РЕФЕРАТ

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

Об'єкт розробки: методи чисельного інтегрування.

Мета кваліфікаційної роботи: створення програмного засобу навчального характеру для ознайомлення з теоретичними відомостями з розділу дисципліни «Чисельні методи в інформатиці» та реалізації методів чисельного інтегрування за допомогою обчислення визначених інтегралів .

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

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

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

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

Практичне значення визначається тим, що впровадження в навчальний процес розробленого програмного засобу надасть студенту можливість підвищити ефективність засвоєння теми «Методи чисельного інтегрування. Інтеграл та його застосування», допоможе розв’язувати задачі на належному рівні, в тому числі й прикладного змісту.

Актуальність даного програмного продукту визначається тим, що інтегральні обчислення є досить складними, громіздкими та часозатратними процесами, тому й запровадження сучасних інформаційних засобів для їх полегшення є дуже необхідним та корисним.

Список ключових слів: ЧИСЕЛЬНІ МЕТОДИ, ІНТЕГРАЛ, ПРОГРАМНИЙ ЗАСІБ, МАТЕМАТИКА, ІНФОРМАТИКА, ПРОГРАМУВАННЯ.

ABSTRACT

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

Object of development: methods of numerical integration.

The purpose of the qualification work: to create a software tool of educational nature to get acquainted with the theoretical information from the discipline "Numerical methods in computer science" and the implementation of methods of numerical integration by calculating certain integrals.

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

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

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

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

The practical significance is determined by the fact that the introduction of the developed software into the educational process will give the student the opportunity to increase the efficiency of mastering the topic "Methods of numerical integration. Integral and its application "will help to solve problems at the appropriate level, including applied content.

The relevance of this software product is determined by the fact that integrated computing is quite complex, cumbersome and time-consuming processes, so the introduction of modern information tools to facilitate them is very necessary and useful.

List of keywords: NUMERICAL METHODS, INTEGRAL, SOFTWARE, MATHEMATICS, INFORMATICS, PROGRAMMING.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ЖЦПЗ – життєвий цикл програмного забезпечення;

ОС – операційна система;

ПЗ – програмне забезпечення;

ПК – персональний комп'ютер;

ЗМІСТ

ВСТУП

Поняття інтегрального числення зустрічається з часів виникнення математики до наших часів і прямує далеко у перспективу. Визначений інтеграл присутній у нашому житті не тільки як наукове поняття, а і як сучасний термін, який зустрічається на кожному кроці у вигляді політичних гасел «євроінтеграція», «економічна інтеграція» тощо. Основні принципи інтегрального числення давно перейшли межі науки і крокують поруч з нами в житті. Можна довго перераховувати сфери застосування визначеного інтеграла, також невичерпними є його можливості у знаходженні площ і об’ємів об’єктів незвичної форми.

Дані розрахунки є досить складними, громіздкими та часозатратними, тому потрібно шукати засоби для їх полегшення.

Актуальні положення, що зазначені вище зумовили вибір теми кваліфікаційної роботи «Розробка програмного забезпечення для реалізації методів чисельного інтегрування засобами програмування C#».

Метою даної роботи є створення програмного засобу навчального характеру для ознайомлення з теоретичними відомостями з розділу дисципліни «Чисельні методи в інформатиці» та реалізації методів чисельного інтегрування за допомогою обчислення визначених інтегралів .

Завдання, які реалізуються в даній роботі :

1. Проаналізувати наукові джерела та з’ясувати основні положення визначеного інтегралу та його застосування.
2. Визначити формулу Ньютона-Лейбніца.
3. Проаналізувати метод Сімпсона.
4. На основі проаналізованого матеріалу та складеного алгоритму написати програму обчислення визначених інтегралів методом Сімпсона.
5. Провести тестування роботи програми обчислення визначених інтегралів.

Практичне значення дослідження визначається тим, що впровадження в навчальний процес розробленого програмного засобу надасть студенту:

– підвищення ефективності засвоєння теми «Методи чисельного інтегрування. Інтеграл та його застосування»;

– допомогу в формуванні вмінь розв’язувати задачі на належному рівні, в тому числі й прикладного змісту;

– організацію математичного розвитку.

Для кращого розуміння та полегшення обчислення прикладних задач з різних галузей створено програмний засіб MetodSimpsona мовою C#. Одними з важливих переваг даного засобу є те, що його не треба інсталювати (для його завантаження необхідно скористатися файлом «MetodSimpsona.exe») та використання у режими офф-лайн.

Інтерфейс програмного засобу є мінімалістичним та інтуїтивно зрозумілим. Основним його призначенням є обчислення визначених інтегралів.

Використання ІТ у навчальному процесі, комп’ютерної підтримки у викладанні шкільних предметів сприяє цікавому і повнішому, всебічному розкриттю, зрозумілішому поданню, навіть дуже складного навчального матеріалу (яким є розділ математики «Інтеграл та його застосування»), й тим самим забезпечує значне скорочення навчального часу для успішного засвоєння теми.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

1.1.1. Чисельні методи інтегрування функцій

Інтеграл - центральне поняття інтегрального числення, узагальнення поняття суми для функції, визначеній на континуумі. Виникає під час розв'язування задач про знаходження площі кривої, знаходження пройденого шляху при нерівномірному русі та інших подібних задачах.

Визначений інтеграл - в математичному аналізі це інтеграл функції з вказаною областю інтегрування. Визначений інтеграл є неперервним функціоналом, лінійним по підінтегральним функціям і адитивним по області інтегрування. У найпростішому випадку область інтегрування - це відрізок числової осі. Геометричний зміст визначеного інтеграла - це площа криволінійної фігури (криволінійної трапеції), обмеженої віссю абсцис, двома вертикалями на краях відрізка і кривою графіка функції [4].

Подальші узагальнення поняття дозволяють розширити його на кратні, поверхневі, об'ємні інтеграли, а також на інтеграли на об'єктах ширшої природи з мірою. Існує кілька різновидів визначених інтегралів: інтеграл Рімана, інтеграл Лебега, інтеграл Стілтьєса тощо.

Якщо функція $f(x)$ неперервна на відрізку $[a;b]$ (кусково-неперервна) і відома її первісна $F(x)$, то визначений інтеграл за формулою Ньютона-Лейбніца:

$$\int_a^b f(x)dx = F(b) - F(a). \tag{1.1}$$

У багатьох випадках первісна $F(x)$ невідома ніколи вона досить складна або підінтегральна функція $f(x)$ задана таблично.

Задача чисельного інтегрування функції – обчислення значення визначеного інтеграла на основі ряду значень підінтегральної функції $f(x)$.

Механічна квадратура – чисельне обчислення однократного інтеграла, а відповідні формули – квадратурні.

Механічна кубатура – чисельне обчислення подвійного інтеграла, а відповідні формули – кубатурні.

Класифікація формул чисельного інтегрування наведена на рис. 1.1.

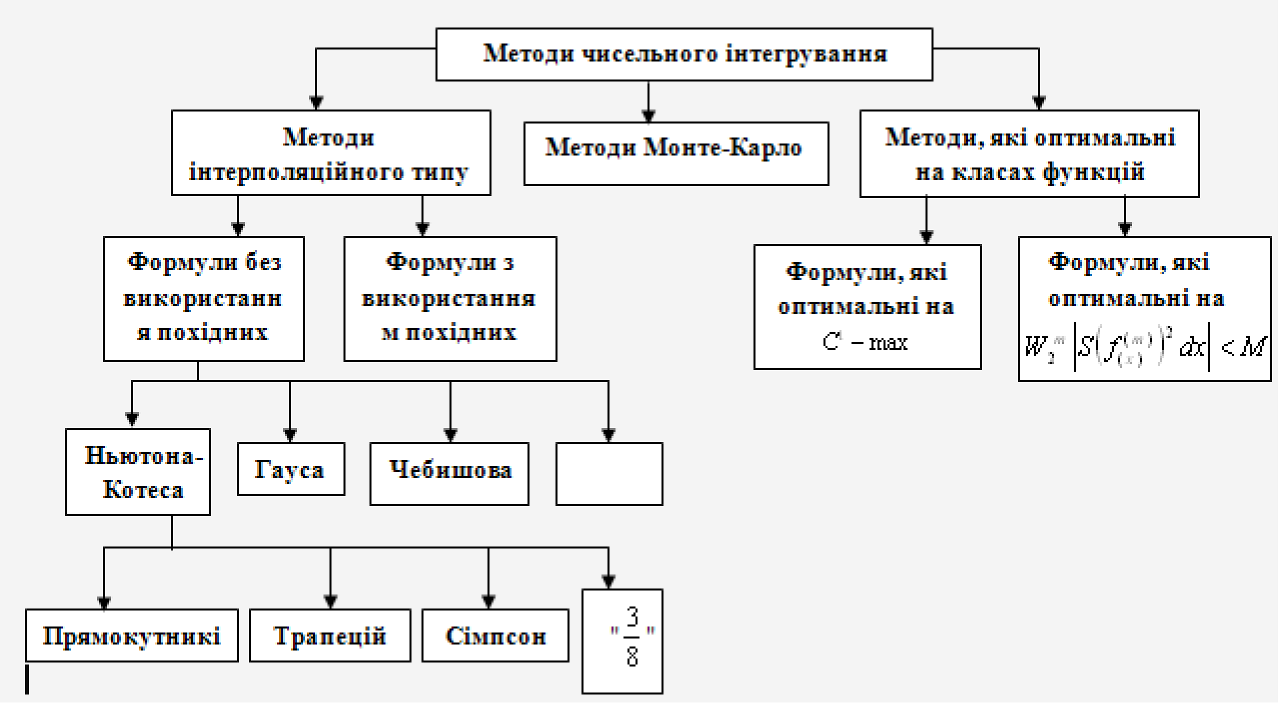


Рис. 1.1. Класифікація формул чисельного інтегрування

Квадратична формула має алгебраїчний ступінь точності *n*, якщо залишковий член квадратурної формули дорівнює 0 на множині π_n - всіх багаточленів не вище *n*-го степеня.

Методи чисельного інтегрування базуються:

– на основі інтерполяційних багаточленів;

– на основі сплайнів;
– на основі розкладу первісних у ряд Тейлора.

1.1.2. Визначений інтеграл

Визначеним інтегралом від неперервної на [a;b] функції f(x) з нижньою межею a і верхньою межею b називають різницею F(b) - F(a), де F(x) - одна з первинних для функції f(x). Позначають визначений інтеграл так f(x)dx [4].

При обчисленні різниці F(b) - F(a) можна брати будь-яку з первісних функцій f(x), що записуються в загальному вигляді F(x) + C. Але прийнято застосовувати ту первісну для якої C = 0.

Нехай функція *f*(x) задана на відрізку [a, b]. Розіб'ємо цей відрізок на n частин точками ділення $a = x_0 < x_1 < x_2 < ... < x_n = b$

У кожному проміжку $[x_{k-1}, x_k]$ довжиною $x_k = x_k - x_{k-1}$ оберемо довільну точку ξ_k і обчислимо відповідне значення функції $f(\xi_k), k = 1, 2, ..., n.$.

Побудуємо суму $\sum_{k=1}^{n} f(\xi_k)\Delta x_k$, яку називають інтегральною сумою для функції *f* (x) на відрізку [a,b].

Означення 1. Якщо існує скінченна границя інтегральної суми при $\max \Delta x_k \to 0$, незалежна від способу ділення відрізка [a,b] на частини та вибору точок ξ_k , то ця границя називається визначеним інтегралом від функції *f* (x) на відрізку [a,b] і позначається як: $\int_a^b f(x)dx$.

Математично це означення можна записати так:

$$\int_b^a f(x)dx = \lim_{\max \Delta x_k \to 0} \sum_{k=1}^{n} f(\xi_k)\Delta x_k \quad . \tag{1.2}$$

Відмітимо, що числа a та b називають нижньою та верхньою межами, відповідно.

Згідно з цим означенням рівності тепер можна записати у вигляді:

$$S = \int_b^a f(x)dx, \; S = \int_{t_0}^T f(t)dt, \qquad (1.3)$$

тобто площа криволінійної трапеції S та шлях S, пройдений точкою із змінною швидкістю V = f (t) виражаються визначеним інтегралом [5].

Теорема 1. Якщо функція f (x) неперервна на відрізку [a, b] або обмежена і має скінченну кількість точок розриву на цьому відрізку, то границя інтегральної суми існує, тобто функція f (x) інтегрована на [a, b].

Із означення визначеного інтеграла та основних теорем про границ випливають слідуючі властивості.

1. Постійний множник можна виносити за знак визначеного інтеграла, тобто якщо A - стала, то:

$$\int_a^b A \cdot f(x)dx = A \cdot \int_a^b f(x)dx. \qquad (1.4)$$

2. Визначений інтеграл від алгебраїчної суми скінченної кількості функцій дорівнює такій самій алгебраїчній сумі інтегралів від кожного доданку, тобто:

$$\int_a^b [f_1(x) \pm f_2(x) \pm \ldots \pm f_m(x)]dx = \int_a^b f_1(x)dx \pm \int_a^b f_2(x)dx \pm \ldots \pm \int_a^b f_m(x)dx. \qquad (1.5)$$

3. Якщо поміняти місцями межи інтегрування, то визначений інтеграл змінює свій знак на протилежний, тобто:

$$\int_a^b f(x)dx = -\int_a^b f(x)dx \quad . \qquad (1.6)$$

4. Визначений інтеграл з рівними межами дорівнює нулю, тобто:

$$\int_a^a f(s)dx = 0 \quad , \qquad (1.7)$$

для будь-якої функції *f* (x) [5].

5. Якщо *f* (x) ≤φ (x), x€[a, b], то $\int_a^b f(x)dx \le \int_a^b \varphi(x)dx$.

6. Якщо m та M - найбільше та найменше значення функції *f* (x) на відрізку [a,b], то:

$$m(b-a) \le \int_a^b f(x)dx \le M(b-a) \quad . \qquad (1.8)$$

7. Значення $\int_a^b f(x)dx = f(\xi)(b-a),$ де $a < \xi < b$.

8. Значення $\int_a^b f(x)dx = \int_a^c f(x)dx + \int_c^b f(x)dx;$ де $a < c < b$.

1.2. Призначення розробки та область застосування

Основна мета навчальної програми – надати студентові допомогу в опануванні теоретичних положень дисципліни щодо використання основних числових методів обчислень для розв'язання конкретних прикладних задач.

Використання ІТ у навчальному процесі, комп'ютерної підтримки у викладанні навчальних дисциплін сприяє цікавому і повнішому, всебічному розкриттю, зрозумілішому поданню, навіть дуже складного навчального матеріалу (яким є розділ математики «Інтеграл та його застосування»), й тим самим забезпечує значне скорочення навчального часу для успішного засвоєння теми.

Навчальна програма не переобтяжена зайвими елементами і водночас має зручну навігацію з можливістю вибору будь-якого завдання на довільному етапі роботи.

Використання розробленого програмного засобу сприятиме не тільки закріпленню отриманих знань, а й на конкретних прикладах виконувати обчислення інтегралів любої складності.

1.3. Підстава для розробки

Підставою для розробки кваліфікаційної роботи бакалавра на тему «Розробка програмного забезпечення для реалізації методів чисельного інтегрування засобами мови програмування C#» є наказ по Національному технічному університету «Дніпровська політехніка» від __.__. 2021р. № ____-_.

1.4. Постановка завдання

Об’єкт роботи – процес створення програмного засобу для обчислення визначеного інтегралу методом Сімпсона.

Предмет роботи – розв’язування задач за допомогою інтегрального обчислення в різних галузях.

Метою даної роботи є створення програмного засобу навчального характеру для ознайомлення з теоретичними відомостями з розділу дисципліни «Чисельні методи в інформатиці» та реалізації методів чисельного інтегрування за допомогою обчислення визначених інтегралів .

Завдання, які реалізуються в даній роботі :

1. Проаналізувати наукові джерела та з’ясувати основні положення визначеного інтегралу та його застосування.
2. Визначити формулу Ньютона-Лейбніца.
3. Проаналізувати метод Сімпсона.
4. На основі проаналізованого матеріалу та складеного алгоритму написати програму обчислення визначених інтегралів методом Сімпсона.
5. Провести тестування роботи програми обчислення визначених інтегралів.

Розроблена програма має забезпечувати наступні можливості:

- отримання теоретичних відомостей про методи чисельного інтегрування;
- можливість розрахування задач для різних даних та видів інтегралів.

Для виконання даної роботи необхідно виконати наступні завдання:

1. Ознайомитися з теоретичним матеріалом на тему «Чисельне інтегрування».
2. Дослідити та проаналізувати тему «Обчислення визначеного інтегралу методом Сімпсона».
3. Структурувати матеріал з даної теми.
4. Розробити алгоритми методів рішення задачі інтегрування.

5. Розробити структуру навчальної системи.
6. Розробити алгоритм роботи програми.
7. Визначити мову програмування і середовище розробки.
8. Розробити програму для використання на ПК із зручним інтерфейсом.

Загальні вимоги до програмного продукту:

– продуктивність;
– легкість і простота використання;
– гнучкість (можливість змінювати, додавати, розширювати);
– здібність до взаємодії (можливість інтегрувати з іншими додатками).

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Розроблений програмний засіб повинен бути багатофункціональним та забезпечувати підтримку вивчення теми з математики «Чисельне інтегрування».

Головне вікно програми повинно містити кнопки для переходу на відповідні сторінки: відображення теоретичного матеріалу (відповідно обраної теми), обчислення та графічне відображення чисельного інтегрування методом Сімпсона, вихід з програми. Також для зручного переміщення по меню програми необхідно мати на усіх вікнах кнопки «На головну» та «Вихід», натиснення на які призводить до повернення на головну сторінку або ж до виходу з програми (відповідно).

Практична частина програми має надавати можливість за вхідними даними (кількість відрізків розбиття, початок та кінець проміжку інтегрування, формула інтегралу) обчислювати значення інтегралу та виводити значення в текстовому виді та у графічному (графіки підінтегральних функцій та результати інтегрування).

1.5.2. Вимоги до інформаційної безпеки

Розроблене програмне забезпечення повинне забезпечити дотримання таких вимог до надійності:

- використовувати ліцензійне програмне забезпечення на сервері;
- здійснювати захист від вірусів на сервері;
- здійснювати захист від несанкціонованого доступу;
- застосовувати на сервері джерело безперебійного живлення для захисту від перепадів напруги або збоїв у живленні
- при псуванні обладнання робота програми повинна бути продовжена при повторному запуску.

1.5.3. Вимоги до складу та параметрів технічних засобів

Дане програмне забезпечення планується використовувати на ПК з наступними технічними характеристиками:

- процесор класу Intel ® Core 2 Duo ™ (E6600) 2.4GHz;
- SVGA монітор;
- клавіатура;
- маніпулятор «миша»;
- не менше 1 Гб ОЗУ;
- HDD 20,0 Гб.

1.5.4. Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинне бути розроблене з використанням наступних програмно-апаратних засобів:

- операційна система Microsoft Windows XP/Vista/7;
- за допомогою середовища програмування .NET C# (WPF + Windows Forms).

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

Метою даної роботи є створення програмного засобу навчального характеру для ознайомлення з теоретичними відомостями з розділу дисципліни «Чисельні методи в інформатиці» та реалізації методів чисельного інтегрування за допомогою обчислення визначених інтегралів .

Розроблена програма забезпечує наступні можливості:

- отримання теоретичних відомостей про методи чисельного інтегрування;
- можливість розрахування задач для різних даних та видів інтегралів.

Розроблений програмний засіб є багатофункціональним та забезпечує підтримку вивчення теми з математики «Чисельне інтегрування». Практична частина програми надає можливість за вхідними даними (кількість відрізків розбиття, початок та кінець проміжку інтегрування, формула інтегралу) обчислювати значення інтегралу та виводити значення в текстовому виді та у графічному (графіки підінтегральних функцій та результати інтегрування).

2.2. Опис застосованих математичних методів

2.2.1. Числове інтегрування

Нехай на відрізку $[a;b]$ задана функція $y = f(x)$ За допомогою точок x_0, x_1, …, x_n розіб'ємо відрізок $[a;b]$ на n елементарних відрізків $[x_{i-1}; x_i]$ ($i = 1,2,...n$), причому $x_0 = a$, $x_n = b$. На кожному з цих відрізків виберемо довільну точку ξ_i $(x_{i-1} \le \xi \le x_i)$ (рис. 2.1):

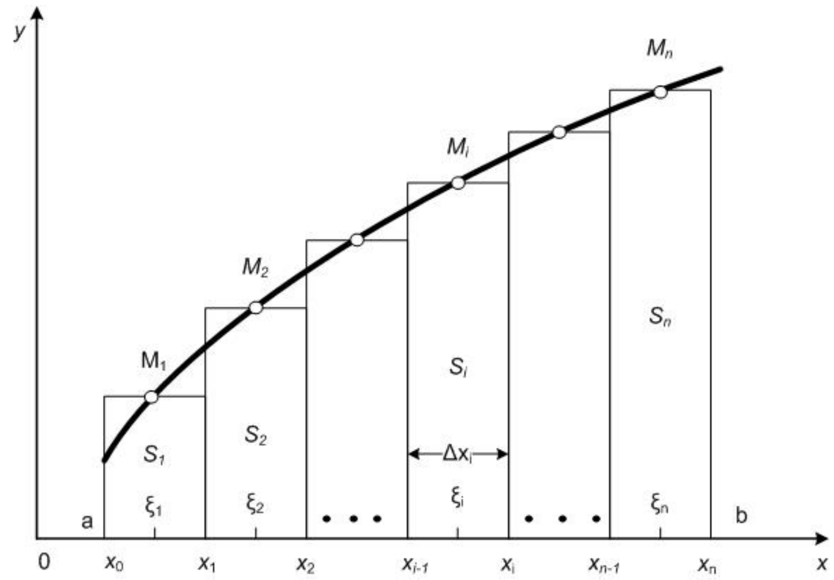


Рис. 2.1. Геометрична інтерпретація методу прямокутників

Знайдемо добутки s_i значення функції в цій точці $f\left(\xi_i\right)$ на довжину елементарного відрізка $\Delta x_i = x_i - x_{i-1}$:

$$s_i = f\left(\xi_i\right) \cdot \Delta x_i. \tag{2.1}$$

Складемо суму всіх таких добутків:

$$S_n = s_1 + s_2 + ... + s_n = \sum_{i=1}^{n} f(\xi_i) \Delta x_i. \tag{2.2}$$

Суму S_n називають інтегральною сумою.

2.2.2. Застосування інтеграла

2.2.2.1. Геометричний зміст визначеного інтеграла

Площа S криволінійної трапеції (фігура, обмежена графіком неперервної додатної на проміжку $[a; b]$ функції $f(x)$, віссю Ox, та прямими x=a, x=b) (рис. 2.2) обчислюється за формулою:

$$S = \int_a^b f(x)dx \quad . \qquad (2.3)$$

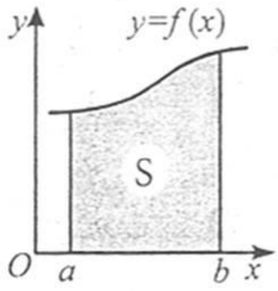


Рис. 2.2. Зображення визначення площі

2.2.2.2. Фізичний зміст визначеного інтеграла

Під час прямолінійного руху переміщення *s* (рис. 2.3.) чисельно дорівнює:

s=∫t1t2v(t)dts=∫t1t2v(t)dt, (2.4)

де $v(t)$ – швидкість руху.

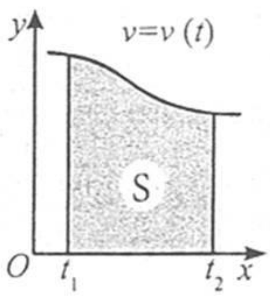


Рис. 2.3. Зображення графіку прямолінійного руху

2.2.2.3. Площа фігури

Якщо на заданому проміжку [*a*; *b*] неперервні функції *f*(*x*) і *g*(*x*) мають властивість *f*(*x*)≥*g*(*x*) для всіх x∈[a;b]x∈[a;b], (рис. 2.4) то:

S=∫ab(f(x)−g(x))dxS=∫ab(f(x)−g(x))dx. (2.5)

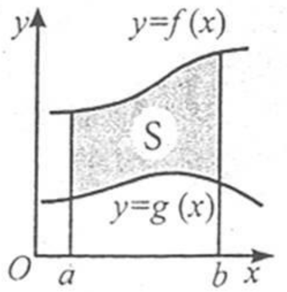


Рис. 2.4. Графік інтеграції

2.2.2.4. Обчислення площ фігур

На прикладах рис. 2.5- 2.8. наведено зображення довільних фігур та формули розрахунку їх площ.

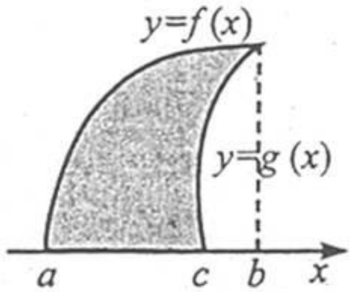


Рис. 2.5. Обчислення площі фігури відбувається за формулою
S=∫abf(x)dx−∫cbg(x)dxS=∫abf(x)dx−∫cbg(x)dx

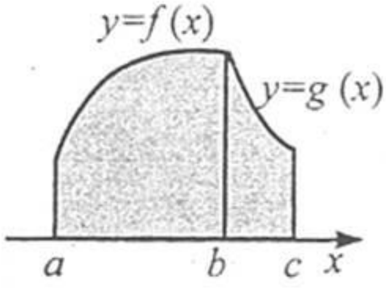


Рис. 2.6. Обчислення площі фігури відбувається за формулою

S=∫abf(x)dx+∫cbg(x)dxS=∫abf(x)dx+∫cbg(x)dx

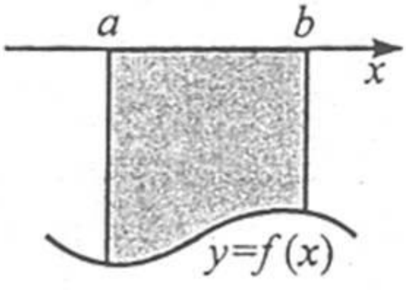


Рис. 2.7. Обчислення площі фігури відбувається за формулою

S=−∫abf(x)dxS=−∫abf(x)dx

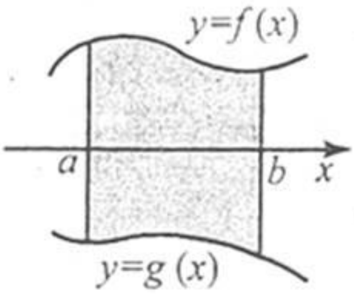


Рис. 2.8. Обчислення площі фігури відбувається за формулою

S=∫ab(f(x)−g(x))dxS=∫ab(f(x)−g(x))dx

2.2.2.5. Об’єм тіла обертання

Об’єм тіла, утвореного обертанням навколо осі O*x* криволінійної трапеції, обмеженої графіком неперервної й невід’ємної на проміжку [a; b] функції f(x) та прямими x=a і x=b (рис. 2.9), дорівнює:

$$V=\pi\int_{ab}f_{2}(x)dxV=\pi\int abf2(x)dx. \qquad (2.6)$$

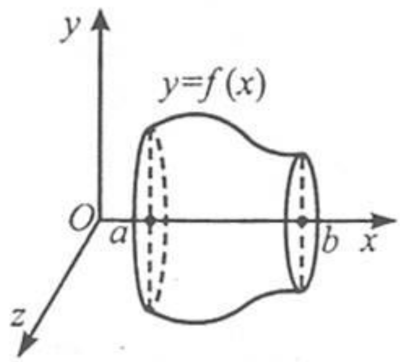
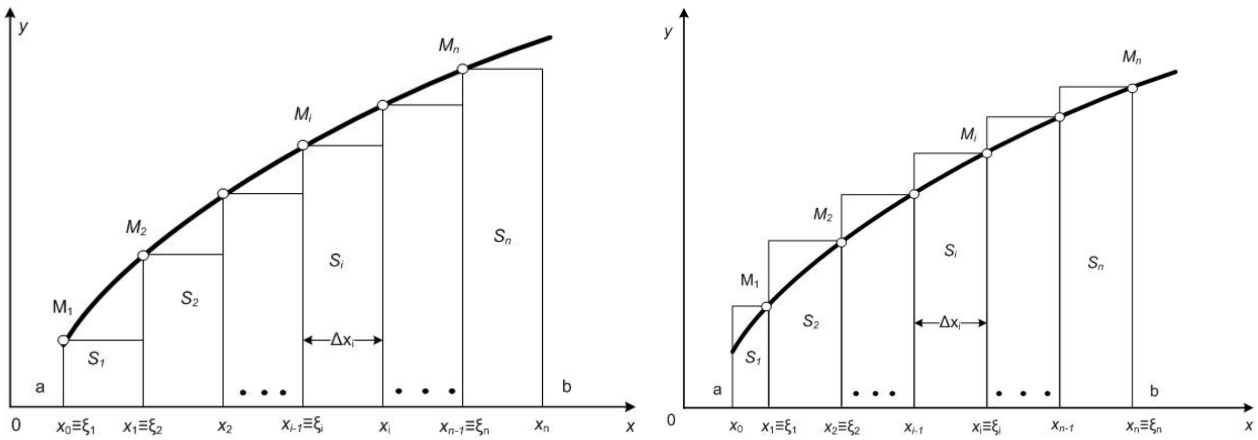


Рис. 2.9. Зображення тіла обертання

2.2.3. Методи чисельного інтегрування

2.2.3.1. Метод прямокутників

Метод прямокутників використовує заміну визначеного інтеграла інтегральною сумою. В якості точок ξ_i можуть обиратися ліві $(\xi_i = x_{i-1})$ чи праві $(\xi_i = x_i)$ межі елементарних відрізків (рис. 2.10).



а б

Рис. 2.10. Геометрична інтерпретація: а) методу лівих прямокутників; б) методу правих прямокутників

Позначивши $f(x_i) = y_i$, $\Delta x_i = h_i$, отримуємо формули методу прямокутників.

Формула лівих прямокутників:

$$\int_a^b f(x)dx = h_1 y_0 + h_2 y_1 + ... + h_n y_{n-1}. \tag{2.7}$$

Формула правих прямокутників:

$$\int_a^b f(x)dx = h_1 y_1 + h_2 y_2 + ... + h_n y_n. \tag{2.8}$$

Проте, більш точним є вид формули прямокутників, що використовує значення функції в середніх точках елементарних відрізків.

Формула середніх прямокутників:

$$\int_a^b f(x)dx = \sum_{i=0}^{n-1} h_i f\left(x_i + \frac{h_i}{2}\right). \tag{2.9}$$

Важливим окремим випадком розглянутих формул є їхнє застосування під час числового інтегрування з постійним кроком: $h_i = h = \dfrac{b-a}{n}$, $(i = 1, 2, ..., n)$. У такому випадку формули прямокутників спрощуються.

Формула лівих прямокутників:

$$\int_a^b f(x)dx = h\sum_{i=0}^{n-1} y_i \tag{2.10}$$

Формула правих прямокутників:

$$\int_a^b f(x)dx = h\sum_{i=1}^{n} y_i \ . \qquad (2.11)$$

Формула середніх прямокутників:

$$\int_a^b f(x)dx = h\sum_{i=0}^{n-1} f\left(x_i + \frac{h}{2}\right). \qquad (2.12)$$

Приклад. Обчислимо $\int_1^2 \sqrt{x dx}$, з використанням формули прямокутників, розділивши інтервал інтегрування на 10 однакових частин.

Розв'язування.

Позначимо:

$$f(x) = \sqrt{x}; \quad n = 10; \quad h = \frac{b-a}{n} = 0{,}1 \text{ , a } x_{i+1} = x_i + h.$$

i = 0,1,...9.

Тоді за формулою прямокутників:

$$\int_1^2 \sqrt{x}dx = h\sum_{i=0}^{9} f\left(\frac{x_i + x_{i+1}}{2}\right).$$

Точки поділу відрізка і значення функції в точках поділу наведені в таблиці 2.1.

Таблиця 2.1

Точки поділу відрізка і значення функції в точках поділу

i	x_i	$\frac{x_i + x_{i+1}}{2}$	$f\left(\frac{x_i + x_{i+1}}{2}\right)$ метод прямокутників	$f\left(\frac{x_i + x_{i+1}}{2}\right)$ метод трапецій
0	1			1
1	1,1	1,05	1,025	
2	1,2	1,15	1,072	1,095
3	1,3	1,25	1,118	
4	1,4	1,35	1,162	1,183
5	1,5	1,45	1,204	
6	1,6	1,55	1,245	1,265
7	1,7	1,65	1,285	
8	1,8	1,75	1,323	1,342
9	1,9	1,85	1,360	
10	2,0	1,95	1,369	1,414

Підставимо всі значення в формулу прямокутників: $\int_1^2 \sqrt{x}dx = h\sum_{i=0}^{9} f\left(\frac{x_i + x_{i+1}}{2}\right) = 1{,}219$.

Оцінимо похибку.

Оскільки $M = \max_{x\in[1,2]} |f''(x)| = \frac{1}{4}$, то

$$|\varepsilon| \le \frac{h^2(b-a)M}{24} \approx 10^{-4}.$$

Отже, $\int_1^2 \sqrt{x}dx \approx 1{,}219; \quad \varepsilon = 10^{-4}$.

Обчислимо для порівняння цей інтеграл за формулою Ньютона-Лейбніца:

$$\int_1^2 \sqrt{x}dx = \int_1^2 x^{\frac{1}{2}}dx \approx 1{,}219 .$$

Отже, інтеграл $\int_1^2 \sqrt{x}dx$, обчислений за формулою прямокутників, має всі цифри правильні.

2.2.3.2. Метод Сімпсона

Метод Сімпсона використовує квадратичну інтерполяцію. Відрізок інтегрування $[a;b]$ розбивається на парне число n рівних частин із кроком h. На кожному відрізку $[x_0;x_2]$, $[x_2;x_4]$, …, $[x_{n-2};x_n]$ підінтегральну функцію $y = f(x)$ замінюють інтерполяційним багаточленом $\varphi(x)$ другого степеня (наприклад, багаточленом Лагранжа). Геометрично – замінюють параболою, що проходить через точки $M_{i-1}(x_{i-1};y_{i-1})$, $M_i(x_i;y_i)$ та $M_{i+1}(x_{i+1};y_{i+1})$, $(i = 1, 3, 5, ..., n-1)$ (рис. 2.11).

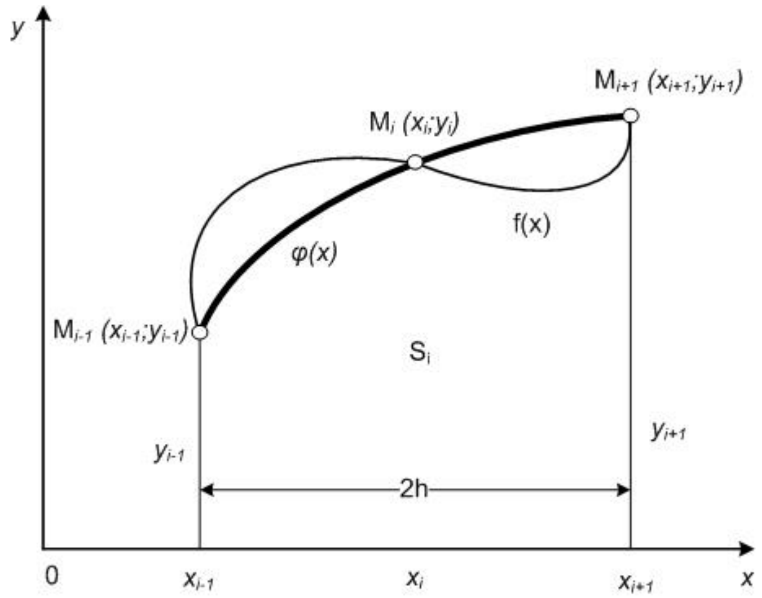


Рис. 2.11. Геометрична інтерпретація методу Сімпсона

На відрізку $[x_0; x_2]$ елементарна площа s_1 може бути обчислена за допомогою визначеного інтеграла

$$s_1 = \int_{x_0}^{x_2} \varphi_1(x)dx = \frac{h}{3}\left(y_0 + 4y_1 + y_2\right). \qquad (2.13)$$

Враховуючи, що n – парне, провівши такі ж самі обчислення для кожного елементарного відрізка $[x_{i-1}; x_{i+1}]$, $(i = 1, 3, 5, ..., n-1)$ та додавши отримані вирази матимемо:

$$\int_a^b f(x)dx = \frac{h}{3}\left[y_0 + 4\left(y_1 + y_3 + y_5 + ... + y_{n-1}\right) + 2\left(y_2 + y_4 + y_6 + ... + y_{n-2}\right) + y_n\right]. \qquad (2.14)$$

Цю формулу називають формулою Сімпсона.

Приклад. Обчислимо за формулою Сімпсона інтеграл $\int_0^1 \frac{dx}{1+x^2}$ при $n = 8$. Обчислення ведемо з шістьма знаками після коми. Оцінимо похибку результату за правилом Рунге. Порівняємо результат з точним значенням інтеграла.

Розв'язування.

Значення підінтегральної функції в окремих точках проміжку запишемо в таблицю 2.2.

Таблиця 2.2

Значення $y_i = f(x_i)$

i	x_i	$f(x_i)$ при $h = 0{,}125$	$f(x_i)$ при $2h = 0{,}25$
0	0	1	1
1	0,125	0,984625	
2	0,25	0,941176	0,941176
3	0,375	0,876712	
4	0,5	0,800000	0,800000
5	0,625	0,719101	
6	0,75	0,640000	0,640000
7	0,875	0,566389	
8	1,0	0,500000	0,500000

За формулою Сімпсона отримаємо для $h = 0{,}125$.

$$I_h = \int_0^1 \frac{dx}{1+x^2} = \frac{h}{3}\left(f(x_0) + f(x_8) + 4\left(f(x_1)\right) + f(x_3) + f(x_5) + f(x_7)\right) + 2\left(f(x_2) + f(x_4) + f(x_6)\right) = 0{,}785398.$$

Для $2h = 0{,}25$

$$I_h = \int_0^1 \frac{dx}{1+x^2} = \frac{h}{3}(f(x_0) + f(x_8) + 4(f(x_2) + f(x_6)) + 2f(x_4)) = 0{,}785398.$$

Похибка обчислень:

$$\varepsilon_h = \frac{1}{15}\left|I_{2h} - I_h\right| \approx 4 \cdot 10^{-7}.$$

Отже всі шість знаків обчисленого інтеграла повинні бути правильними. Знайдемо точне значення інтеграла:

$$I = \int_0^1 \frac{dx}{1+x^2} = \operatorname{arctg} x \Big|_0^1 = \frac{\pi}{4} \approx 0{,}785398163 .$$

,

що підтверджує отриманий за формулою Сімпсона результат.

2.2.3.3. Метод трапецій

Метод трапецій використовує лінійну інтерполяцію. Геометрично графік функції $y = f(x)$ представляється у вигляді ламаної, що з'єднує точки $M_i(x_i; y_i)$, $(i = 0, 1, 2, ..., n)$ (рис. 2.12).

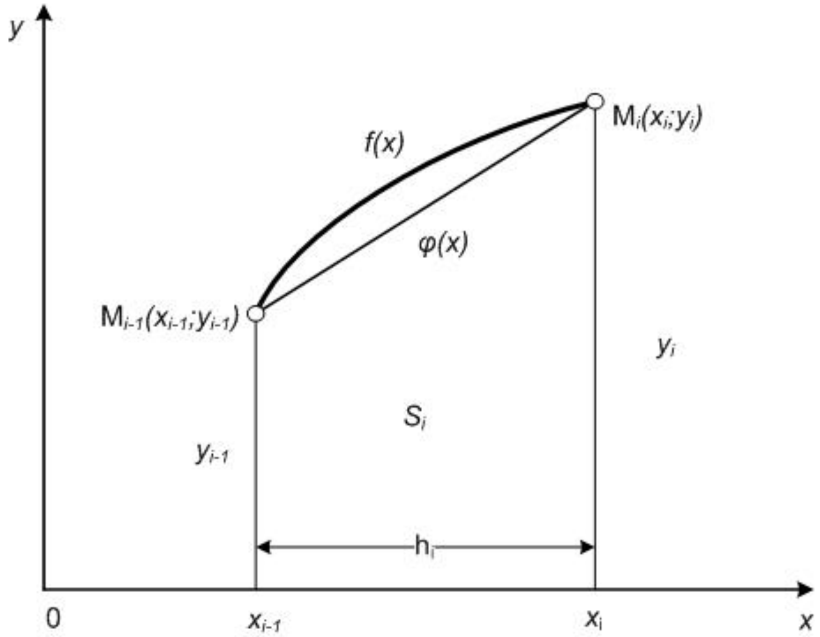


Рис. 2.12. Геометрична інтерпретація методу трапецій

В цьому випадку площа всієї фігури (криволінійної трапеції) складається з площ елементарних прямолінійних трапецій.

Площа кожної такої трапеції дорівнює добутку півсуми основ на висоту:

$$s_i = \frac{y_{i-1} + y_i}{2} h_i, \; i = 1, 2, ..., n. \tag{2.15}$$

Додавши отримані вирази, отримуємо формулу трапецій для числового інтегрування:

$$\int_a^b f(x)dx = \frac{1}{2}\sum_{i=1}^{n} h_i (y_{i-1} + y_i). \tag{2.16}$$

Важливим частковим випадком розглянутих формул є їхнє застосування для числового інтегрування з постійним кроком $h_i = h = \frac{b-a}{n}$, $(i = 1, 2, ..., n)$.

Формула трапецій в цьому випадку приймає вигляд:

$$\int_a^b f(x)dx = h\left(\frac{y_0}{2} + y_1 + y_2 + ... + y_{n-1} + \frac{y_n}{2}\right) = h\left(\frac{y_0 + y_n}{2} + \sum_{i=1}^{n-1} y_i\right). \tag{2.17}$$

Приклад. Обчислимо інтеграл $\int_1^2 x dx$, використовуючи формулу трапецій при $n = 5$; $h = 0,2$.

Розв’язування.

Відшукуємо точки поділу відрізка і значення функції в точках поділу. Отримані дані записано в табл. 2.3 їхнім урахуванням отримаємо:

$$I = \int_1^2 \sqrt{x}dx = h\left(\frac{f(a) + f(b)}{2} + \sum_{i=1}^{4} f(x_i)\right) = 1,218.$$

Оцінимо похибку обчислення:

$$M = \max_{x\in[1,2]}\left|f''(x)\right| = \frac{1}{4}\ ;\ |\varepsilon| \le \frac{h^2(b-a)M}{12} \approx 10^{-3}.$$

Отже, $I = \int_1^2 \sqrt{x}dx = 1{,}218 \pm 0{,}001.$

З використанням формули трапецій отримуємо два правильні знаки після коми, враховуючи, що точне значення інтеграла I = 1,219.

2.3. Опис використаної архітектури та шаблонів проектування

2.3.1. Опис ЖЦПЗ

Життєвий цикл програмного забезпечення – це безперервний процес, який починається з моменту прийняття рішення про необхідність створення ПЗ і закінчується в момент його повного вилучення з експлуатації.

Існує декілька підходів при визначенні фаз та робіт життєвого циклу програмного забезпечення (ЖЦПЗ), кроків процесу програмування, каскадна і спіральна моделі. Але всі вони містять загальні основні компоненти: постановка завдання, проектування рішення, реалізація, обслуговування.

В даному проекті використаний процес програмування по Райлі.

Процес програмування включає чотири кроки:

- постановка задачі, тобто отримання адекватного уявлення про те, яке завдання має виконати програма;
- проектування рішення вже поставленого завдання (загалом, таке рішення є менш формальним, ніж остаточна програма);
- кодування програми, тобто переклад спроектованого рішення в програму, яка може бути виконана на машині;
- супровід програми, тобто безперервний процес усунення в програмі неполадок і додавання нових можливостей.

Програмування починається з того моменту, коли користувач, тобто той, хто потребує програми для вирішення завдання, викладає проблему системного аналітику. Користувач і системний аналітик спільно визначають постановку задачі. Остання потім передається алгоритмісту, який відповідає за проектування рішення. Рішення (або алгоритм) представляє послідовність операцій, виконання яких призводить до вирішення завдання. Оскільки алгоритм часто не пристосований до виконання на машині, його слід перевести в машинну програму. Ця операція виконується кодувальником. За наступні зміни в програмі несе відповідальність супроводжуючий програміст. І системний аналітик, і алгоритміст, та кодер, і супроводжуючий програміст - всі вони є програмістами.

В разі великого програмного проекту число користувачів, системних аналітиків і алгоритмістів може виявитися значним. Крім того, може виникнути необхідність повернутися до попередніх кроків в силу непередбачених обставин. Все це служить додатковим аргументом на користь ретельного проектування програмного забезпечення: результати кожного кроку повинні бути повними, точними і зрозумілими.

Одним з найбільш важливих кроків програмування є постановка задачі. Вона виконує функції контракту між користувачем і програмістом (програмістами). Як і юридично погано складений контракт, погана постановка задачі марна. При гарній постановці завдання як користувач, так і програміст ясно і недвозначно представляють завдання, яку необхідно виконати, тобто в цьому випадку враховуються інтереси як користувача, так і програміста. Користувач може планувати використання ще нествореного програмного забезпечення, спираючись на знання того, що воно може. Гарна постановка завдання служить основою для формування її рішення.

Постановка завдання (специфікація програми), по суті, означає точне, повне і зрозуміле опис того, що відбувається при виконанні конкретної програми. Користувач зазвичай дивиться на комп'ютер, як на чорний ящик: для нього неважливо, як працює комп'ютер, а важливо, що може комп'ютер з того, що цікавить користувача. При цьому основна увага фокусується на взаємодії людини з машиною.

Д. Райлі пропонує для постановки задачі користуватися стандартною формою, яка забезпечує максимальну точність, повноту, ясність і включає:

— найменування завдання (схематичне визначення);

— загальний опис (короткий виклад завдання);

— введення;

— висновок;

— помилки (явно перераховані незвичайні варіанти введення, щоб показати користувачам і програмістам ті дії, які зробить машина в подібних ситуаціях);

— приклад (хороший приклад може передати сутність завдання, а також проілюструвати різні випадки).

Проектування рішення програмування є найбільш важким. На даній стадії постановка задачі має бути перетворена на алгоритм. Тому алгорітміст повинен володіти достатнім досвідом програмування і підходити до кожної нової задачі, спираючись на твердо встановлену методику проектування. Щоб уникнути помилок в програмах, алгорітмісти повинні використовувати ретельно розроблені процедури конструювання, засновані на правилах логічного висновку.

Завдання проектувальника – створення алгоритму, що виконує функції сполучної ланки між постановкою завдання і готова для виконання програмою. Перевірку створеного алгоритму, тобто наскільки останній відображає постановку завдання, здійснює системний аналітик. У силу цього і системний аналітик, і проектувальник повинні вміти читати і розуміти алгоритм. Кожен алгоритм записується на деякій псевдомова. Алгоритми, звані також псевдокод, не можуть бути виконані ні на якому комп'ютері.

Кодування алгоритму полягає в перекладі алгоритму в програму. Для створення повної, точної та зрозумілої програми необхідні відповідні методи запису програм. На відміну від природних, мови програмування створені спеціально для такого подання рішення завдання, яке може бути виконано комп'ютером.

Перш ніж завершити роботу, кодіровщік повинен переконатися, що програма відповідає псевдокод. Потім системний аналітик, алгоритміст і, що найголовніше, користувач повинні протестувати і підтвердити, що вона працює правильно. Після цього можна вважати, що програма готова для передачі користувачеві в комплекті з усією необхідною документацією.

Однак на цьому програмування не закінчується, далі слідує крок супроводу. Справа в тому, що в програмі можуть бути помилки, зумовлені або неадекватною постановкою завдання, або тим, що проект не задовольняє постановці задачі або програма не відповідає проекту. Яка б не була причина, користувач має право вимагати коригування програми, оскільки він не уявляв, що програма буде працювати таким чином. Виправлення помилок є однією з головних завдань супроводу програм. Інший не менш важливим завданням супроводу програм є її модифікація, тобто додавання в програму нових можливостей або зміна існуючих. Користувач може змінити вимоги до роботи програми, що, у свою чергу, призведе до необхідності її переписати. Складність операцій із супроводу програми залежить від типу змін, які повинні бути зроблені: у гіршому випадку може знадобитися повна переробка програми від постановки до кодування. Зазвичай на супровід програми витрачається більше часу, ніж на її створення.

Останньою складовою процесу програмування є документування. Воно включає широкий спектр описів, що полегшують процес програмування і збагачують результуючу програму. Постійне документування має становити невід'ємну частину кожного кроку програмування. Постановка завдання, проектні документи, алгоритми і програми - все це документи. Внутрішня документація, зазначена безпосередньо в програму, полегшує читання коду.

Відповідно до моделі, представленої в цьому проекті, програмування можна розділити на чотири кроки: постановку задачі, проектування рішень, кодування програми, супровід програми. Додатково модель включає документування програми як дії, які необхідно виконувати протягом всього процесу програмування.

2.3.2. Основні аспекти розробки ПЗ навчального призначення

Особливістю сучасності, коли все менше уваги приділяється «паперовим» технологіям обробки інформації і їх замінюють електронні, диктується впровадження у навчальний процес засобів автоматизованого навчання.

До переваг застосування комп'ютера у навчанні слід віднести наступні [12]:

1. Розширені можливості подання навчальної інформації.
2. Підсилення мотивації навчання.
3. Усунення однієї з причин негативного відношення до навчання – неуспіху, обумовленого нерозумінням проблеми, значними пропусками в знаннях та ін.
4. Активне залучення учнів до навчального процесу - розширюються типи навчальних завдань не стільки за постановкою, скільки за управлінням процесом їх вирішення.
5. Якісна реалізація контролю за діяльністю учнів та гнучке управлінням ним.
6. Сприяння формуванню в учнів рефлексії своєї діяльності - комп'ютер дозволяє наочно проілюструвати результат своїх дій.

Серед прикладного програмного забезпечення виділяють навчальні, тренувальні, контролюючі, імітаційні, навчально-ігрові, комбіновані програмні засоби.

Існують загальні вимоги до програмного продукту [6], такі як:

- продуктивність;
- легкість і простота використання;
- гнучкість (можливість змінювати, додавати, розширювати);
- здібність до взаємодії (можливість інтегрувати з іншими додатками).

Світовим лідером у виробництві програмного забезпечення і, зокрема, програмних засобів навчання, є корпорація Microsoft (США). В Америці, та й у всьому світі, такі продукти, як довідники, енциклопедії, навчальні програми, користуються величезним попитом. Вони є не лише надзвичайно зручними підручниками, а й також відмінним прикладом практичного застосування ІТ.

Але навіть добротні зарубіжні програми вимагають адаптації та перекладу на рідну мову. Тому відповідні дослідження здійснюються і в Україні.

2.4. Опис використаних технологій та мов програмування

У програмі використані технології платформи .Net 4.0 - 4.6 та мови програмування C#. Для розробки та дизайну користувацького інтерфейсу використана технологія WPF.

Упродовж усієї історії розвитку обчислювальної техніки еволюція мов програмування означала зміну обчислювального середовища, способу мислення програмістів і самого підходу до програмування. Мова C# – не виняток. Ця мова – черговий щабель нескінченної еволюції мов програмування. Вона є дуже зручним та ефективним засобом написання програм для сучасного середовища обчислювальної обробки даних, яке охоплює операційну систему Windows, Internet, проектування людино-машинної взаємодії тощо.

Мова C# спирається на багату спадщину в галузі програмування. Створена вона компанією Microsoft для підтримки платформи Framework. Net, що стала однією з новинок у сфері створення й управління додатками. C# є прямою спадкоємицею таких відомих мов, як C та C++ і тісно пов'язана з не менш відомою мовою Java. Отже, причиною розробки компанією Microsoft нової мови можна вважати створення компонентно-орієнтованої мови для платформи NET, що була б альтернативою мові Java. Ідеологом і розробником нової мови став Хейлсберг, який увіходив до складу провідних творців одного з найпопулярніших середовищ розробки – Delphi. Мова C# створювалася як мова об'єктно-орієнтованого програмування. Однією з головних переваг мови є її спрямованість на можливість повторного використання створених компонентів. З інших об'єктивних чинників звернемо увагу на такі:

– C# створювалася паралельно з каркасом Framework.Net і повністю враховує всі його можливості як FCL, так і CLR;

– C# є цілком об'єктно-орієнтованою мовою з можливостями спадкоємництва, де навіть типи, вбудовані в мову, представлені класами;

– C# – спадкоємиця мов C/C++, вона зберігає кращі риси цих популярних мов програмування: з ними у неї спільний синтаксис, а знайомі оператори мови полегшують перехід програмістів від C++ до C#;

– C#, зберігши основні властивості попередніх мов, стала простішою та більш надійною. Простота й надійність, головно, пов'язані з тим, що на C#, хоча й допускаються, проте не заохочуються такі небезпечні властивості C++, як покажчики, адресація, адресна арифметика;

– завдяки каркасу Framework .Net, що став надбудовою над операційною системою Windows, програмісти C# отримують ті ж переваги роботи з віртуальною машиною, що й програмісти Java. Ефективність коду навіть підвищується, оскільки середовище CLR є компілятором проміжної мови, тоді як віртуальна Java-машина є інтерпретатором байт-кода;

– потужна бібліотека каркаса підтримує зручність побудови різних типів додатків на C# і дає змогу легко будувати Web-служби, інші види компонентів.

– реалізація, що поєднує побудову надійного й ефективного коду, є важливим чинником, який сприяє успіху C#.

Головне в мові C# – це реалізація принципів об'єктно-орієнтованого програмування. Об'єктно-орієнтована методика невіддільна від C#, і всі C# програми деякою мірою мають об'єктну орієнтацію. Також варто зазначити: C# – перша мова, яка працює з XML-тегами коментарів, що може використовуватися як компілятор для створення документації безпосередньо з початкового коду.

Зважаючи на дуже зручний об'єктно-орієнтований дизайн, C# є гарним вибором для швидкого конструювання різних компонентів – від високорівневої бізнес-логіки до системних додатків, що використовують низькорівневий код. Необхідно зауважити, що C# є також і Web-орієнтованою: використовуючи прості вбудовані конструкції мови, компоненти легко перетворити на Web-сервіси, до яких можна буде звертатися з Internet за допомогою будь-якої мови на будь-якій операційній системі. Додаткові можливості й переваги перед іншими мовами надає C# використання передових Web-технологій, таких, як: XML (Extensible Markup Language) і SOAP (Simple Object Access Protocol). Середовище розробки Web-сервісів дає змогу програмісту сприймати наявні на сьогодні Web-додатки як рідні C# об'єкти. Це уможливлює співвіднесення тих Web-сервісів, що вже є, з їхніми пізнаннями в об'єктно-орієнтованому програмуванні .

У програмі використані технології платформи .Net 4.0 - 4.6 та мови програмування C#. Для розробки та дизайну користувацького інтерфейсу використана технологія WPF. Застосовані анімації, тригери на події, графічні частини, екран запуску (Splash screen). Програма використовує багатопотоковість, методи, події, класи, наслідування та інші зручності для роботи великих проектів

Ці засоби були вибрані з таких причин:

– наявність зручного простору для створення та перевірки програми;

– великі можливості, які є доступними за умов використання малої кількості системних ресурсів.

Для розробки та дизайну користувацького інтерфейсу використана технологія WPF. Застосовані анімації, тригери на події, графічні частини, екран запуску (Splash screen). Програма використовує багатопотоковість, методи, події, класи, наслідування та інші зручності для роботи великих проектів. Для виводу теоретичної частини використовуються TreeView та Browser. Для тестування застосовується файл бази даних XML з рандомізацією кожного наступного питання та відповідей у ньому. Оптимізовано пошук варіантів шляху агента до його цілі. Для цього використане розподілення прорахунку на конфігурації, розподілення розрахунків на потоки. Для того, щоб питання та відповіді в тестуванні виводилися в випадковому порядку, до типу «List<T>» був створений та доданий метод «Shuffle<T>», який можна використовувати навіть у різних потоках. Для оптимізації швидкості виконання, в коді використані LINQ-вирази, обернені умови для оператора if, лямбда-вирази, авто-властивості, наслідування конструкторів та ін.

Розробка велася в інтегрованому середовищі розробки Microsoft Visual Studio 2017 Community. Переваги даної IDE:

- безкоштовне середовище для розробки як ПЗ, так і веб-додатків на майже будь-якій мові;
- повна підтримка мови C#;
- візуальний конструктор форм та візуальне представлення властивостей елементів форми;
- гнучкі інструменти управління проектом;
- зручний редактор файлу ресурсів;
- підтримка сторонніх модулів;
- велика кількість зручних комбінацій клавіш для збільшення швидкості розробки;
- вбудований відладчик коду;
- підсвічування синтаксису та інструменти для рефакторингу коду;
- наявність IntelliSence.

2.5. Опис структури програми та алгоритмів її функціонування

2.5.1. Проектування навчальної програми

На етапі реалізації проекту був обраний найбільш відповідний інструментальний засіб розробки програмного продукту. У програмі використані технології платформи .Net 4.0 - 4.6.

Навчальна програма розроблена за допомогою мови програмування C#, яка дозволяє створювати зручний інтерфейс, наближений до системи Windows і легкий для розуміння, навіть для недосвідчених користувачів. Для розробки та дизайну користувацького інтерфейсу використана технологія WPF.

Програмна реалізація проекту «Обчислення визначених інтегралів» виконувалась у п’ять етапів:

1. Збір матеріалів з даної теми.
2. Розробка алгоритмів розрахунків.
3. Створення вікон майбутньої програми
4. Переніс інформації на підготовлені місця у вікнах
5. Доповнення програми ефектами, формування інтерфейсу.
6. Апробація програмного засобу

2.5.2. Опис алгоритмів обчислення числових інтегралів

В основі програмної реалізації практичної частини навчального додатку лежить обчислення визначених інтегралів методом Сімпсона. Для цього, спираючись на математичне обґрунтування був розроблений програмний алгоритм його реалізації.

Блок-схема програмної реалізації методу Сімпсона зображена на рис. 2.13:

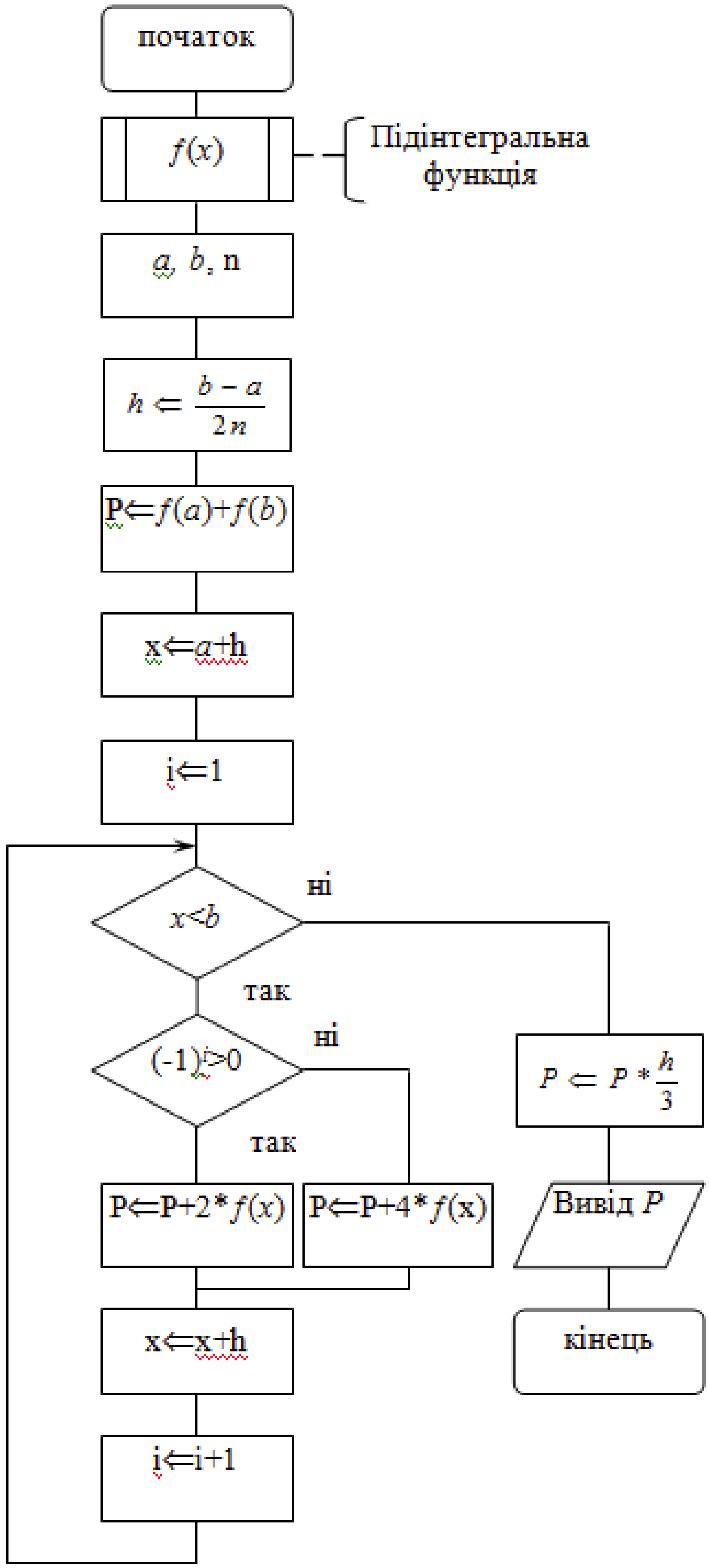


Рис. 2.13. Блок-схема реалізації методу Сімпсона

В розрахунках також використовувався метод прямокутників. Схема алгоритму обчислення інтегралів за формулою прямокутників зображена на рис. 2.14:

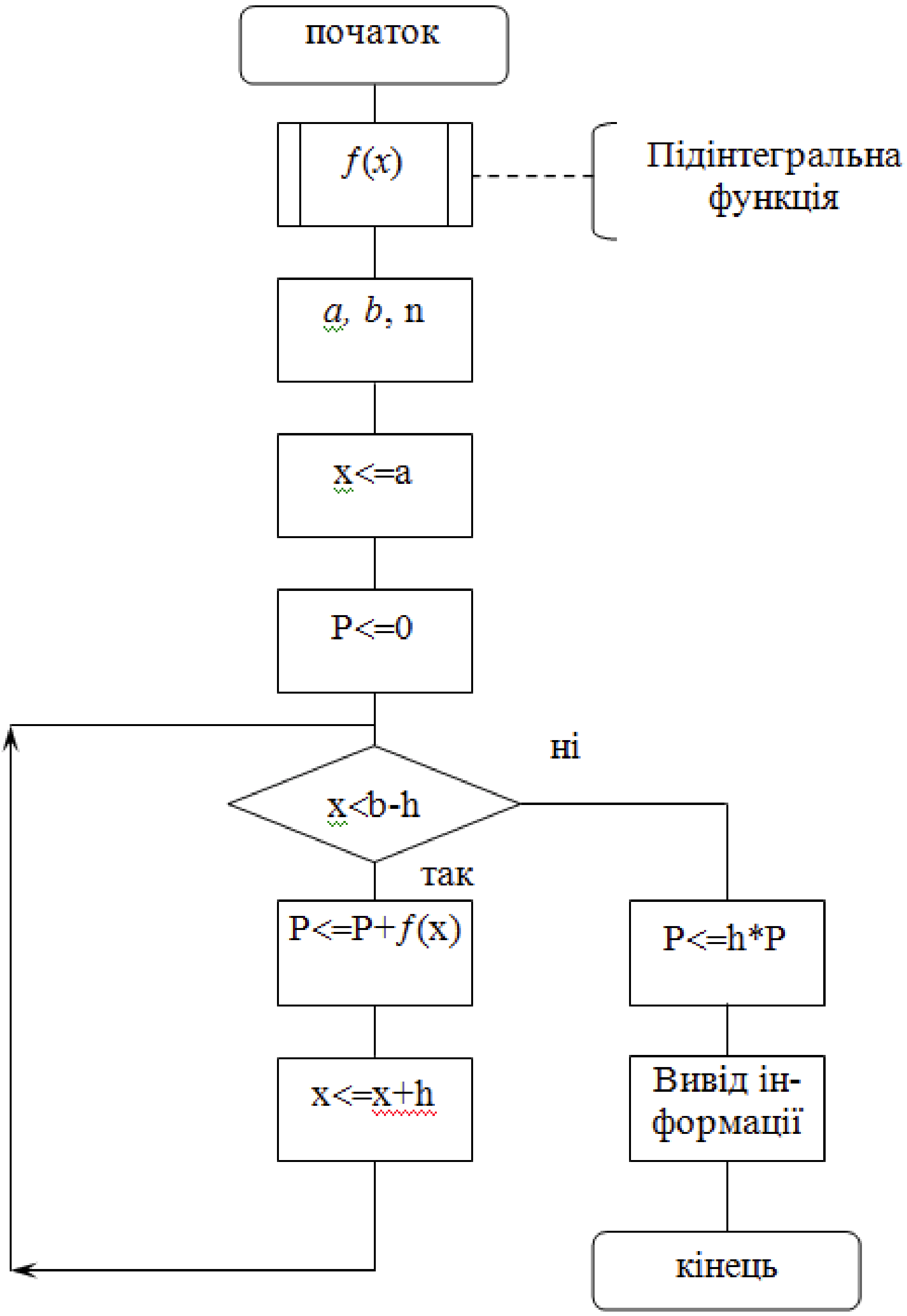


Рис. 2.14. Блок-схема реалізації методу прямокутників

2.5.3. Опис структури програми

Згідно до поставленого завдання, головне вікно програми повинно містити кнопки для переходу на відповідні сторінки: відображення теоретичного матеріалу (відповідно обраної теми), обчислення та графічне відображення чисельного інтегрування методом Сімпсона, вихід з програми. Також для зручного переміщення по меню програми необхідно мати на усіх вікнах кнопки «На головну» та «Вихід», натиснення на які призводить до повернення на головну сторінку або ж до виходу з програми (відповідно).

Практична частина програми має надавати можливість за вхідними даними (кількість відрізків розбиття, початок та кінець проміжку інтегрування, формула інтегралу) обчислювати значення інтегралу та виводити значення в текстовому виді та у графічному (графіки підінтегральних функцій та результати інтегрування).

Тому, з огляду на це, була розроблена наступна структура програми, що наведена на рис. 2.15:

Рис. 2.15. Структура інтерфейсу додатку

2.6. Обгрунтування та організація вхідних та вихідних даних програми

Програма за вхідними даними, такими як кількість відрізків розбиття, початок та кінець проміжку інтегрування, формула інтегралу надає вихідні дані, що є результатом обчислення значення інтегралу та виводить значення в текстовому виді та у графічному (графіки підінтегральних функцій та результати інтегрування).

2.7. Опис роботи розробленого програмного продукту

2.7.1. Використані технічні засоби

Дане програмне забезпечення розроблялося на ПК з наступними технічними характеристиками:

- процесор класу Intel ® Core 2 Duo ™ (E6600) 2.4GHz;
- SVGA монітор;
- клавіатура;
- маніпулятор «миша»;
- не менше 1 Гб ОЗУ;
- HDD 20,0 Гб.

2.7.2. Використані програмні засоби

У програмі використані технології платформи .Net 4.0 - 4.6 та мови програмування C#. Для розробки та дизайну користувацького інтерфейсу використана технологія WPF.

2.7.3. Виклик та завантаження програми

Перевагами даного програмного засобу є те, що його не треба інсталювати (для його завантаження необхідно скористатися файлом «MetodSimpsona.exe») та використання у режими офф-лайн.

2.7.4. Опис інтерфейсу користувача

Програмний засіб MetodSimpsona є багатофункціональним та забезпечує підтримку вивчення теми з математики «Чисельне інтегрування».

Головне вікно програми (рис. 2.16) містить 6 кнопок, по натисненні на перші п’ять кнопок відбувається відображення теоретичного матеріалу (відповідно обраної тематики): зразки наведено на рис. 2.17 - 2.25; остання кнопка завантажує форму для обчислення та графічного відображення реалізації чисельного інтегрування.

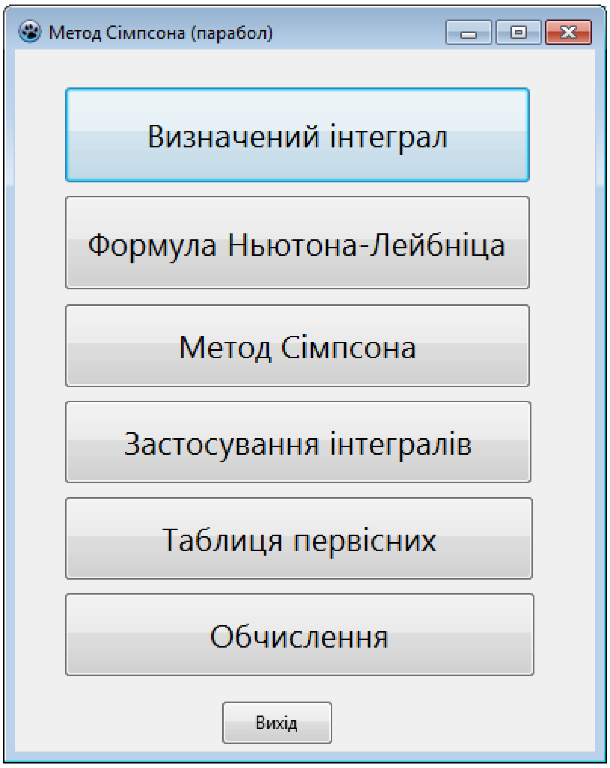
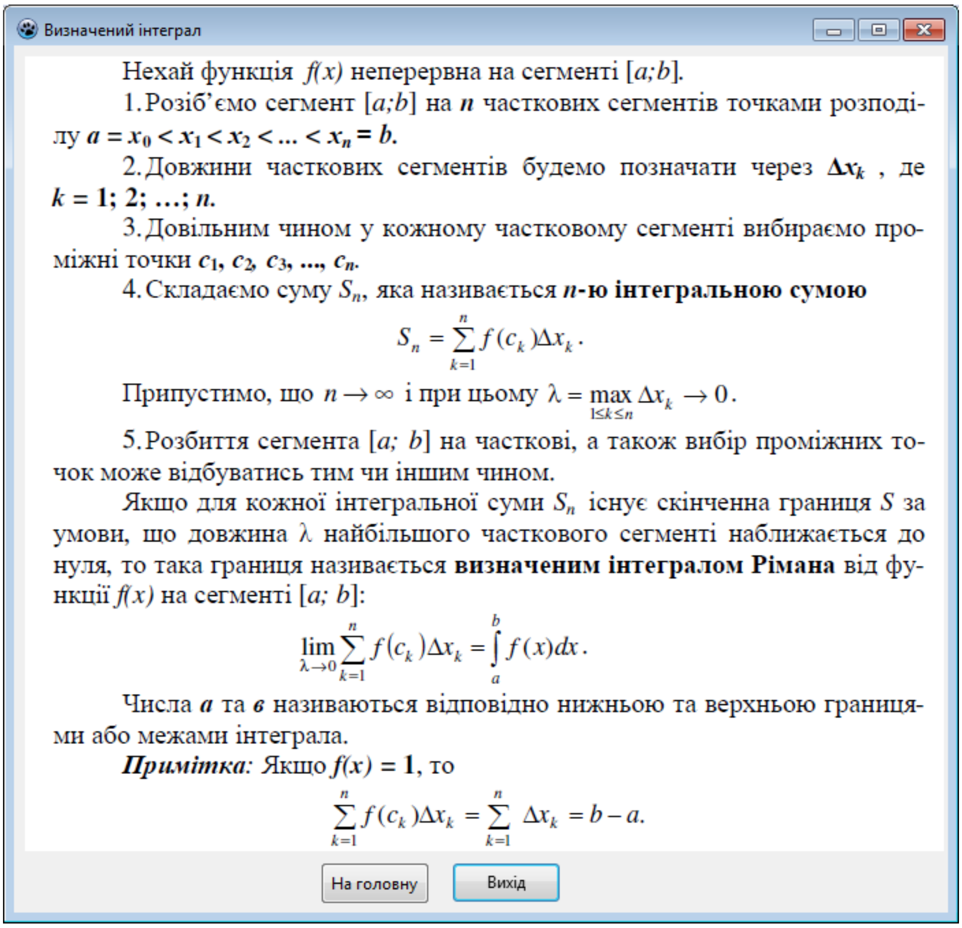


Рис. 2.16. Головне вікно програми



Нехай функція $f(x)$ неперервна на сегменті $[a;b]$.

1. Розіб'ємо сегмент $[a;b]$ на $\boldsymbol{n}$ часткових сегментів точками розподілу $\boldsymbol{a = x_0 < x_1 < x_2 < \ldots < x_n = b}$.

2. Довжини часткових сегментів будемо позначати через $\boldsymbol{\Delta x_k}$, де $\boldsymbol{k = 1; 2; \ldots; n}$.

3. Довільним чином у кожному частковому сегменті вибираємо проміжні точки $\boldsymbol{c_1, c_2, c_3, \ldots, c_n}$.

4. Складаємо суму S_n, яка називається **n-ю інтегральною сумою**

$$S_n = \sum_{k=1}^{n} f(c_k)\Delta x_k .$$

Припустимо, що $n \to \infty$ і при цьому $\lambda = \max\limits_{1\le k\le n} \Delta x_k \to 0$.

5. Розбиття сегмента $[a;\ b]$ на часткові, а також вибір проміжних точок може відбуватись тим чи іншим чином.

Якщо для кожної інтегральної суми S_n існує скінченна границя S за умови, що довжина λ найбільшого часткового сегменті наближається до нуля, то така границя називається **визначеним інтегралом Рімана** від функції $f(x)$ на сегменті $[a;\ b]$:

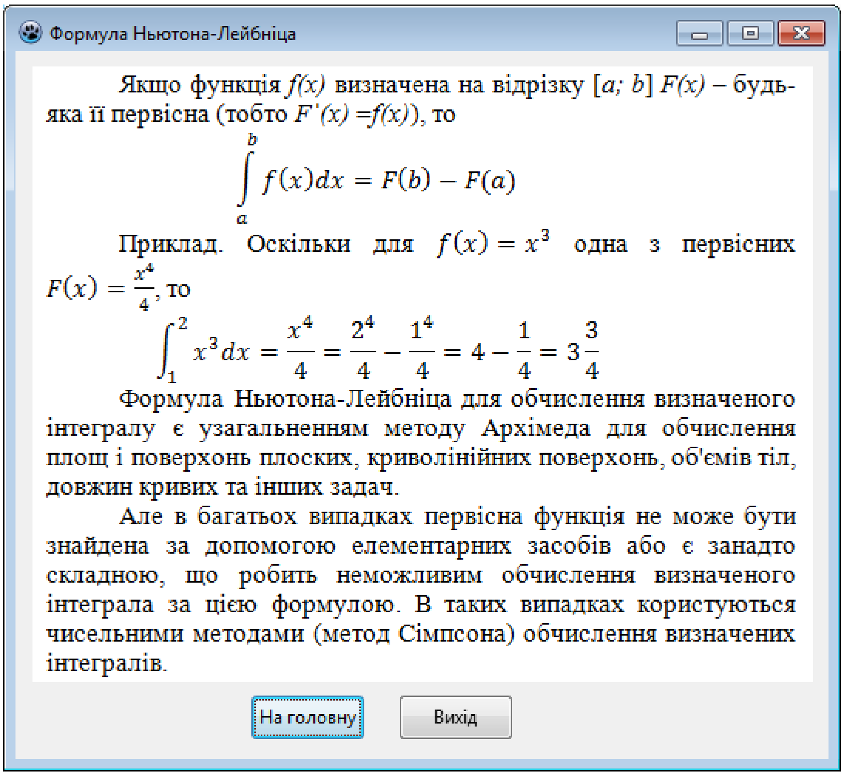
$$\lim_{\lambda\to 0}\sum_{k=1}^{n} f(c_k)\Delta x_k = \int_a^b f(x)dx .$$

Числа $\boldsymbol{a}$ та $\boldsymbol{в}$ називаються відповідно нижньою та верхньою границями або межами інтеграла.

Примітка: *Якщо* $\boldsymbol{f(x) = 1}$, то

$$\sum_{k=1}^{n} f(c_k)\Delta x_k = \sum_{k=1}^{n} \Delta x_k = b - a .$$

Рис. 2.17. Визначений інтеграл



Якщо функція $f(x)$ визначена на відрізку $[a;\ b]$ $F(x)$ – будь-яка її первісна (тобто $F'(x) = f(x)$), то

$$\int_a^b f(x)dx = F(b) - F(a)$$

Приклад. Оскільки для $f(x) = x^3$ одна з первісних $F(x) = \frac{x^4}{4}$, то

$$\int_1^2 x^3 dx = \frac{x^4}{4} = \frac{2^4}{4} - \frac{1^4}{4} = 4 - \frac{1}{4} = 3\frac{3}{4}$$

Формула Ньютона-Лейбніца для обчислення визначеного інтегралу є узагальненням методу Архімеда для обчислення площ і поверхонь плоских, криволінійних поверхонь, об'ємів тіл, довжин кривих та інших задач.

Але в багатьох випадках первісна функція не може бути знайдена за допомогою елементарних засобів або є занадто складною, що робить неможливим обчислення визначеного інтеграла за цією формулою. В таких випадках користуються чисельними методами (метод Сімпсона) обчислення визначених інтегралів.

Рис. 2.18. Формула Ньютона-Лейбніца

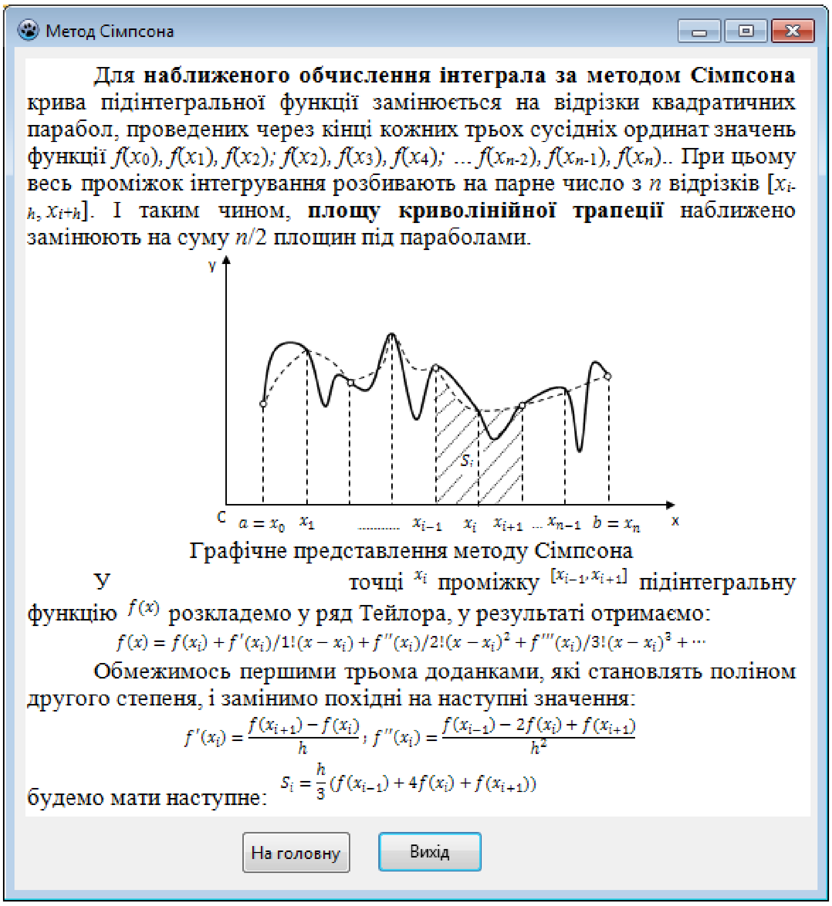


Рис. 2.19. Метод Сімпсона

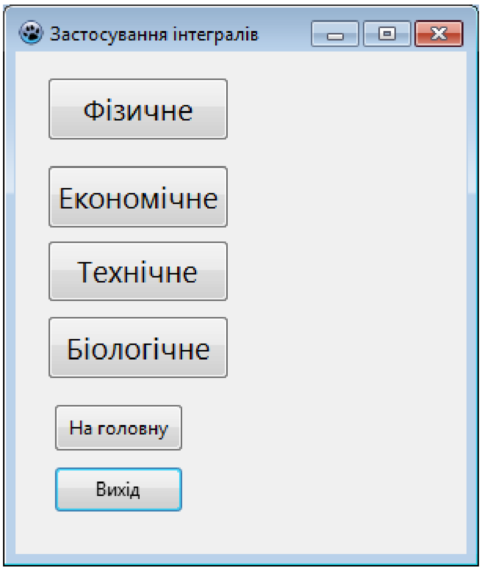
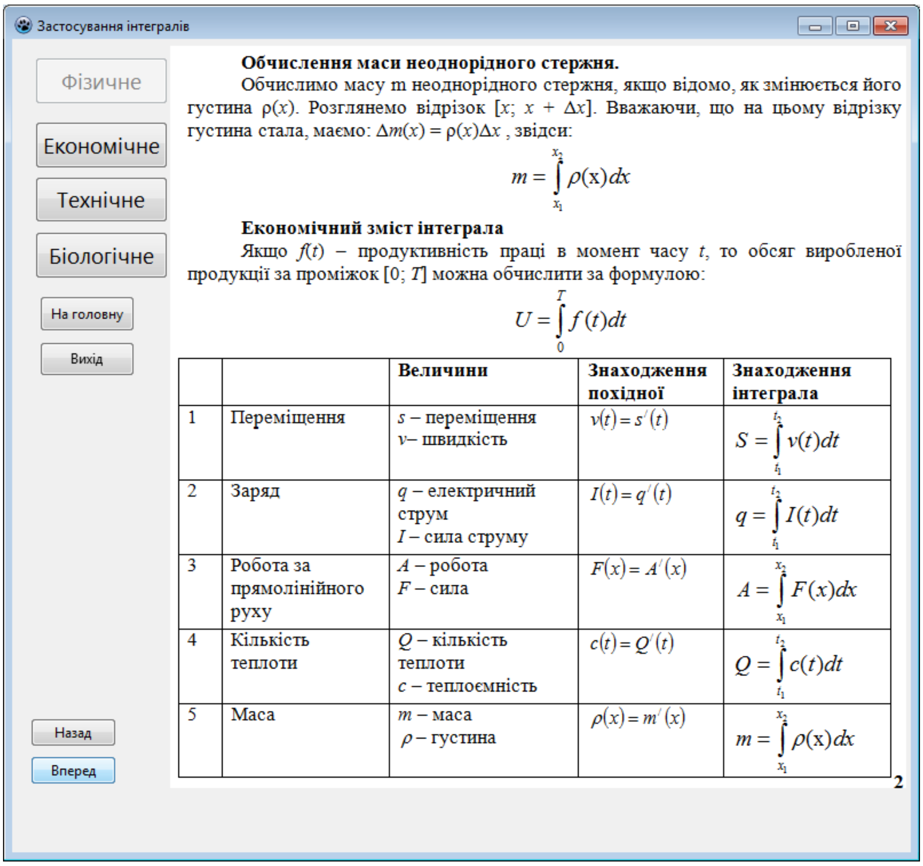


Рис. 2.20. Початковий вигляд вікна «Застосування інтегралів»



Обчислення маси неоднорідного стержня.

Обчислимо масу m неоднорідного стержня, якщо відомо, як змінюється його густина $\rho(x)$. Розглянемо відрізок $[x; x+\Delta x]$. Вважаючи, що на цьому відрізку густина стала, маємо: $\Delta m(x)=\rho(x)\Delta x$, звідси:

$$m=\int_{x_1}^{x_2}\rho(x)dx$$

Економічний зміст інтеграла

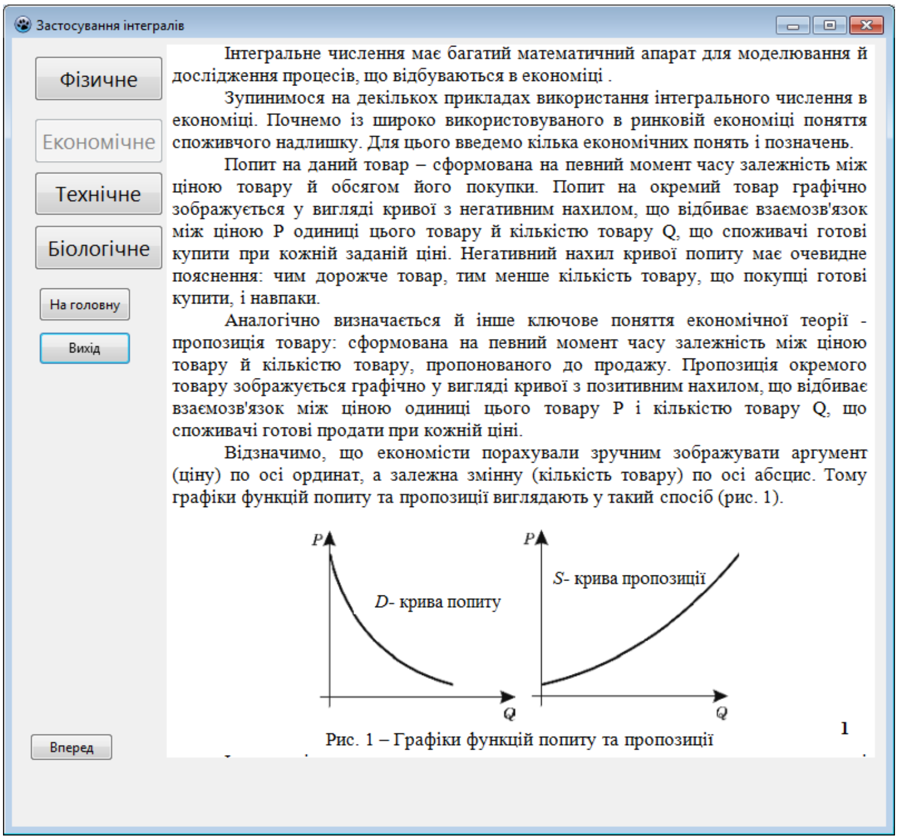
Якщо $f(t)$ – продуктивність праці в момент часу t, то обсяг виробленої продукції за проміжок $[0; T]$ можна обчислити за формулою:

$$U=\int_0^T f(t)dt$$

		Величини	**Знаходження похідної**	**Знаходження інтеграла**
1	Переміщення	s – переміщення v– швидкість	$v(t)=s'(t)$	$S=\int_{t_1}^{t_2}v(t)dt$
2	Заряд	q – електричний струм I – сила струму	$I(t)=q'(t)$	$q=\int_{t_1}^{t_2}I(t)dt$
3	Робота за прямолінійного руху	A – робота F – сила	$F(x)=A'(x)$	$A=\int_{x_1}^{x_2}F(x)dx$
4	Кількість теплоти	Q – кількість теплоти c – теплоємність	$c(t)=Q'(t)$	$Q=\int_{t_1}^{t_2}c(t)dt$
5	Маса	m – маса ρ – густина	$\rho(x)=m'(x)$	$m=\int_{x_1}^{x_2}\rho(x)dx$

2

Рис. 2.21. Фізичне застосування



Інтегральне числення має багатий математичний апарат для моделювання й дослідження процесів, що відбуваються в економіці .

Зупинимося на декількох прикладах використання інтегрального числення в економіці. Почнемо із широко використовуваного в ринковій економіці поняття споживчого надлишку. Для цього введемо кілька економічних понять і позначень.

Попит на даний товар – сформована на певний момент часу залежність між ціною товару й обсягом його покупки. Попит на окремий товар графічно зображується у вигляді кривої з негативним нахилом, що відбиває взаємозв'язок між ціною P одиниці цього товару й кількістю товару Q, що споживачі готові купити при кожній заданій ціні. Негативний нахил кривої попиту має очевидне пояснення: чим дорожче товар, тим менше кількість товару, що покупці готові купити, і навпаки.

Аналогічно визначається й інше ключове поняття економічної теорії - пропозиція товару: сформована на певний момент часу залежність між ціною товару й кількістю товару, пропонованого до продажу. Пропозиція окремого товару зображується графічно у вигляді кривої з позитивним нахилом, що відбиває взаємозв'язок між ціною одиниці цього товару P і кількістю товару Q, що споживачі готові продати при кожній ціні.

Відзначимо, що економісти порахували зручним зображувати аргумент (ціну) по осі ординат, а залежна змінну (кількість товару) по осі абсцис. Тому графіки функцій попиту та пропозиції виглядають у такий спосіб (рис. 1).

Рис. 1 – Графіки функцій попиту та пропозиції

1

Рис. 2.22. Економічне застосування

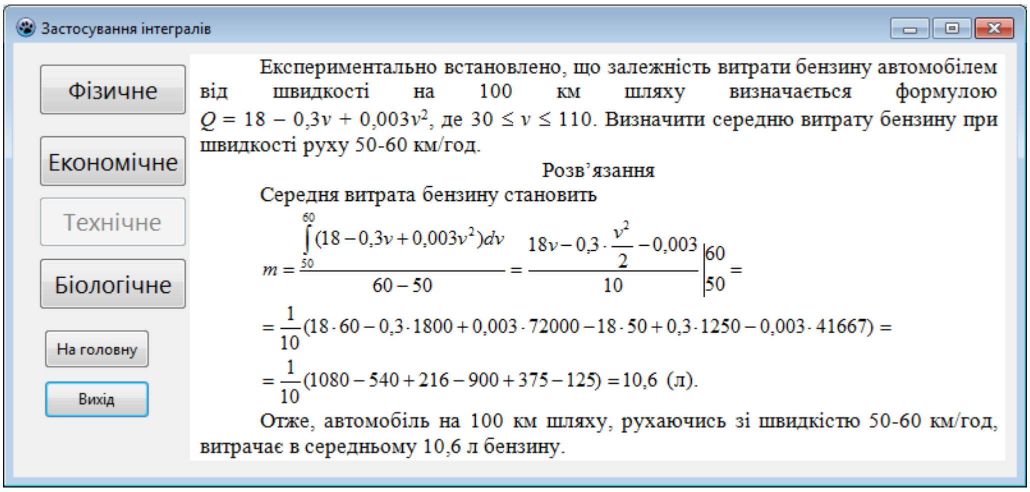


Рис. 2.23. Технічне застосування

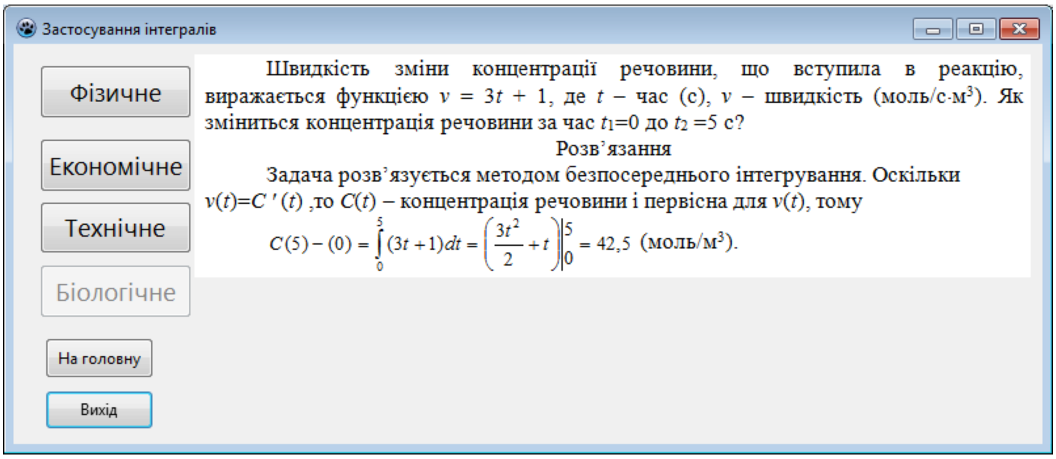
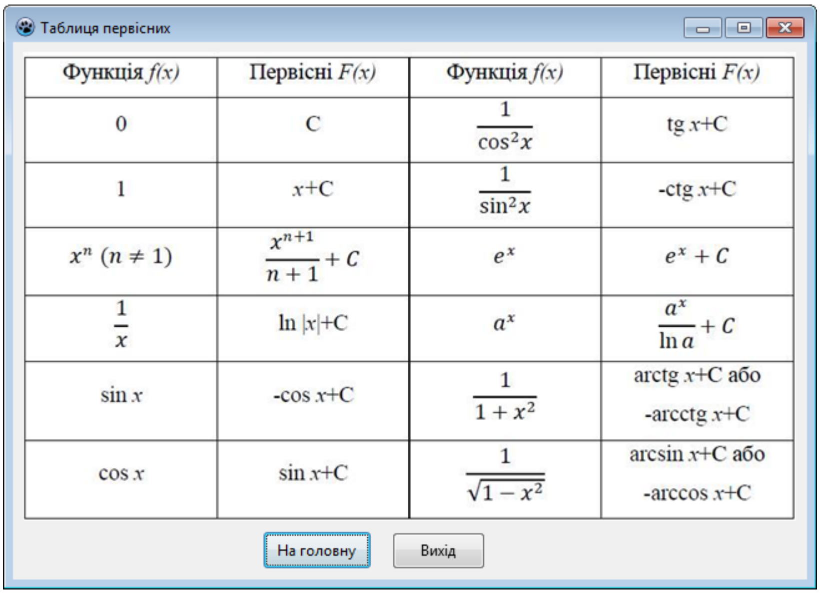


Рис. 2.24. Біологічне застосування


Таблиця первісних

Функція $f(x)$	Первісні $F(x)$	Функція $f(x)$	Первісні $F(x)$
0	C	$\frac{1}{\cos^2 x}$	tg x+C
1	x+C	$\frac{1}{\sin^2 x}$	-ctg x+C
$x^n\ (n \neq 1)$	$\frac{x^{n+1}}{n+1} + C$	e^x	$e^x + C$
$\frac{1}{x}$	ln $\lvert x\rvert$+C	a^x	$\frac{a^x}{\ln a} + C$
sin x	-cos x+C	$\frac{1}{1+x^2}$	arctg x+C або -arcctg x+C
cos x	sin x+C	$\frac{1}{\sqrt{1-x^2}}$	arcsin x+C або -arccos x+C

На головну Вихід

Рис. 2.25. Таблиця первісних

Практична частина програми надає можливість за вхідними даними (кількість відрізків розбиття, початок та кінець проміжку інтегрування, формула інтегралу) обчислювати значення інтегралу та виводити значення в текстовому виді та у графічному (графіки підінтегральних функцій та результати інтегрування).

На формі Обчислення визначеного інтеграла методом Сімпсона є 3 поля для введення числових даних (кількість відрізків, лівої і правої меж інтегрування) та поле для введення функції (визначено такі операції: '^ ^', '/', '*', '+', '-'; функції: 'Abs', 'Sqr', 'Sqrt', 'Exp', 'Ln', 'Sin', 'Cos', 'Tan', 'ArcTan').

Також на формі розміщено рисунок, для графічного зображення даного методу.

При натисненні на кнопку «Обчислити» у полі «Відповідь» з’являється результат, а при натисненні «Очистити» - усі поля для введення очищаються, окрім кількості відрізків – його значення змінюється на 2 (задля уникнення помилок).

Нижче наведено кілька зразків обчислення (рис. 2.26 - 2.29).

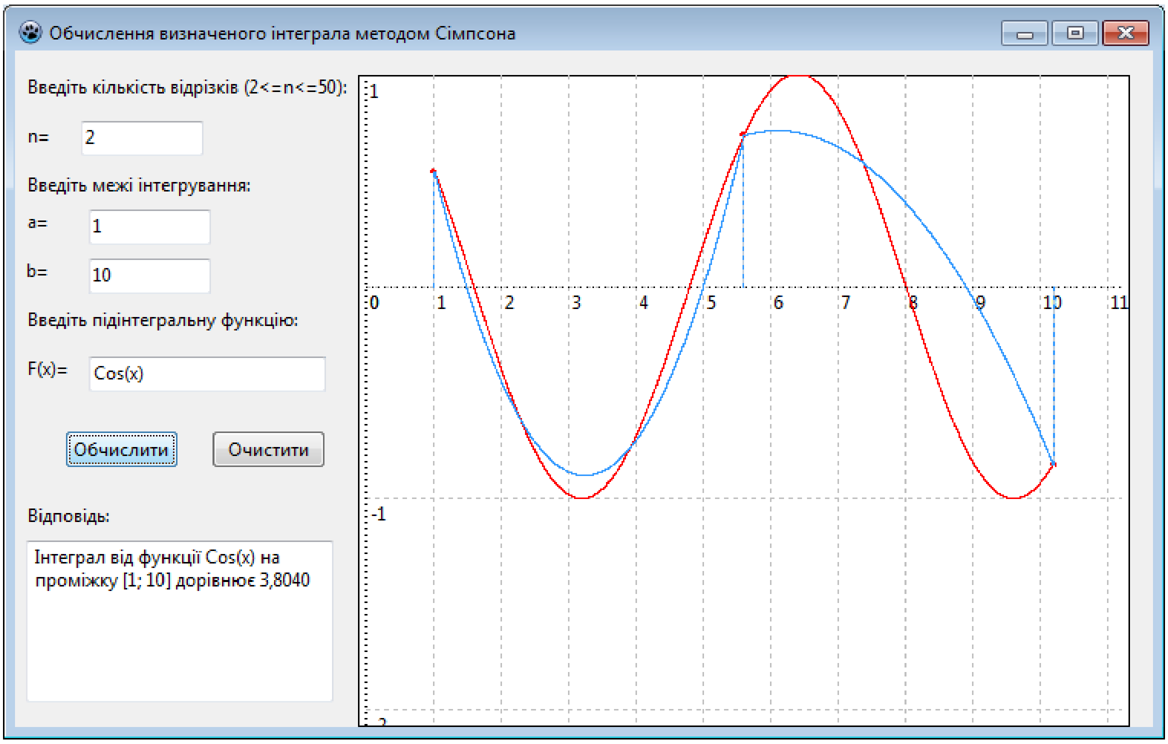


Рис. 2.26. Приклад1 реалізації програми обчислення інтегралу

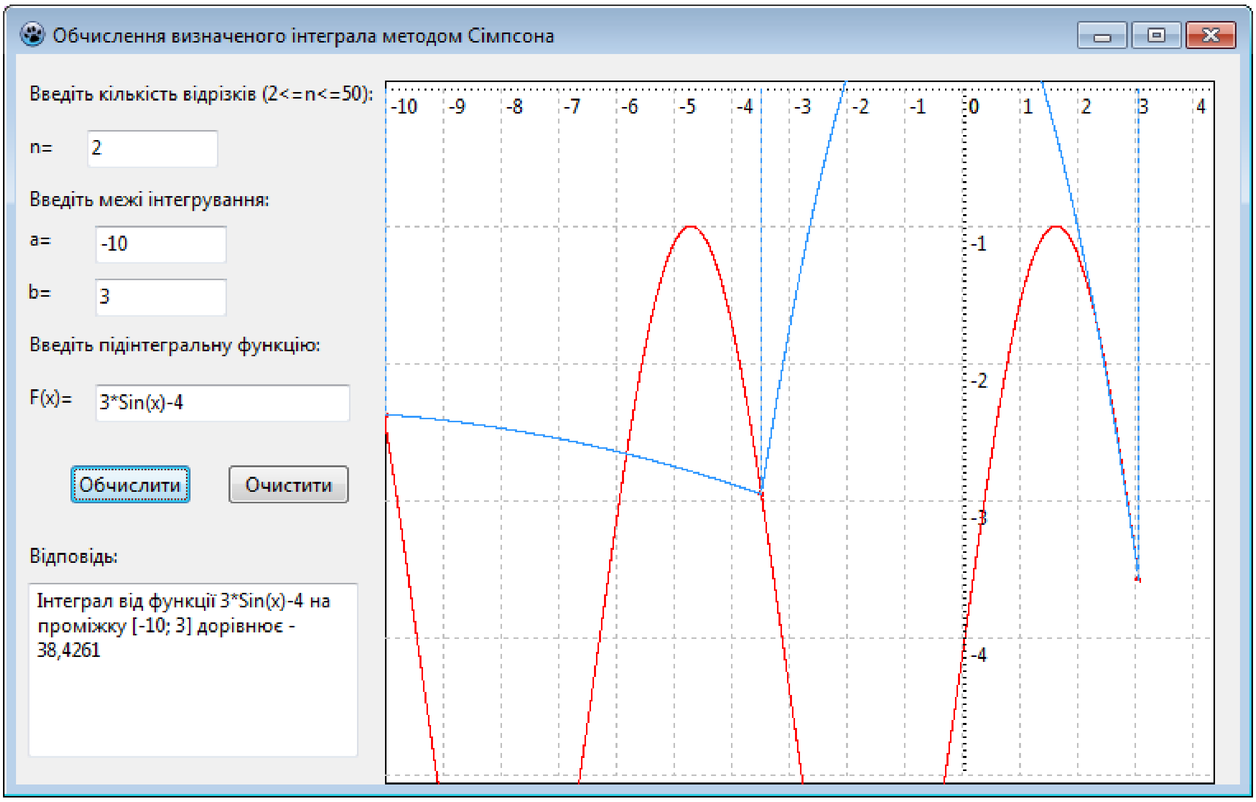


Рис. 2.27. Приклад2 реалізації програми обчислення інтегралу

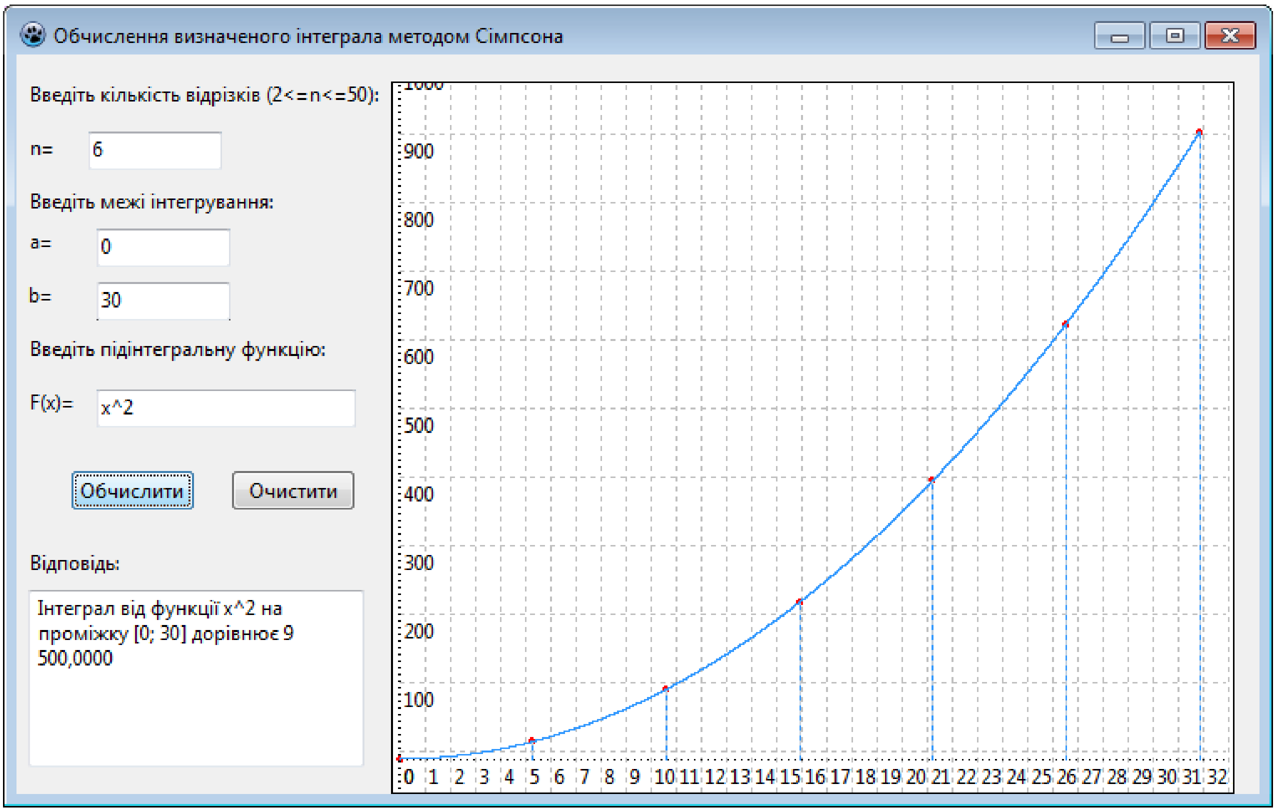


Рис. 2.28. Приклад3 реалізації програми обчислення інтегралу

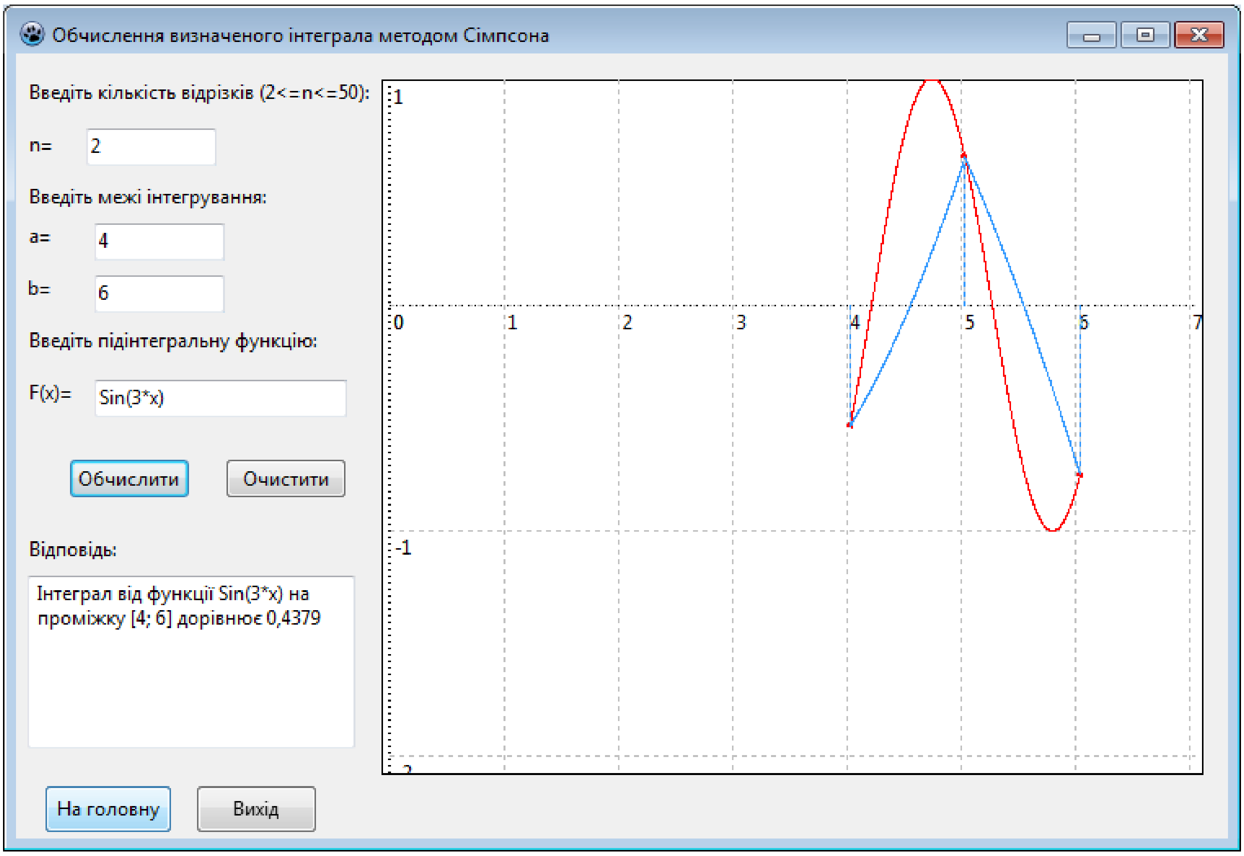


Рис. 2.29. Приклад3 реалізації програми обчислення інтегралу

Для зручного переміщення по меню ПЗ на усіх вікнах є кнопки «На головну» та «Вихід», натиснення на які призводить до повернення на головну сторінку програми або ж до виходу з програми (відповідно).

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Вхідні дані:

- передбачуване число операторів - 1500;
- коефіцієнт складності програми - 2;
- коефіцієнт корекції програми в ході її розробки - 0,08;
- годинна заробітна плата програміста, грн / год - 50.

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

Трудомісткість розробки ПЗ можна розрахувати за формулою:

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

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

tи - витрати праці на дослідження алгоритму рішення задачі,

ta - витрати праці на розробку блок-схеми алгоритму,

tп - витрати праці на програмування по готовій блок-схемі,

tотл - витрати праці на налагодження програми на ЕОМ,

tд - витрати праці на підготовку документації.

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

Умовне число операторів (підпрограм):

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

де q - передбачуване число операторів,

C - коефіцієнт складності програми,

p - коефіцієнт кореляції програми в ході її розробки.

$$Q = 1500 \cdot 2 \cdot (1 + 0{,}08) = 3240 \text{ людино-годин.}$$

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

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

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

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

Витрати праці на розробку алгоритму рішення задачі:

$$t_u = \frac{3240 \cdot 1{,}3}{80 \cdot 1{,}2} = 44\text{, людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_a = \frac{Q}{(20\ldots25)K} \quad \text{людино-годин.} \tag{3.4}$$

$$t_a = \frac{3240}{22 \cdot 1{,}2} = 123 \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20 \ldots 25)K} \quad \text{людино-годин.} \tag{3.5}$$

$$t_n = \frac{3240}{23 \cdot 1{,}2} = 117 \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

$$t_{\text{отл}} = \frac{Q}{(4 \ldots 5)K} \quad \text{людино-годин.} \tag{3.6}$$

$$t_{\text{отл}} = \frac{3240}{5 \cdot 1{,}2} = 540 \text{ людино-годин.}$$

Витрати праці на підготовку документації:

$$t_{\partial} = t_{\partial \text{р}} + t_{\partial o}, \text{ людино-годин,} \tag{3.7}$$

де tдр - трудомісткість підготовки матеріалів і рукопису.

$$t_{\partial \text{р}} = \frac{Q}{(15 \ldots 20)K}, \text{ людино-годин.} \tag{3.8}$$

$$t_{\partial p} = \frac{3240}{18 \cdot 1{,}2} = 150 \text{ людино-годин,}$$

tдо - трудомісткість редагування, печатки й оформлення документації

$$t_{\partial o} = 0{,}75 \cdot t_{\partial \text{р}}, \text{ людино-годин.} \tag{3.9}$$

$$t_{\partial o} = 150 + 73 = 223 \text{ людино-годин.}$$

Отримуємо трудомісткість розробки програмного забезпечення:

$$t = 50 + 44 + 123 + 117 + 540 + 223 = 1097$$ людино-годин.

3.2. Розрахунок витрат на створення програми

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

$$K_{ПО} = З_{ЗП} + З_{МВ}\text{, грн,} \qquad (3.10)$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot C_{ПР}\text{, грн,} \qquad (3.11)$$

де t - загальна трудомісткість, людино-годин,

Спр - середня годинна заробітна плата програміста, грн/година.

$$Ç_{ÇÏ} = 1097 \cdot 50 = 54850\ \text{ãðí.}$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} \times C_{М}\text{, грн,} \qquad (3.12)$$

де tотл - трудомісткість налагодження програми на ЕОМ, год,

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

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

$$З_{МВ} = 540 \times 5 = 2700\ \text{грн.}$$

$$\hat{E}\ddot{\imath}\hat{\imath} = 2700 + 54850 = 57550 \text{ ãðí.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p} \text{ міс.} \tag{3.13}$$

де Bk - число виконавців,

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

$$T = \frac{1097}{1 \cdot 176} = 6.2 \text{ міс.}$$

Висновки: визначено трудомісткість розробленого додатку (1097 люд-год), проведений підрахунок вартості роботи по створенню програми (57550 грн.) та розраховано час на його створення (6,2 міс).

ВИСНОВКИ

Поняття інтегрального числення зустрічається з часів виникнення математики до наших часів і прямує далеко у перспективу. Визначений інтеграл присутній у нашому житті не тільки як наукове поняття, а і як сучасний термін, який зустрічається на кожному кроці у вигляді політичних гасел «євроінтеграція», «економічна інтеграція» тощо. Основні принципи інтегрального числення давно перейшли межі науки і крокують поруч з нами в житті. Можна довго перераховувати сфери застосування визначеного інтеграла, також невичерпними є його можливості у знаходженні площ і об'ємів об'єктів незвичної форми.

Дані розрахунки є досить складними, громіздкими та часозатратними, тому потрібно шукати засоби для їх полегшення.

Для кращого розуміння та полегшення обчислення прикладних задач з різних галузей створено програмний засіб MetodSimpsona.

У програмі використані технології платформи .Net 4.0 - 4.6 та мови програмування C#. Для розробки та дизайну користувацького інтерфейсу використана технологія WPF.

Одними з важливих переваг даного засобу є те, що його не треба інсталювати (для його завантаження необхідно скористатися файлом «MetodSimpsona.exe») та використання у режими офф-лайн.

Під час написання роботи було проаналізовано наукові джерела та з'ясовані основні положення визначеного інтегралу та його застосування, розглянуті різні види чисельних методів.

Одними з важливих переваг даного засобу є те, що його не треба інсталювати (для його завантаження необхідно скористатися файлом «MetodSimpsona.exe») та використання у режими оффлайн.

Інтерфейс програмного засобу є мінімалістичним та інтуїтивно зрозумілим. Основним його призначенням є обчислення визначених інтегралів.

Використання ІТ у навчальному процесі, комп’ютерної підтримки у викладанні шкільних предметів сприяє цікавому і повнішому, всебічному розкриттю, зрозумілішому поданню, навіть дуже складного навчального матеріалу (яким є розділ математики «Інтеграл та його застосування»), й тим самим забезпечує значне скорочення навчального часу для успішного засвоєння теми.

В економічному розділі визначено трудомісткість розробленого додатку (1097 люд-год), проведений підрахунок вартості роботи по створенню програми (57550 грн.) та розраховано час на його створення (6,2 міс).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання : (ГОСТ 7.1-2003, IDT) : ДСТУ ГОСТ 7.1:2006. – Чинний з 2007–07–01. – К. : Держспоживстандарт України, 2007. – 47 с. – (Система стандартів з інформації, бібліотечної та видавничої справи) (Національний стандарт України).
2. Бусигін Б.С., Коротенко Г.М., Коротенко Л.М. Прикладна інформатика. Підручник для студентів комп'ютерних спеціальностей. – Дніпропетровськ: Видавництво НГУ, 2004. – 559 с. URL: http://www.programmer.dp.ua/book-ua-k01.php. дата звернення: 15.03.2021.
3. Бусыгин Б.С., Дивизинюк М.М., Коротенко Г.М., Коротенко Л.М. Введение в современную информатику. Учебник. – Севастополь: Издательство СНУЯЭиП, 2005. – 644 с. / URL: http://www.programmer.dp.ua/book-ru-k02.php. Дата звернення: 15.03.2021.
4. Визначений інтеграл //Вища математика в прикладах і задачах: Навчальний посібник. 2-ге видання. – К.: Центр учбової літератури / Клепко В.Ю., Голець В.Л.. - 2009. - С. 412. - 594 с.
5. Власий О.О. Особливості розробки програмного забезпечення для початкової та середньої освіти/ О.О. Власий, О.М. Дудка, Л.М. Кібірєва// науковий журнал «Компютерно-інтегровані технології: освіта, наука, виробництво» - 2015. - №19. – С. 182-188.
6. Гради Буч, Роберт А. Максимчук, Майкл У. Энгл, Бобби Дж. Янг, Джим Коналлен, Келли А. Хьюстон. Объектно-ориентированный анализ и проектирование с примерами приложений. - 3-е издание. - «Вильямс», 2010.
7. Методичні вказівки з виконання економічного розділу в дипломних проектах студентів спеціальності “Комп’ютерні системи” / Укладачі О.Г. Вагонова, Нікітіна О.Б. Н.Н. Романюк – Дніпропетровськ: Національний гірничий університет. – 2013. – 23с.

8. Методичні рекомендації до виконання кваліфікаційних робіт бакалаврів напряму підготовки 121 «Програмна інженерія» галузі знань 12 Інформаційні технології/, Л.М. Коротенко , О.С. Шевцова; Нац. гірн. ун-т. – Д : ДВНЗ НГУ, 2021. – 65 с.

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

10. Офіційний сайт середовища розробки Visual Studio Code URL: https://code.visualstudio.com/docs Дата звернення: 15.03.2021.

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

12. Формула парабол (формула Сімпсона) // Вища математика в прикладах і задачах: Навчальний посібник. 2-ге видання. – К.: Центр учбової літератури / Клепко В.Ю., Голець В.Л.. - 2009. - С. 450. - 594 с.

13. Інформатика: підруч. для 9-го кл. загальноосвіт. навч. закл. /Й.Я. Ривкінд [та ін.]. – Київ: Генеза, 2017. – 288с.: іл.

14. Дидактичні вимоги до компютерно-орієнтованих засобів навчання. URL: http://lib.iitta.gov.ua/6717/1/ Дата звернення: 15.03.2021.

15. Проблема розробки та використання програмного забезпечення навчального призначення у складі сучасних освітніх технологій URL: http://virtkafedra.ucoz.ua/el_gurnal/pages/vyp7/konf4/Cimmerman.pdf/ Дата звернення: 15.03.2021.

16. Стиллмен Э., Грин Дж. – «C# для чайников» – Л.: Питер, 2012. – 680 с.

17. Царев, В.А. Проектирование, анализ и программная реализация структур данных и алгоритмов: Учебное пособие / В.А. Царев, А.Ф. Дробанов. – Череповец., 2007. – 77 с.

ДОДАТОК А

КОД ПРОГРАМИ

Main.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using WpfApp1.Sections;
namespace WpfApp1
{
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
            Home_page home_Page = new Home_page();
            Frame_main.Navigate(home_Page);
        }
        private void Exit_Click(object sender, RoutedEventArgs e)
        {
            this.Close();
        }
```

```
private void WindowMinimize_Click(object sender, RoutedEventArgs e)
{
    this.WindowState = WindowState.Minimized;
}
private void Grid_MouseDown(object sender, MouseButtonEventArgs e)
{
    try
    {
        this.DragMove();
    }
    catch
    {
    }
}
private void bt_section_1_Click(object sender, RoutedEventArgs e)
{
    Section_1 section_1 = new Section_1();
    Frame_main.Navigate(section_1);
}
private void bt_section_2_Click(object sender, RoutedEventArgs e)
{
    Section_2 section_2 = new Section_2();
    Frame_main.Navigate(section_2);
}
private void bt_section_3_Click(object sender, RoutedEventArgs e)
{
    Section_3_theory section_3 = new Section_3_theory();
    Frame_main.Navigate(section_3);
}
private void Menu_Click(object sender, RoutedEventArgs e)
{
    if (Section_1.mark >= 5)
    {
        bt_section_2.IsEnabled = true;
        if (Section_2.mark >= 5)
```

```
                {
                    bt_section_3.IsEnabled = true;
                }
                else
                {
                    bt_section_3.IsEnabled = false;
                }
            }
            else
            {
                bt_section_2.IsEnabled = false;
            }
        }
        private void bt_introduction_Click(object sender, RoutedEventArgs e)
        {
            Introduction introduction = new Introduction();
            Frame_main.Navigate(introduction);
        }
        private void bt_cval_Click(object sender, RoutedEventArgs e)
        {
            Pracktice pracktice = new Pracktice();
            Frame_main.Navigate(pracktice);
        }
    }
}
            par.Clear();
            listb_analizet.Items.Clear();
            int i_bonded = 0, i_notrealistic = 0, i_angles = 0, i_dihedralas = 0;
            for (int i = 0; i < param.Length; i++)
            {
                if (param[i].StartsWith("#bonding parameters") == true)
                {
                    i_bonded = i;
                }
                else if (param[i].StartsWith("#not realistic bonds") == true)
```

```
    {
        i_notrealistic = i;
    }
    else if (param[i].StartsWith("#angles") == true)
    {
        i_angles = i;
    }
    else if (param[i].StartsWith("#dihedrals") == true)
    {
        i_dihedralas = i;
    }
}
for (int i = i_dihedralas + 1; i < param.Length; i++)
{
    for (int j = 0; j < 10; j++)
    {
        dihedrals_param[i - i_dihedralas - 1, j] = param[i].Split(tab)[j];
    }
}
for (int i = i_angles + 1; i < i_dihedralas; i++)
{
    for (int j = 0; j < 5; j++)
    {
        angles_param[i - i_angles - 1, j] = param[i].Split(tab)[j];
    }
}
for (int i = i_notrealistic + 1; i < i_angles; i++)
{
    for (int j = 0; j < 4; j++)
    {
        not_realistic[i - i_notrealistic - 1, j] = param[i].Split(tab)[j];
    }
}
for (int i = i_bonded + 2; i < i_notrealistic; i++)
{
```

```
    for (int j = 0; j < 4; j++)
    {
        bonds_param[i - i_bonded - 2, j] = param[i].Split(tab)[j];
    }
}
for (int i = 1; i < i_bonded; i++)
{
    param_sab = param[i].Split(tab);
    for (int j = 0; j < 4; j++)
    {
        param_sub[j] = param_sab[j + 1];
    }
    string[] clone = new string[param_sub.Length];
    for (int k = 0; k < clone.Length; k++)
    {
        clone[k] = param_sub[k];
    }
    par.Add(param_sab[0], clone);
}
int i_angels = 0;
for (int i = 0; i < connections.Length; i++)
{
    if (connections[i].StartsWith("#angles") == true)
    {
        i_angels = i;
    }
}
i_angels = i_angels + 1;
for (int i = 1; i < i_angels - 1; i++)
{
    bonds_sab = connections[i].Split(tab);
    bonds_sub[i - 1, 0] = bonds_sab[0];
    bonds_sub[i - 1, 1] = bonds_sab[1];
}
```

```
int i_dihedrals = 0;
for (int i = 0; i < connections.Length; i++)
{
    if (connections[i].StartsWith("#dihedrals") == true)
    {
        i_dihedrals = i;
    }
}
i_dihedrals = i_dihedrals + 1;
for (int i = i_angels; i < i_dihedrals - 1; i++)
{
    angels_sab = connections[i].Split(tab);
    angels_sub[i - 16, 0] = angels_sab[0];
    angels_sub[i - 16, 1] = angels_sab[1];
    angels_sub[i - 16, 2] = angels_sab[2];
}
for (int i = i_dihedrals; i < connections.Length; i++)
{
    dihedrals_sab = connections[i].Split(tab);
    dihedrals_sub[i - 33, 3] = dihedrals_sab[3];
}
string decSep = NumberFormatInfo.CurrentInfo.NumberDecimalSeparator;
for (int i = 1; i < coord.Length; i++)
{
    coord_sab = coord[i].Split(tab);
    string name = coord_sab[1];
    ptNames[i - 1] = name;
    for (int j = 0; j < 6; j++)
    {
        if (j == 0)
        {
    }
}

using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.ComponentModel;
using System.Runtime.CompilerServices;
using System.Windows.Input;
using System.Windows;
using System.Collections.ObjectModel;
using System.Windows.Interactivity;

{    public partial class Form1 : Form
        public Form1()
      {
      InitializeComponent();
      }
      private void button1_Click(object sender, EventArgs e)
      {
      textBox2.Text = Me.Codec(textBox1.Text, (int)numericUpDown1.Value);
      }
      private void button2_Click(object sender, EventArgs e)
      {
      textBox3.Text = Me.Codec(textBox2.Text, -(int)numericUpDown1.Value);
      }
   }
}
class Clent
{
   string le;
   public Clent(string m)
   {
      le = m;
   }
   public string Repl(string m, int key)
   {
```

```
      int pos=le.IndexOf(m);
      if (pos == -1) return "";
  pos = (pos+key) % le.Length;
if (pos < 0) pos += le.Length;
return le.Substring(pos, 1);
 }
}
public static void Main()
    {
        try
        {
            using (var myAes = Aes.Create())
            {
                byte[] encrypted = EncryptStringToBytesAes(original, myAes.Key, myAes.IV);
                string roundtrip = DecryptStringFromBytesAes(encrypted,myAes.Key, myAes.IV);
                Console.WriteLine("Original:   {0}", original);
                Console.WriteLine("Round Trip: {0}", roundtrip);
            }
        }
        catch (Exception e)
        {
        }
    }
static byte[] EncryptStringToBytesAes(string plainText, byte[] Key,byte[] IV)
    {
        if (plainText == null || plainText.Length <= 0)
            throw new ArgumentNullException("plainText");
        if (Key == null || Key.Length <= 0)
            throw new ArgumentNullException("Key");
        if (IV == null || IV.Length <= 0)
            throw new ArgumentNullException("IV");
        byte[];
        using (Aes aesAlg = Aes.Create())
        {
            aesAlg.Key = Key;
```

```
            aesAlg.IV = IV;
            ICryptoTransform encryptor = aesAlg.CreateEncryptor(aesAlg.Key, aesAlg.IV);
                using (var msEncrypt = new MemoryStream())
            {
                {
                    {
                        swEncrypt.Write(plainText);
                    }
                    encrypted = msEncrypt.ToArray();
                }
            }
        }
    return encrypted;
    }
static string DecryptStringFromBytesAes(byte[] cipherText, byte[] Key, byte[] IV)
    {
        if (cipherText == null || cipherText.Length <= 0)
            throw new ArgumentNullException("cipherText");
        if (Key == null || Key.Length <= 0)
            throw new ArgumentNullException("Key");
        if (IV == null || IV.Length <= 0)
            throw new ArgumentNullException("IV");
        string plaintext;
        using (Aes aesAlg = Aes.Create())
        {
            aesAlg.Key = Key;
            aesAlg.IV = IV;
            ICryptoTransform decryptor = aesAlg.CreateDecryptor(aesAlg.Key, aesAlg.IV);
            using (var msDecrypt = new MemoryStream(cipherText))
            {
              using (var csDecrypt = new CryptoStream(msDecrypt, decryptor,
CryptoStreamMode.Read))
                {
                    using (var srDecrypt = new StreamReader(csDecrypt))
                    {
```

```
                        plaintext = srDecrypt.ReadToEnd();
                    }
                }
            }
        }
        return plaintext;
    }
KeyExpansion(byte key[4*Nk], word w[Nb*(Nr+1)], Nk)
begin
word temp
i = 0
while (i < Nk)
w[i]=(key[4*i], key[4*i+1], key[4*i+2], key[4*i+3])
i = i+1
end while
i = Nk
while (i < Nb * (Nr+1))
temp = w[i-1]
if (i mod Nk = 0
temp = (RotWord(temp)) xor Rcon[i/Nk]
else if (Nk > 6 and i mod Nk = 4)
temp = SubWord(temp)
end if
w[i] = w[i-Nk] xor temp
i = i+1
end while
end
byte[] B_0 = new byte[16];
private static ushort ExtractBit(ushort source, ushort number)
{
if (number < 16)
{
return (ushort)((source >> number) & 1);
}
```

```
return 0;
}
private static ushort SetBit(ushort source, ushort number, ushort val)
{
if (number>15) return 0;
if (val>1) return 0;
if (ExtractBit(source, number) == val) return source;
return (ushort)(source ^ (1< }
private static byte PolynomsMuliply(byte a, byte b)
{
ushort result = 0;
ushort temp;
for (ushort i = 0; i < 15; i++)
{
temp = 0;
for (ushort k = 0; k <= i; k++)
{
ushort ak = ExtractBit(a, k);
ushort bik = ExtractBit(b, (ushort)(i - k));
temp = (ushort) (temp ^ (ak & bik));
}
result = SetBit(result, i, temp);
}
ushort mod = 283;
return (byte)result;
}
private static void MixColumns(ref byte[,] state)
{
}
state = new_state;
return;
}
private static void ShiftRows(ref byte[,] state)
{
byte temp1 = state[1, 0];
```

```
state[1, 0] = state[1, 1];
state[1, 1] = state[1, 2];
state[1, 2] = state[1, 3];
state[1, 3] = temp1;
temp1 = state[2, 0];
byte temp2 = state[2, 1];
state[2, 0] = state[2, 2];
state[2, 1] = state[2, 3];
state[2, 2] = temp1;
state[2, 3] = temp2;
temp1 = state[3, 0];
temp2 = state[3, 1];
byte temp3 = state[3, 2];
state[3, 0] = state[3, 3];
state[3, 1] = temp1;
state[3, 2] = temp2;
state[3, 3] = temp3;
}
private static void (ref byte[,] state)
{
byte x;
byte y;
for (int i = 0; i < 4; i++)
{
for (int j = 0; j < 4; j++)
{
x = (byte)(state[i, j] >> 4);
y = (byte)(state[i, j] & 0xf);
state[i, j] = S(x, y);
}
}
}
private static void AddRoundKey(ref byte[,] state, uint w0, uint w1, uint w2, uint w3 )
{
```

```
state[0, 0] = (byte)(state[0, 0] ^ (w0 >> 24));
}
private static uint[] KeyExpansion(byte[] key)
{
uint[] w = new uint[60];
for (int i = 0; i < 60; i++) w[i] = 0;
for (int i = 0; i < 8; i++)
{
w[i] = (((uint)key[4 * i]) << 24) ^ ((((uint)key[4 * i + 1]) << 16)) ^ (((uint)key[4 * i + 2]) <<  ^
key[4 * i + 3];
}
for (int i = 8; i < 60; i++)
{
uint temp = w[i - 1];
if ((i %  ==
{
temp = SubWord(RotWord(temp)) ^ Rcon[i / 8];
}
else if (i % 8 == 4)
{
temp = SubWord(temp);
}
w[i] = w[i - 8] ^ temp;
}
return w;
}
private static uint (uint p)
{
byte p0 = (byte)(p >> 24);
byte p1 = (byte)(p >> 16);
byte p2 = (byte)(p >> 8);
byte p3 = (byte)p;
byte x = (byte)(p0 >> 4);
byte y = (byte)(p0 & 0x0f);
p0 = S(x, y);
```

```
x = (byte)(p1 >> 4);
y = (byte)(p1 & 0x0f);
p1 = S(x, y);
x = (byte)(p2 >> 4);
y = (byte)(p2 & 0x0f);
p2 = S(x, y);
x = (byte)(p3 >> 4);
y = (byte)(p3 & 0x0f);
p3 = S(x, y);
return ((((uint)p0) << 24) ^ (((uint)p1) << 16) ^ (((uint)p2) <<  ^ p3);
}
private static uint RotWord(uint w)
{
uint temp = w >> 24;
temp = (w <<  ^ temp;
return temp;
}
}
}
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

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