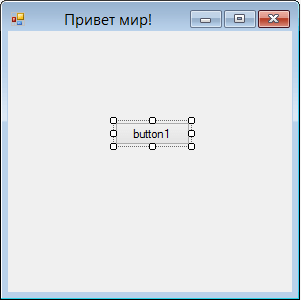
Тепер знайдемо в цьому вікні властивість форми Text і змінимо його значення на будь-яке інше:



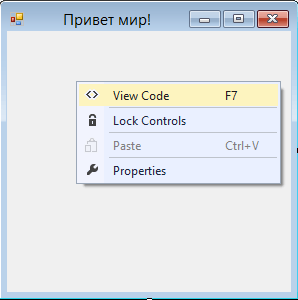
Таким чином ми поміняли заголовок форми. Тепер перенесемо на поле який-небудь елемент управління, наприклад, кнопку. Для цього знайдемо в лівій частині Visual Studio вкладку Toolbox (Панель інструментів). Натиснемо на цю вкладку, і у нас відкриється панель з елементами, звідки ми можемо за допомогою миші перенести на форму будь-який елемент:



Знайдемо серед елементів кнопку і, захопивши її покажчиком миші, перенесемо на форму:



Це візуальна частина. Тепер приступимо до самого програмування. Додамо найпростіший код на мові C #, який би виводив повідомлення після натискання кнопки. Для цього ми повинні перейти в файл коду, який пов'язаний з цією формою. Якщо у нас не відкритий файл коду, ми можемо натиснути на форму правою кнопкою миші і в меню вибрати View Code (Подивитися файл коду):



Однак скористаємося іншим способом, щоб не писати багато зайвого коду. Наведемо курсор миші на кнопку і клацнемо по ній подвійним клацанням. Ми автоматично потрапляємо в файл коду Form1.cs, який виглядає так:

|  |
| --- |
| 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 HelloApp  {      public partial class Form1 : Form      {          public Form1()          {              InitializeComponent();          }            private void button1\_Click(object sender, EventArgs e)          {            }      }  } |

Додамо вивід повідомлення після натискання кнопки, змінивши код наступним чином:

|  |
| --- |
| 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 HelloApp  {      public partial class Form1 : Form      {          public Form1()          {              InitializeComponent();          }            private void button1\_Click(object sender, EventArgs e)          {              MessageBox.Show("Привiт");          }      }  } |

**Запуск програми**

Щоб запустити додаток в режимі налагодження, натиснемо на клавішу F5 або на зелену стрілку на панелі Visual Studio. Після цього запуститься наша форма з самотньою кнопкою. І якщо ми натиснемо на кнопку на формі, то нам буде відображено повідомлення з привітанням.

Після запуску програми студія компілює його в файл з розширенням exe. Знайти даний файл можна, зайшовши в папку проекту і далі в каталог bin / Debug або bin / Release

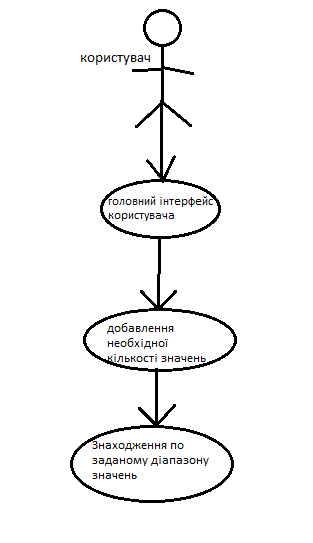
Розглянувши коротко створення проекту графічного додатку, ми можемо перейти до огляду основних компонентів і почнемо ми з форм.

# РОЗДІЛ 3. РЕЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ

## 3.1. Діаграма варіантів використання

Діаграма класів програмного забезпечення складається з 9 класів, кожен з яких виконує свою частину функціоналу, або представляє собою певний об’єкт, наприклад, вузол ігрового поля.

Діаграма класів програмного забезпечення зображена на рис. 3.1.

Рис 3.1 – Діаграма використання

## 3.3. Розробка графічного інтерфейсу користувача

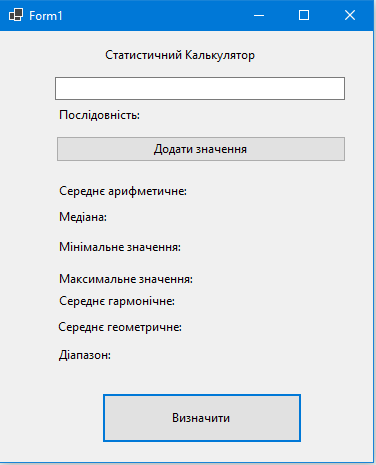


Рис. 3.2 – Інтерфейс користувача

## 3.3. Розробка методу

Основною частиною розробки Статичного Калькулятора було додавання методу, який міг перевірити, чи користувач ввів коректні дані при додаванні значень в інтерфейс програми.

Наведений нижче код показує як працює метод (коментарі до кожного кроку знаходяться в коді):

try // Перевірка чи правильні значення ввів користувач

{

var value = Convert.ToDouble(input.Text); // конвертує строку яку ввів користувач у число

Values.Add(value); // додається значення

this.textMessage.Text = "Статистичний калькулятор"; // якщо є помилка, то буде інший текст, і щоб вернути на нормальний тут вертається

this.allValues.Text = Values.Aggregate("", (current, el) => current + (el + ",")); // вивід всіх значень які є в послідовності

this.input.Text = ""; // Очищає строку де користувач ввід число

}

catch(Exception ex) // Відловлює помилку, якщо щось неправильно введене

{

this.textMessage.Text = "Input correct value!"; // Виводить цей текст, якщо користувач ввів неправильне число

this.input.Text = "";

}

}

Якщо послідовність не пуста, то калькулятор обчислює по заданим користувачем значення:

if (this.Values.Any()) // Ящко послідовність не пуста, то обчислює

{

this.Average.Text = CalculateAverage(this.Values).ToString(); // середнє арифм

this.Median.Text = CalculateMedian(this.Values).ToString(); // медіану

this.Min.Text = this.Values.Min().ToString(); // мінімальне

this.Max.Text = this.Values.Max().ToString(); // максимальне

this.HarmonicMean.Text = CalculateHarmonicMean(this.Values).ToString(); // гармонічне

this.GeometricMean.Text = CalculateGeometricMean(this.Values).ToString(); // геометричне

this.Range.Text = CalculateRange(this.Values).ToString(); // діапазон

}

else

{

this.textMessage.Text = "Input value!"; // якщо послідовність пуска, то нема з яких чисел обчилсяти, просить користувача ввести числа

}

# ВИСНОВКИ

В процесі розробки програмного забезпечення були виконані наступні завдання:

* Провести огляд предметної області
  + Визначити, що таке математика
  + Визначити поняття калькулятора
* Обрати засоби реалізації програмного забезпечення
* Розробити діаграму варіантів діяльності
* Розробити графічний інтерфейс користувача
* Розробити основні методи

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

* 1. <http://leningrad.su/calc/cforum.php?page=2&topicid=2105>
  2. <https://forum.infostart.ru/forum9/topic165489/>
  3. https://docs.microsoft.com/ru-ru/cpp/get-started/tutorial-console-cpp?view=msvc-170

4. <https://forum.itvdn.com/t/sozdanie-kalkulyatora-na-metodah-ref-out->return/3586

# ЛІСТИНГ ПРОГРАМНОГО КОДУ

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 WinFormsApp3

{

public partial class Form1 : Form

{

public List<double> Values = new List<double>();

public Form1()

{

InitializeComponent();

}

private void AddValue(object sender, EventArgs e)

{

try

{

var value = Convert.ToDouble(input.Text);

Values.Add(value);

this.textMessage.Text = "Статистичний калькулятор";

this.allValues.Text = Values.Aggregate("", (current, el) => current + (el + ","));

this.input.Text = "";

}

catch(Exception ex)

{

this.textMessage.Text = "Input correct value!"; this.input.Text = "";

}

}

private void Calculate(object sender, EventArgs e)

{

if (this.Values.Any()) {

this.Average.Text = CalculateAverage(this.Values).ToString(); this.Median.Text = CalculateMedian(this.Values).ToString(); this.Min.Text = this.Values.Min().ToString();

this.Max.Text = this.Values.Max().ToString(); this.HarmonicMean.Text = CalculateHarmonicMean(this.Values).ToString(); this.GeometricMean.Text = CalculateGeometricMean(this.Values).ToString();

this.Range.Text = CalculateRange(this.Values).ToString(); }

else

{

this.textMessage.Text = "Input value!"; }

}

private static double CalculateAverage(List<double> arr) {

var sum = arr.Sum();

var average = sum / arr.Count();

return average;

}

private static double CalculateMedian(List<double> arr) {

var index = arr.Count / 2;

return arr[index];

}

private static double CalculateRange(List<double> arr) {

var max = arr.Max();

var min = arr.Min();

return max - min;

}

private static double CalculateGeometricMean(List<double> arr)

{

var multiplication = Convert.ToDouble(arr.Aggregate((x, y) => x \* y));

var power = 1 / Convert.ToDouble(arr.Count);

var geometricMean = Math.Pow(multiplication, power);

return geometricMean;

}

private static double CalculateHarmonicMean(List<double> arr)

{

var sum = arr.Aggregate((x, y) => 1 / x + 1 / y);

var harmonicMean = arr.Count / sum;

return harmonicMean;

}

}

}