**Дніпровський ліцей інформаційних технологій**

**при Дніпропетровському національному університеті**

**імені Олеся Гончара**

**Випускна робота**

**на тему:**

**«Знаходження Z фальшивих монет серед N справжніх»**

**Виконавець:**

**ліцеїст 11-Г класу**

**Олешко Олександр**

**Науковий керівник:**

**Ентін Й. А. \_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Дніпро**

**2017**

**ЗМІСТ**

[**Вступ** 4](#_Toc499507196)

[***Пошук монет методами сортування*** 5](#_Toc499507197)

[***Чим метод, використаний у програмі, кращий від методу попарного зважування?*** 7](#_Toc499507198)

[***Алгоритм пошуку фальшивих монет*** 8](#_Toc499507199)

[***Алгоритм пошуку фальшивих монет*** 11](#_Toc499507200)

[**Опис програми** 12](#_Toc499507201)

[***Теорія*** 14](#_Toc499507202)

[***Практика*** 14](#_Toc499507203)

[***Графіки порівняння алгоритму з попарним зважуванням*** 15](#_Toc499507204)

[**Посібник користувача** 17](#_Toc499507205)

[**Програмно-апаратні вимоги** 21](#_Toc499507206)

[**Висновки** 22](#_Toc499507207)

[**Використані джерела** 23](#_Toc499507208)

# Вступ

**Умова задачі.** Існує N монет, серед яких Z фальшиві – легші за справжні, причому маси фальшивих можуть бути різними. Потрібно знайти мінімальну кількість зважувань на шалькових терезах без важків, за які можливо знайти усі фальшиві монети.

**Мета роботи.** Розробити алгоритм, який визначає усі фальшиві монети за мінімальну кількість зважувань на шалькових терезах без важків.

**Актуальність роботи.** Аналогічні задачі можна розв’язувати для аналогічних об’єктів методами сортування. Тому має сенс визначити, чи існують швидші від сортування алгоритми розв’язання даної, що є практично важливим.

**Головні задачі випускної роботи.**

1. Оптимізація алгоритму знаходження фальшивих монет із різними масами.
2. Візуалізація вище зазначеного.

**Теоретична частина задачі**

***Пошук монет методами сортування***

Метод QuickSort:

Швидке сортування — алгоритм сортування, добре відомий, як алгоритм розроблений Тоні Гоаром, який не потребує додаткової пам'яті. Однак, у найгіршому випадку необхідні n2 порівнянь. Цей алгоритм швидкий: він зазвичай більш ніж удвічі швидший порівняно з сортуванням злиттям.

Ідея алгоритму полягає в переставлянні елементів масиву таким чином, щоб його можна було розділити на дві частини і кожний елемент з першої частини був не більший за будь-який елемент з другої. Впорядкування кожної з частин відбувається рекурсивно. Алгоритм швидкого сортування може бути реалізований як у масиві, так і в двозв'язному списку.

Швидке сортування є алгоритмом на основі порівнянь і не є стабільним.

Попарне зважування:

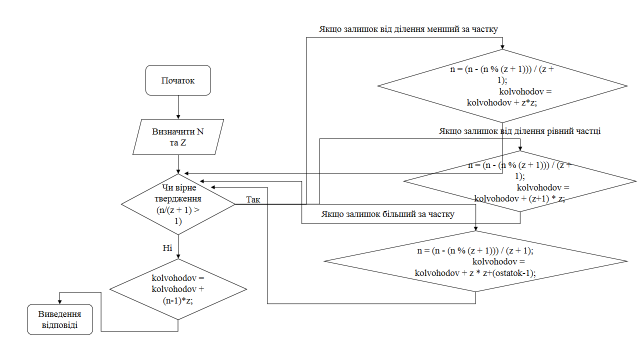
Попарне зважування – алгоритм пошуку фальшивих монет методом попарного зважування монет на шалькових терезах. *Кількість кроків у цьому методі: N-1, коли N – кількість фальшивих монет. Немає значення яка кількість фальшивих монет.*

Метод, використаний у програмі:

Його ідея закладається у тому, що N монет ділиться на Z + 1 купу та залишок. Ці купи порівнюються, і одна з них (у найгіршому випадку, а ми розглядаємо саме його), тобто найважча, не містить у собі фальшиві монети. Її ми викреслюємо. Кількість порівнянь цих куп буде дорівнювати Z, якщо наш залишок (від ділення) менший, то ми його не розглядаємо, адже це кращий варіант для нас. Якщо він має таку ж саму кількість монет, як і інші купи, то ми його порівнюємо разом з іншими і викреслюємо дві купи. У даному випадку кількість зважувань буде дорівнювати Z + 1. У випадку, коли ми маємо залишок, більший за усі інші купи, ми працюємо з ним окремо. Тобто, порівнюємо у ньому усі фальшиві монети. І працюємо з купами. У цьому випадку кількість кроків буде дорівнювати Z + кількість монет у залишку – 1.

Виконуємо ці дії, поки ми маємо можливість ділити наші монети на Z + 1 купу. Після цього порівнюємо усі монети, що залишилися і маємо загальну кількість зважувань.

Пояснення до блок-схеми: N – кількість монет загальна. Z – кількість фальшивих монет. Kolvohodov – кількість кроків, а ostatok – остача від ділення.

Опис алгоритму:

*Рис. 1*

Після зчитки N (загальної кількості монет) та Z (кількості фальшивих монет) перевіряється умова чи N > Z (за умовою задачі N > Z), та чи дорівнює Z нулю. Якщо Z дорівнює нулю, то кількість кроків також дорівнює нулю, адже ми не маємо фальшивих монет у нашій купі. Якщо ж Z більше за нуль, то перевіряється твердження (дамо йому назву: *Твердження 1*) чи ми маємо можливість поділити усю кількість монет (N) на кількість куп Z+1 (тобто на одну більше, ніж кількість фальшивих монет). Якщо ж ми маємо таку можливість, то ми розглядаємо три випадки:

1. Якщо залишок після ділення N монет на Z + 1 менший за частку. У такому випадку, якщо у залишку буде знаходитись фальшива(і) монета(и), то це буде не найгірший випадок, а ми розглядаємо найгірші. Отже, ми не звертаємо уваги на залишок та працюємо лише з купами, які поділили. Тобто порівнюємо ці купи, та, знайшовши найважчу, відмовляємося від неї, адже у неї не може бути фальшивих монет. Тобто кількість зважувань буде дорівнювати кількості зважувань до цього. Треба додати ще Z \* Z порівнянь.
2. Якщо залишок після ділення дорівнює частці, то можливий варіант, що у частці також буде знайдена фальшива монета. Тоді порівнюємо усі купи та частку. Знайшовши дві найважчі, відкладаємо них. Кількість зважувань у даному випадку - Z \* (Z + 1).
3. Якщо залишок після ділення більший за частку, працюємо спочатку з залишком (використовуємо попарне порівняння), а потім - з іншими купами. У даному випадку кількість зважувань - Z \* Z + остача – 1.

Кожен наступний раз ми звертаємося до «*Твердження 1*», доки умова виконується.

Наступний крок: попарне зважування для усіх куп, що залишилися. Кількість кроків буде дорівнювати кількості кроків та додаванню до цього числа (N – 1) \* Z. Тобто кількість кроків попарного зважування для Z куп.

***Чим метод, використаний у програмі, кращий від методу попарного зважування?***

Метод, використаний у програмі, кращий від попарного зважування тим, що воно сортує усі маси монет, а цей метод зразу ж виявляє, де саме немає фальшивих монет (тобто знаходить найважчу купу монет, якщо фальшивих монет Z, а куп Z+1) та прибирає цю купу. Це призводить до скорочення зважувань задля більш швидшого пошуку фальшивих монет.

Існує умова, при виконанні якої, алгоритм, використаний у програмі, не є більш вигідним за попарне зважування. Тобто, загальна кількість монет (N) при діленні на кількість фальшивих монет + 1 (Z+1) дає менше, ніж 2, то кількість зважувань даного алгоритму буде дорівнювати кількості зважувань при використанні попарного зважування.

*Рис. 2*

Розглянемо вигідність алгоритму на прикладі. Нехай існує 12 монет, серед яких 3 фальшивих. Для попарного зваження, щоб визначити фальшиві монети, нам потрібно зробити 11 порівнянь. Порахуємо кількість зважувань, потрібну для знахождення фальшивих монет за допомогою алогоритму, використаного у програмі:

Ми маємо N = 12, Z = 3, розбиваємо 12 на 3 + 1 купу. Тобто ми маємо 4 купи по 3 монети. Це вже 3 зважування. Одна з цих куп зайва (найважча). Викреслюємо її та працюємо з іншими. 3 монети неможливо поділити на 4 купи, порівнюємо самі монети: 3 купи по 3 монети. Це ще 6 зважувань. Загалом 9. Це вигідніше за попарне зважування.

Вигіднішим алгоритм є через те, що ми виключаємо деяку кількість монет, у яких фальшиві монети не містяться і ми у цьому впевнені.

***Алгоритм пошуку фальшивих монет***

Розглянемо алгоритм пошуку однієї фальшивої монети для загальної кількості монет, що дорівнює N:

Перевіряємо умову чи більше число N за кількість фальшивих монет. Потім відбувається перевірка чи ми маємо можливість поділити на дві купи (тобто на одну більше, ніж кількість фальшивих монет).

|  |  |  |
| --- | --- | --- |
| # | Кількість монет у купі | Кількість кроків задля знаходження фальшивої монети |
| 1. | 1 | Усі монети фальшиві. 0 зважувань. |
| 2. | 2 | 1 зважування. |
| 3. | 3 | 2 зважування. |
| 4. | 4 | 2 зважування. |
| 5. | 5 | 2 зважування. |
| 6. | 6 | 3 зважування. |
| 7. | 7 | 3 зважування. |
| 8. | 8 | 3 зважування. |
| 9. | 9 | 3 зважування. |
| 10. | 10 | 3 зважування. |

Відбувається пошук кількості зважувань, відштовхуючись від частки та залишку від ділення. Повторюємо перевірку умови можливості поділу на Z + 1, тобто на дві купи.

*Таблиця 1*

Після неможливості виконання умови до кількості порівнянь додаємо кількість монет купи, помноженої на два (кількість куп, де є фальшиві).

*Таблиця 1*. Зображення кількості кроків для пошуку 1 фальшивої монети серед від 3 до 10 монет у купі. Проаналізувавши, розуміємо, що цей метод більш вигідний, ніж попарне зважування.

Розглянемо випадки, коли ми маємо 12 фальшивих монет та від 2 до 6 фальшивих.

Маємо 12 монет, серед них 2 фальшиві. Ділимо їх на 3 купи. І маємо 3 купи по 4 монети. Це 2 зважування. Одна з цих куп (найважча) не буде мати фальшивих монет. Її відкидаємо. Залишилося 2 купи по 4 монети. Поділяємо кожну купу на 3 та залишок. Маємо 3 купи по 1 монеті та 1 у залишку. Це 6 зважувань. Загалом 8.

Маємо 12 монет, серед яких 3 фальшиві. Розділимо їх на 4 купи по 3 монети. Найважчу відкидаємо. Це 3 зважування. Потім працюємо з купами, що залишились. Можливості поділити 3 монети на 4 купи не маємо, отже у кожній купі попарно зважуємо монети. Найлегші і будуть фальшивими. Це 6 кроків. Загалом 9.

Маємо 12 монет, серед яких 4 фальшиві. Розділимо їх на 5 куп по 2 монети та залишок 2 (його також порівнюємо з іншими купами). Дві найважчі купи відкидаємо. Це 5 зважувань. Потім працюємо з купами, що залишились. Можливості поділити 2 монети на 4 купи не маємо, отже у кожній купі попарно зважуємо монети. Найлегші і будуть фальшивими. Це 4 кроки. Загалом 9.

Маємо 12 монет, серед яких 5 фальшивих. Розділимо їх на 6 куп по 2 монети. Найважчу купу відкидаємо. Це 5 зважувань. Потім працюємо з купами, що залишились. Можливості поділити 2 монети на 5 куп не маємо, отже у кожній купі попарно зважуємо монети. Найлегші і будуть фальшивими. Це 5 кроків. Загалом 10.

Маємо 12 монет, серед яких 6 фальшивих. Можливості поділити 12 монет на 7 куп, та виконанні рівняння N / Z + 1>=2 не маємо, отже кількість кроків алгоритму буде співпадати з кількістю кроків попарного зважування. Загалом N – 1, тобто 11.

**Теоретична частина з інформатики**

В ході написання коду до даної програми використовувалися деякі функції програмування.

***Алгоритм пошуку фальшивих монет***

Найважчим у алгоритмі пошуку фальшивих монет було порівняння залишку та діленого. Дії, що виконувалися, якщо залишок менший:

if ((n-(n%(z+1)))/(z+1)>n%(z+1))

{

listBox1.Items.Add(((n - (n % (z + 1))) / (z + 1)).ToString()+" монет \* "+(z+1).ToString()+" куп(и) +залишок:"+(n%(z+1)).ToString()+". Кількість зважувань: "+(z\* st).ToString());

n = (n - (n % (z + 1))) / (z + 1);

kolvohodov = kolvohodov + z\*st;

}

Якщо залишок дорівнював діленому:

else if((n - (n % (z + 1))) / (z + 1) == n % (z + 1))

{

listBox1.Items.Add(((n - (n % (z + 1))) / (z + 1)).ToString() + " монет \* " + (z + 1).ToString() + " куп(и) +залишок:" + (n % (z + 1)).ToString() + ". Кількість зважувань: " + (z \* st).ToString());

n = (n - (n % (z + 1))) / (z + 1);

kolvohodov = kolvohodov + (z+1) \* st;

}

Коли залишок був більшим за ділене:

else

{

ostatok1 = n % (z + 1);

listBox1.Items.Add(((n - (n % (z + 1))) / (z + 1)).ToString() + " монет \* " + (z + 1).ToString() + " куп(и) +залишок (з попарним зважуванням):" + (n % (z + 1)).ToString() + ". Кількість зважувань: " + ((z \* st)+(ostatok1-1)).ToString());

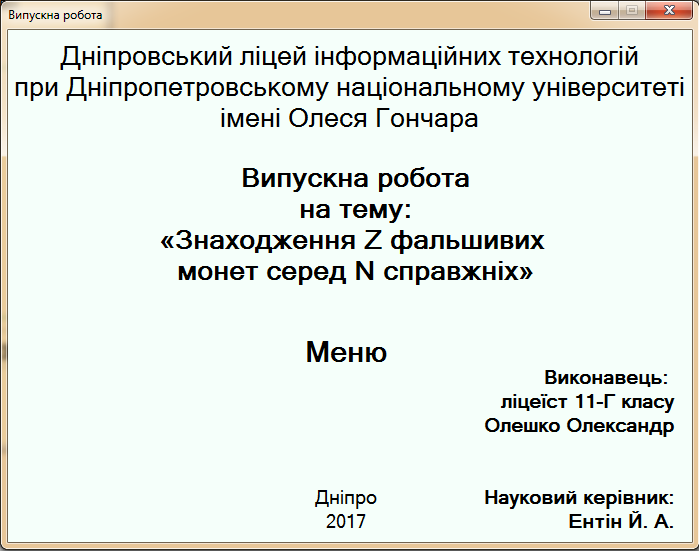
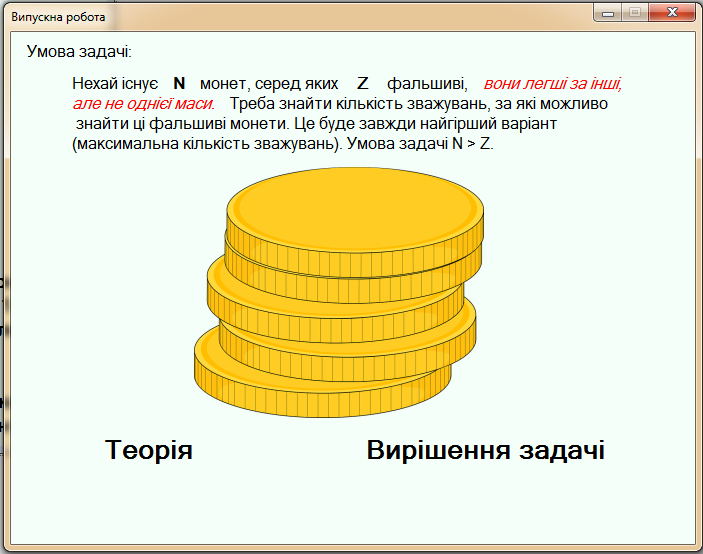
n = (n - (n % (z + 1))) / (z + 1);

kolvohodov = kolvohodov + z \* st+(ostatok1-1);

}

**Опис програми**

Ця робота повністю створена в середовищі програмування Microsoft Visual Studio 2013 WinForms. При запуску програми відкривається титульна сторінка (рисунок 3).



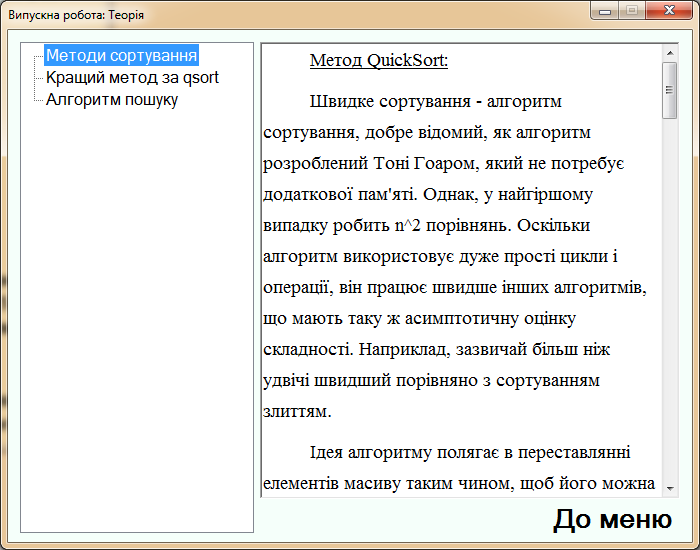
*Рис. 4*

*Рис. 3*

Далі переходимо до головного меню. Із нього можна переходити до інших розділів, зокрема прочитати умову задачі (рис. 4).

Перелік розділів роботи

1. **Теорія** – тут розглянуто різні методи сортування; показано, **чим і за яких умов** метод, використаний у програмі, краще від qsort, зображено алгоритм пошуку одної фальшивої монети серед N справжніх (опис дій).
2. **Практика** – розділ практики, де визначається кількість зважувань, за які можливо знайти усі фальшиві монети.

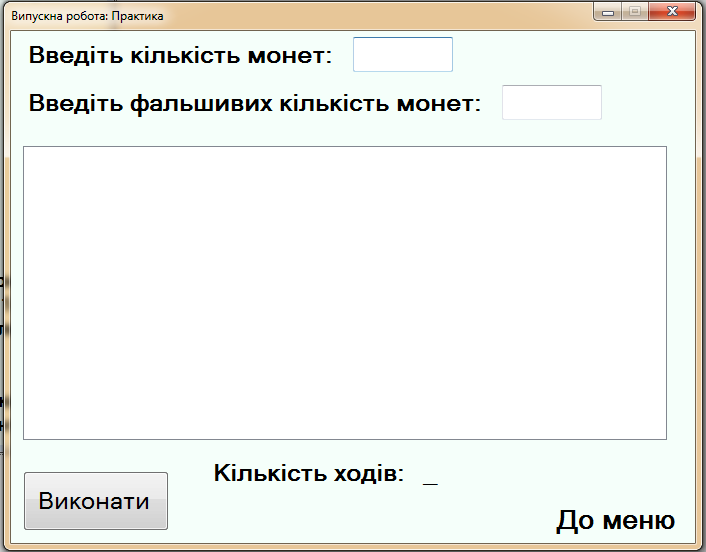


*Рис. 5*

***Теорія***

Перейшовши до розділу «Теорія» (рис. 5) користувач знайде потрібну інформацію.

При натисканні на назву розділу у TreeView буде виводитися матеріал по даній темі.

***Практика***

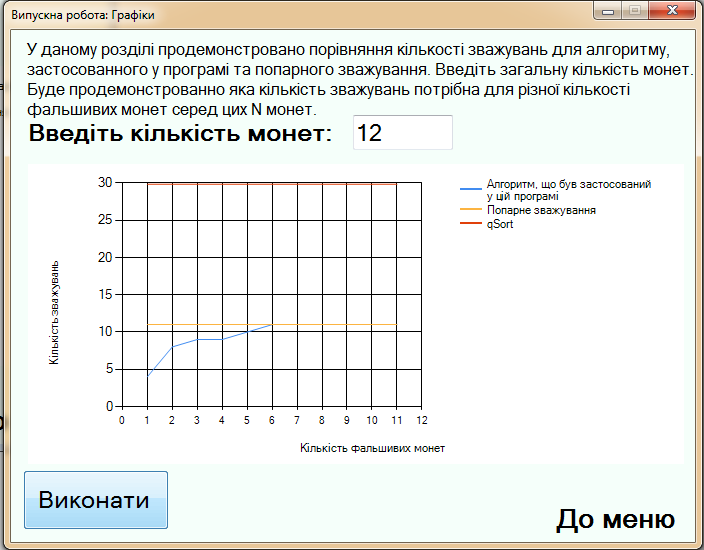
*Рис. 6*

Ознайомившись із теорією, можна перейти до головного розділу «Практика» (рис. 6).

В даному розділі ви можете побачити:

* робочу область, де буде виведено кроки порівнянь при пошуку фальшивих монет;
* функціональну область, де можна ввести дані про загальну кількість монет та кількість фальшивих монет. При натисканні кнопки «Виконати» виводиться список кроків, які виконав алгоритм, та про загальну кількість зважувань. Якщо введено кількість фальшивих монет 0, або рівну загальній кількості монет, ця інформація з’явиться у MessageBox і з’явиться напис щодо введення коректних даних. Якщо ж кількість зважувань буде дорівнювати кількості зважувань при використанні попарного зважування, повідомлення про це буде в MessageBox.

***Графіки порівняння алгоритму з попарним зважуванням***

 У цьому розділі продемонстровано порівняння ефективності використання попарного зважування, qSort’у та алгоритму, використаного у програмі.

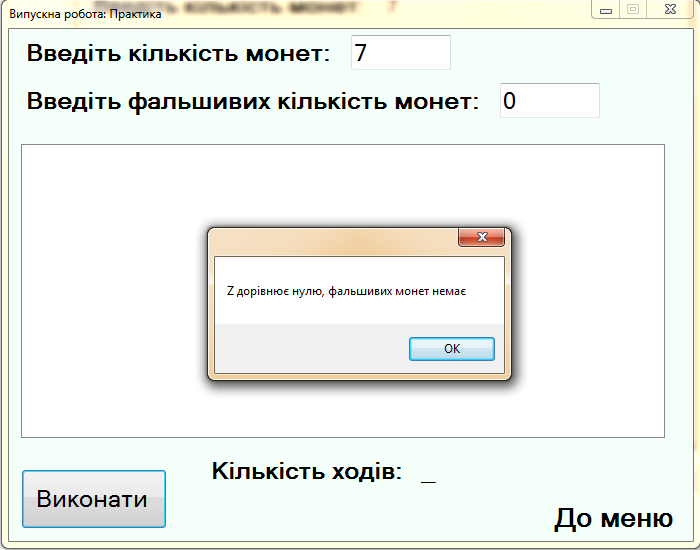
*Рис. 7*

При введенні загальної кількості монет, відбуваються розрахунки, починаючи з одної фальшивої монети, і до N-1, на екран виводиться графік кількості зважувань, потрібних для алгоритму, використаного у програмі, при Z фальшивих монетах, та попарного зважування. Подивившись на графік, робимо висновок, що при рівнянні N / Z + 1 > 2, алгоритм, використаний у програмі, вигідніший за попарне зважування. Загалом алгоритм вигідніший за qSort. У іншому випадку, потрібно також N - 1 зважувань.

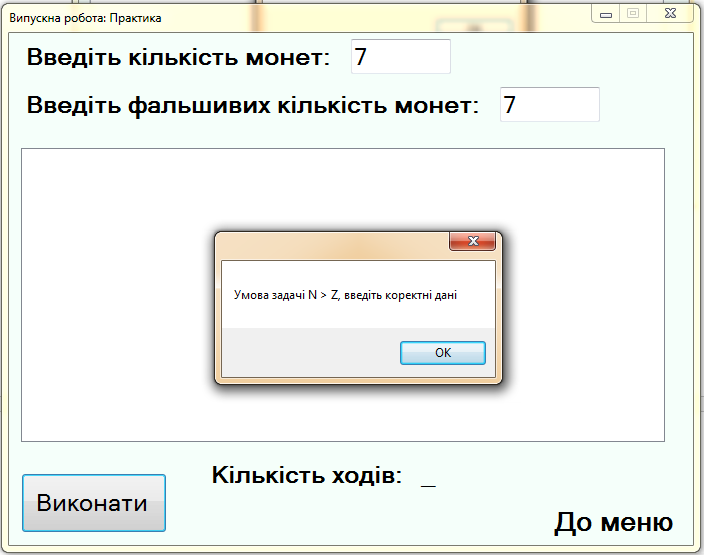
**Посібник користувача**

Представлено навігацію по проекту. Перейшовши до розділу «Практика», побачите, як він працює.

В розділі «Практика» можна визначити необхідну кількість зважувань для знаходження фальшивих монет серед справжніх.

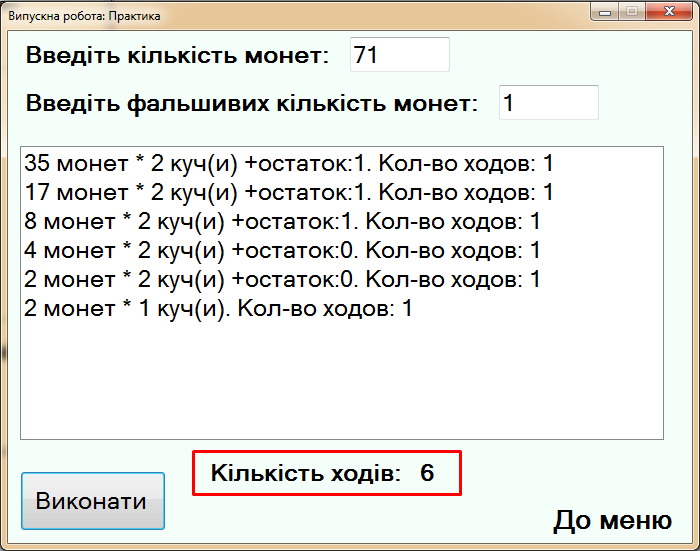
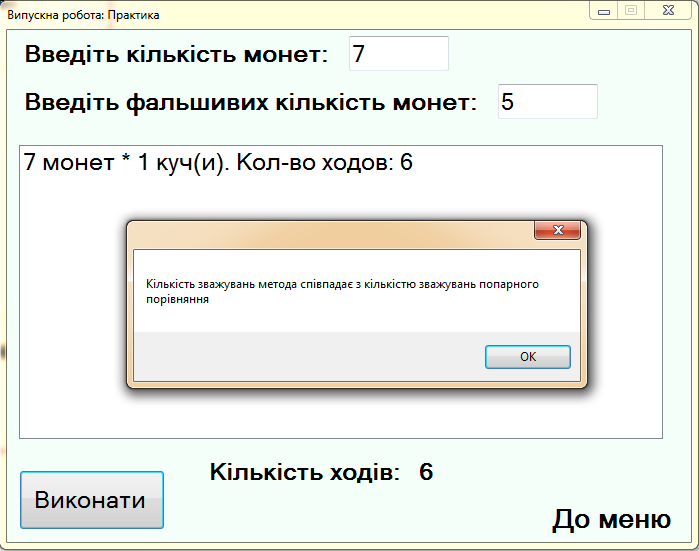
Для того, щоб почати, потрібно ввести дані. Ми маємо обмеження: кількість фальшивих монет має бути більшою від нуля й меншою від кількості всіх монет. Якщо ці умови не виконані, з’явиться напис про необхідність ввести коректні дані (рис. 8 та 9).

*Рис. 8*

Після введення коректних даних, буде проведено кількість потрібних зважувань, задля знаходження серед N монет Z фальшивих. До listBox буде виведено кроки, які виконувалися для знаходження фальшивих монет. Кількість потрібних зважувань буде виведено до спеціального label (рис. 10).

*Рис. 9*

Якщо ж кількість зважувань алгоритму буде дорівнювати кількості зважувань при використанні попарних зважувань, то ця інформація буде виведена до MessageBox (рис. 11).



*Рис. 11*

*Рис. 10*

**Структурна схема**

**Програмно-апаратні вимоги**

Операційна система:

Windows 8/ Windows 7/ Windows XP/ Windows Vista

Програмна платформа:

Microsoft .NET Framework 4

Частота процесора:

1.5 ГГц і вище

Оперативна пам'ять:

1 ГБ і вище

Місце на жорсткому диску:

20 МБ

**Висновки**

Під час виконання роботи знайдено цікаві статті з теми «Пошук фальшивих монет серед справжніх» та мультимедійні сайти з пізнавальною інформацією. Обравши найголовніші аспекти теорії, мені вдалося створити теоретичну довідку, яка ввійшла до програми.

Поглиблено знання з програмування. Для редагування зображень використовувався графічний редактор Paint.

Створена оптимізуюча програма під назвою «Пошук Z фальшивих монет серед N справжніх, що є легшими, але різних мас, за допомогою шалькових ваг». Вона дозволяє користувачеві прискорити пошук фальшивих монет. Дану програму можна використовувати не лише як пошук фальшивих монет, а й у аналогічних задачі для інших подібних об’єктів, що часто зустрічаються, тому розв’язання даної задачі має практичну цінність.

Усі задачі випускної роботи виконано.

**Використані джерела**

* «Олімпіадні задачі з інформатики» Пасіхова й Непомнящого,
* «Інформатика. Олімпіадні завдання з розв’язаннями» Мельника,
* «Теорія графів у школі» Скляр
* «Программирование в алгоритмах» и «Дискретная математика» Окулова,
* «Алгоритмы. Руководство для разработки» Скиены
* «Решение сложных и олимпиадных задач по программированию» Долинского,
* «Алгоритмы на С++» Седжвик
* «Алгоритмы и программы» Порублёва и Ставровского.
* Підручники з C# Зіборова Культіна, Фльонова, Лабора://Informatics/Programming/C#),
* Internet-джерела:
  + Algolist.manual.ru
  + Informatics.mccme.ru
  + Olympiads.ru
  + Olymp.vinnica.ua