

Лабораторна робота №1. Створення проекту консольного застосування.

Тема: Створення проекту консольного застосування.

Мета заняття: навчитись створювати проектні рішення з проектами консольних застосувань мовою C#; навчитись визначати стартовий проект, з якого буде починатись виконання програми; навчитись використовувати простори імен, викликати статичні методи класів, отримувати та змінювати значення статичних властивостей класів; навчитись визначати параметри вікна консолі; навчитись оголошувати змінні та використовувати їх значення у виразах з метою виконання арифметичних, логічних та бітових операцій; навчитись виконувати операції введення і виведення даних, перетворення типів та форматування виведення; навчитись формувати набори тестових даних, які забезпечують перевірку правильності виконання програми.

Теоретичні відомості

Структура рішення .NET¹

Рішення містить один або кілька проектів, ресурси, необхідні цим проектам, можливо, додаткові файли, які не входять в проекти. Один з проектів рішення повинен бути виділений і призначений стартовим проектом. Виконання рішення починається зі стартового проекту.

Стартовий проект повинен мати точку входу – клас, що містить статичну процедуру з ім'ям Main, якій автоматично передається керування в момент запуску рішення на виконання.

Проект складається з класів, зібраних в одному або декількох просторах імен. Простори імен дозволяють структурувати проекти, що містять велику кількість класів, об'єднуючи в одну групу близькі класи. Якщо над проектом працює кілька виконавців, то, як правило, кожен з них створює свій простір імен. Крім структуризації, це дає можливість присвоювати класам імена, не замислюючись про їхню унікальність. У різних просторах імен можуть існувати однойменні класи. Проект – це основна одиниця, з якої працює програміст. Він вибирає тип проекту, а Visual Studio (або інше середовище розробки) створює скелет проекту відповідно до обраного типу.

Методи²

Метод – це блок коду, який містить ряд інструкцій. Програма ініціює виконання інструкцій, викликаючи метод і вказуючи всі аргументи, необхідні для цього методу. У C# всі інструкції виконуються в контексті методу. Метод Main є точкою входу для кожної програми C#, і він викликається загальномовним середовищем виконання (CLR) в момент запуску програми.

Методи оголошуються в класі або структурі, з визначенням рівня доступу (public, private тощо), необов'язкових модифікаторів, типу значення, що повертається, назви методу та будь-яких параметрів методу. Ці частини разом є заголовком методу.

Параметри методу записують в дужках і розділяють комами. Порожні дужки вказують, що параметри методу не потрібні.

¹ <http://www.intuit.ru/studies/courses/2247/18/lecture/542>

² <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/methods>

Властивості³

Властивість є членом, який надає гнучкий механізм для читання, запису або обчислення значення приватного поля. Властивості можна використовувати так, ніби вони є публічними членами даних, але вони насправді є спеціальними методами, які називаються "accessors". Це дає змогу легко отримувати доступ до даних і допомагає забезпечити безпеку та гнучкість методів.

Властивості дозволяють класу визначати загальнодоступний спосіб отримання та встановлення значень при приховуванні коду реалізації або перевірки.

Статичні класи і члени статичних класів⁴

Статичний клас в основному такий же, як і нестатичних клас, але є одна відмінність: не можна створювати екземпляри статичного класу.

Статичний клас може використовуватися як звичайний контейнер для наборів методів, які працюють на вхідних параметрах.

Нестатичний клас може містити статичні методи, поля, властивості або події. Статичний член викликається для класу навіть в тому випадку, якщо не створено екземпляр класу. Доступ до статичного члени завжди виконується по імені класу, а не примірника... Статичні методи і властивості не можуть звертатися до нестатичних полів ... в типі, що їх містить, і вони не можуть звертатися до змінної примірника об'єкта, якщо він не передається явно в параметрі методу.

Клас Console⁵

Консоль – це вікно операційної системи, в якому користувачі взаємодіють з операційною системою або текстовим консольним додатком, здійснюючи введення тексту за допомогою клавіатури комп'ютера і зчитуючи текстові дані з терміналу комп'ютера... Клас Console надає базову підтримку для додатків, що зчитують символи з і записуючих символи в консоль.

Зокрема у класі Console визначені наступні властивості:

BackgroundColor – повертає або встановлює колір фону тексту, що виводиться (тип *ConsoleColor*).

ForegroundColor – повертає або встановлює колір тексту, що виводиться (тип *ConsoleColor*).

InputEncoding – повертає або встановлює значення кодування тексту, що вводиться.

OutputEncoding – повертає або встановлює значення кодування тексту, що виводиться.

Title – повертає або встановлює текст заголовка вікна консолі.

Деякі статичні методи класу Console, специфічні для консольного введення–виведення:

Clear – очищає буфер консолі і вікно консолі від тесту.

Read – читає наступний символ зі стандартного потоку введення.

ReadKey – отримує інформацію про натиснуту клавішу (об'єкт класу *ConsoleKeyInfo*).

ReadLine – повертає наступний рядок тексту зі стандартного потоку введення.

Write – здійснює виведення інформації в стандартний потік виведення.

³ <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/objects>

⁴ <https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/classes-and-structs/static-classes-and-static-class-members>

⁵ <https://docs.microsoft.com/ru-ru/dotnet/api/system.console?view=netframework-4.8>

WriteLine – здійснює виведення інформації в стандартний потік виведення доповнюючи рядок символом «\n» (переводить текст на наступний рядок).

Клас Encoding⁶

Клас Encoding представляє кодування символів.

Кодування – це процес перетворення набору символів Юнікоду в послідовність байтів. На відміну від декодування – це процес перетворення послідовності закодованих байтів в набір символів Юнікоду.

Клас Encoding в основному призначений для перетворення між різними кодуваннями і Unicode.

Клас Convert⁷

Статичні методи класу Convert використовуються в основному для підтримки перетворення в базові типи даних в .NET Framework і з них. Підтримуються наступні базові типи: Boolean, Char, SByte, Byte, Int16, Int32, Int64, UInt16, UInt32, UInt64, Single, Double, Decimal, DateTime, String. Крім того, клас Convert включає методи для підтримки інших типів перетворень.

Клас Convert містить статичні методи, які можна викликати для перетворення цілочисельних значень в недесяткові рядкові представлення, а також для перетворення рядків, що представляють недесяткові числа, в цілочисельні значення. Кожен з цих методів перетворення включає аргумент *base*, який дозволяє вказати систему числення: двійкову (основа 2), вісімкову (основа 8), шістнадцяткову (основа 16), а також десяткову (основа 10). Існує набір методів для перетворення кожного з CLS-сумісних цілочисельних типів в рядок, а другий – для перетворення рядка в кожен з примітивних цілочисельних типів.

Клас Math⁸

Клас Math надає константи і статичні методи для тригонометричних, логарифмічних та інших спільних математичних функцій.

Практичні завдання



У процесі виконання завдань лабораторної роботи необхідно формувати набори тестових даних для перевірки правильності виконання програмного коду. Створений код і результати перевірки його роботи потрібно помістити у звіт. Тестувати роботу програми рекомендується після додання чи зміни кожного оператора виведення.

Завдання 1. Створення проекту для виконання арифметичних операцій з дійсними числами

1) Роботу над реалізацією програми почніть зі створення проектного рішення з проектом консольного застосування, назву яких утворить згідно формату:

назва проекту:

Expressions_floating_Номер_варіанту_Прізвище_виконавця

⁶ <https://docs.microsoft.com/ru-ru/dotnet/api/system.text.encoding?view=netframework-4.8>

⁷ <https://docs.microsoft.com/ru-ru/dotnet/api/system.convert?view=netframework-4.8>

⁸ <https://docs.microsoft.com/ru-ru/dotnet/api/system.math?view=netframework-4.8>

назва рішення:

Expressions_Номер_варіанту_Прізвище_виконавця

2) Протестуйте роботу програми.

3) З метою затримки відображення вікна консолі до моменту натискання користувачем довільної клавіші у методі Main класу Program проекту консольного застосування опишіть оператор виклику статичного методу ReadKey класу Console з простору імен System з логічним значенням true у якості аргументу, наприклад (тут і далі курсивом позначені раніше створені рядки коду):

```
static void Main(string[] args) {  
    Console.ReadKey(true);  
}
```

4) Текст умови індивідуального варіанту задачі на виконання арифметичних операцій з дійсними числами використайте у якості коментаря до методу Main класу Program проекту, наприклад:

```
class Program {  
    // Дано початкову швидкість v0, час руху t та прискорення a.  
    // Визначити шлях для випадку рівноприскореного руху.  
    static void Main(string[] args) {
```

5) У тілі методу Main перед оператором виклику методу ReadKey класу Console опишіть:

a) оператор присвоєння статичній властивості Title класу Console значення рядка з назвою проекту, наприклад:

```
Console.Title = "Expressions_floating_0_Stets";
```

b) оператор присвоєння статичній властивості BackgroundColor класу Console значення White перерахування ConsoleColor, наприклад:

```
Console.BackgroundColor = ConsoleColor.White;
```

c) оператор виклику статичного методу Clear класу Console, наприклад:

```
Console.Clear();
```

d) оператор присвоєння статичній властивості ForegroundColor класу Console значення Black перерахування ConsoleColor, наприклад:

```
Console.ForegroundColor = ConsoleColor.Black;
```

e) оператор виклику статичного методу WriteLine класу Console з метою відображення рядка тексту, що вказує на призначення програми, наприклад:

```
Console.WriteLine("Обчислення шляху "  
    + "у випадку рівноприскореного руху");
```

6) Збережіть зміни програмного коду проекту. Протестуйте роботу програми.

7) Якщо текст у вікні консолі відображується неправильно, то:

a) додайте у метод Main перед оператором відображення рядка з умовою задачі оператор присвоєння статичній властивості OutputEncoding класу Console значення властивості Unicode класу Encoding з простору імен System.Text, наприклад:

```
Console.OutputEncoding = Encoding.Unicode;  
Console.InputEncoding = Encoding.Unicode;
```

b) відкрийте вікно властивостей вікна консолі, виберіть закладку Шрифт, вкажіть на використання шрифту Consolas і натисніть кнопку ОК (рисунок 1.1);

c) закрийте вікно консолі і повторно запустіть програму на виконання (рисунок 1.2).

8) Проаналізуйте умову індивідуального варіанту задачі на виконання арифметичний операцій з дійсними числами, сформулюйте тестовий набір значень, що буде використовуватись у якості вхідних даних програми і вручну виконайте обчислення для отримання очікуваного результату роботи програми (хоча б одне значення повинно бути представлене дробовим числом), наприклад:

вхідні дані: $v0 = 10,25$, $t = 2$, $a = 5$

результат: $s = v0 \cdot t + a \cdot t^2 / 2 = 10,25 \cdot 2 + 5 \cdot 2^2 / 2 = 20,5 + 10 = 30,5$

9) У тілі методу Main перед оператором виклику методу ReadKey класу Console опишіть:

a) оператори оголошення змінних типу double, що призначені для збереження значень вхідних даних програми, та їх ініціалізації значеннями з тестового набору, наприклад:

`double v0 = 10.25, t = 2, a = 5;`

b) оператор відображення значень змінних, що зберігають вхідні дані, наприклад:
`Console.WriteLine("\tv0 = {0:F2}, t = {1:F2}, a = {2:F2}", v0, t, a);`

c) оператор обчислення значення виразу згідно умови задачі і збереження результату у змінній типу double, наприклад:

`double s = v0 * t + a * t * t / 2;`

d) оператор відображення результату рішення задачі, наприклад:

`Console.WriteLine("\tШлях s = {0:F2}", s);`

10) Збережіть зміни програмного коду проекту. Протестуйте роботу програми (рисунок 2.1).

Самоконтроль:

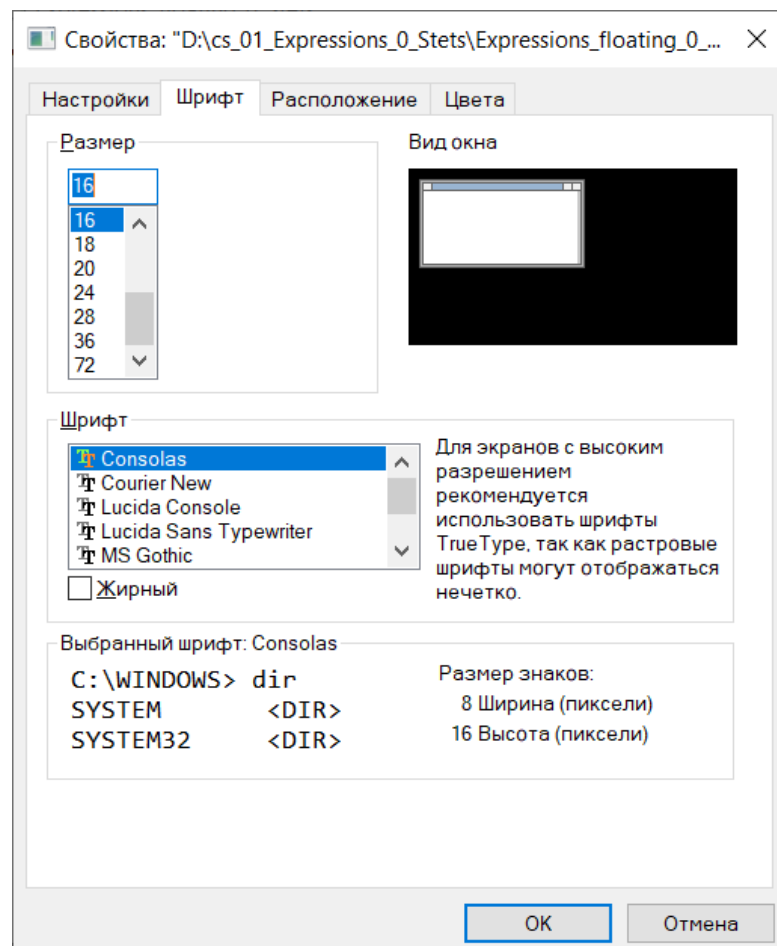


Рисунок 1.1 – Закладка Шрифт вікна властивостей вікна консолі

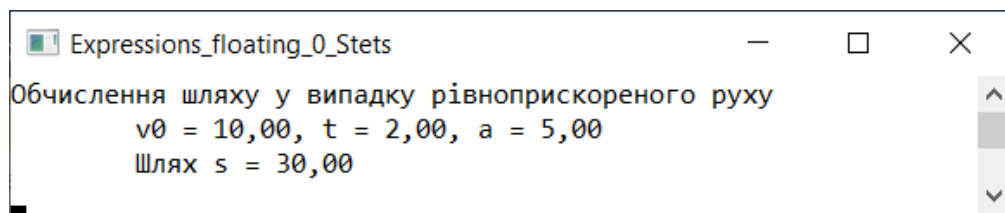


Рисунок 1.2 – Зразок результатів виконання поточного завдання

Завдання 2. Реалізація введення значень вхідних параметрів

- 1) У тілі методу Main після оператора оголошення та ініціалізації змінних, що призначені для збереження вхідних даних, опишіть:
 - a) оператор оголошення змінної str типу string, що призначена для збереження рядків з текстовим представленням чисел, що вводяться користувачем, наприклад:
`double v0 = 10.25, t = 2, a = 5;`
`string str;`
 - b) оператор виклику методу Write класу Console з метою відображення запиту на введення одного із значень з набору вхідних даних, наприклад:
`Console.Write("\tПочаткова швидкість: ");`
 - c) оператор виклику методу ReadLine класу Console з метою отримання значення рядка з текстовим представленням числа, що вводиться користувачем, і збереження посилання на отриманий рядок у змінній str, наприклад:
`str = Console.ReadLine();`
 - d) оператор виклику методу ToDouble класу Convert з метою перетворення значення рядка, на який посилається змінна str у число, що представлене у форматі з плаваючою крапкою подвійної точності, і присвоєння отриманого числового значення відповідній змінній, наприклад:
`v0 = Convert.ToDouble(str);`
- 2) Збережіть зміни програмного коду проекту. Протестуйте роботу програми (рисунок 2.1).
- 3) Протестуйте роботу програми у випадках використання символів коми та крапки у якості розділювача цілої та дробової частин дійсного числа. Проаналізуйте результат (рисунок 2.2).
- 4) Перед оператором перетворення рядкового значення у значення типу double опишіть оператор, який забезпечує заміну у введеному рядку символу крапки на символ коми, наприклад:
`str = str.Replace('.', ',');`
- 5) Збережіть зміни програмного коду проекту. Протестуйте роботу програми.
- 6) У випадку наявності інших змінних, що призначені для збереження значень вхідних даних (у програмі, що використовується у якості зразка такими змінними є змінні t і a, що призначені для збереження часу руху і прискорення), самостійно опишіть оператори, які реалізують діалог з користувачем з метою введення значень для цих змінних, і протестуйте роботу програми.

Самоконтроль:

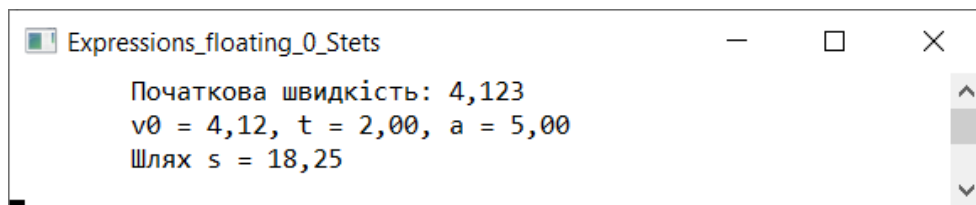


Рисунок 2.1 – Зразок результату виконання обчислень на основі значення, що вводитьься

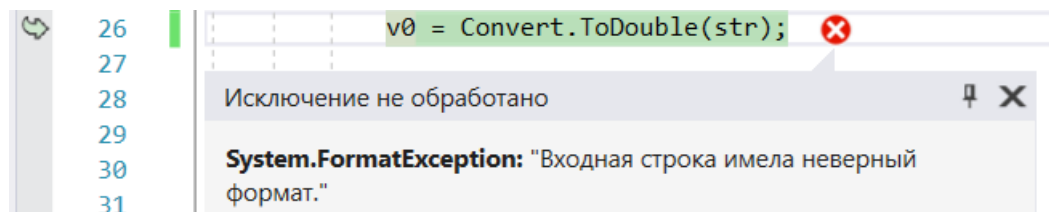


Рисунок 2.2 – Зразок реакції програми у випадку використання символу крапки

Завдання 3. Створення проекту для виконання арифметичний операцій з цілими числами

- 1) Додайте у проектне рішення новий проект консольного застосування з назвою згідно формату:
Expressions_integer_Номер_варіанту_Прізвище_виконавця
- 2) Протестуйте роботу програми.
- 3) Призначте новий проект стартовим проектом і повторно протестуйте роботу програми.
- 4) Текст умови індивідуального варіанту задачі на виконання арифметичний операцій з цілими числами використайте у якості коментаря до методу Main класу Program створеного проекту.
- 5) У тілі методу Main опишіть оператори:
 - а) присвоєння статичній властивості Title класу Console значення рядка з назвою проекту;
 - б) присвоєння властивості BackgroundColor класу Console значення White перерахування ConsoleColor;
 - в) виклику статичного методу Clear класу Console;
 - г) присвоєння властивості ForegroundColor класу Console значення Black перерахування ConsoleColor;
 - д) виклику статичного методу WriteLine класу Console з метою відображення рядка тексту, що вказує на призначення програми;
 - е) виклику методу ReadKey класу Console з логічним значенням true у якості аргументу.
- 6) Збережіть зміни програмного коду проекту. Протестуйте роботу програми.
- 7) Якщо текст у вікні консолі відображується неправильно, то налаштуйте необхідні параметри кодування тексту та виберіть потрібний шрифт для його використання у вікні консолі.
- 8) Проаналізуйте умову індивідуального варіанту задачі на виконання арифметичний операцій з цілими числами, сформууйте тестовий набір значень, що буде

використовуватись у якості вхідних даних програми і вручну виконайте обчислення для отримання очікуваного результату роботи програми.

- 9) Визначтесь з типом даних, який буде використовуватись у програмі для представлення цілих чисел.
- 10) У методі Main перед оператором виклику методу ReadKey класу Console опишіть:
 - a) оператори оголошення змінних вибраного цілочисельного типу, що призначені для збереження значень вхідних даних програми, та їх ініціалізації значеннями з тестового набору;
 - b) оператор відображення значень змінних, що зберігають вхідні дані;
 - c) оператор обчислення значення виразу згідно умови задачі і збереження результату у змінній відповідного типу;
 - d) оператор відображення вхідних даних та результату рішення задачі у десятковому представленні (а, за потреби, в шіснадцятковому і/або в двійковому).
- 11) Збережіть зміни програмного коду проекту. Протестуйте роботу програми (рисунок 3.1).
- 12) Якщо завдання передбачає обчислення і відображення значень інших величин, то реалізуйте відповідні оператори, збережіть зміни програмного коду і протестуйте роботу програми.

Самоконтроль:

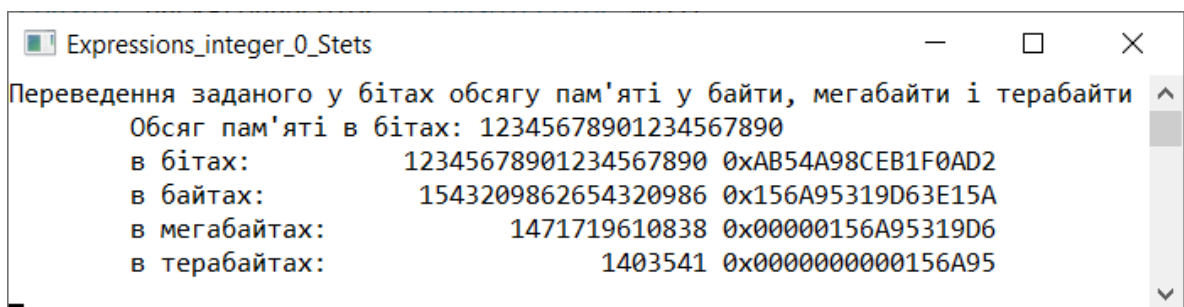


Рисунок 3.1 – Зразок результатів виконання поточного завдання

Завдання 4. Створення проекту для виконання логічних операцій

- 1) Додайте у проектне рішення новий проект консольного застосування з назвою згідно формату:

`Expressions_logical_Номер_варіанту_Прізвище_виконавця`

- 2) Призначте новий проект стартовим проектом і протестуйте роботу програми.
- 3) Текст умови індивідуального варіанту задачі на виконання логічних операцій використовуйте у якості коментаря до методу Main класу Program створеного проекту, наприклад:

```
// Дано шестизначне число N - номер квитка.
// Перевірити істинність висловлювання: «Квиток є щасливим»
// -----
// Квиток вважати щасливим, якщо сума перших трьох цифр співпадає
// із сумою останніх трьох цифр або сума цифр у парних позиціях
// співпадає із сумою цифр у непарних позиціях
```


- 4) Реалізуйте програмне і, за потреби, ручне налаштування параметрів консолі з метою відображення вікна з назвою проекту у рядку заголовку вікна та тексту завдання у робочій області вікна (рисунок 4.1). Протестуйте роботу програми.
- 5) Проаналізуйте умову індивідуального варіанту задачі на виконання логічних операцій, сформулюйте тестовий набір значень, що буде використовуватись у якості вхідних даних програми з метою отримання логічних значень true та false у якості результату обчислення кожної із простих складових логічного виразу, наприклад:
- 129345 – сума перших трьох цифр співпадає із сумою останніх трьох цифр;
 - 132495 – сума цифр у парних позиціях співпадає із сумою цифр у непарних позиціях;
 - 123456 – жодна з умов, за яких квиток вважається «щасливим», не виконується.
- 6) У методі Main перед оператором виклику методу ReadKey класу Console опишіть:
- a) оператори оголошення змінних, що призначені для збереження значень вхідних даних програми, та їх ініціалізації варіантами значень з тестового набору, залишаючи не закоментованими лише один варіант, наприклад:

```
int n = 129345;
//int n = 132495;
//int n = 123456;
```
 - b) оператор відображення значень змінних, що зберігають вхідні дані, наприклад:

```
Console.WriteLine("\tn = {0}", n);
```
 - c) оператори оголошення та ініціалізації змінних, що призначені для збереження числових значень, які обчислюються на основі вхідних даних з метою подальшого використання у логічному виразі, якщо такі передбачені умовою задачі, наприклад:

```
int d1, d2, d3, d4, d5, d6;
int t = n;
d1 = t % 10;
t /= 10;
d2 = t % 10;
t /= 10;
d3 = t % 10;
t /= 10;
d4 = t % 10;
t /= 10;
d5 = t % 10;
t /= 10;
d6 = t % 10;
```
 - d) оператор обчислення значення виразу згідно умови задачі і збереження результату у змінній логічного типу, наприклад:

```
bool truth = d1 + d2 + d3 == d4 + d5 + d6 || d1 + d3 + d5 == d2 + d4 + d6;
```
 - e) оператор відображення результату обчислення логічного виразу, наприклад:

```
Console.WriteLine("\tРезультат: {0}", truth);
```
- 7) Збережіть зміни програмного коду проекту. Протестуйте роботу програми для кожного з варіантів значень вхідних даних (переміщуючи символи коментаря) (рисунки 4.1 – 4.3). Результати тестування додайте у звіт.

- 8) Перед оператором відображення вхідних даних опишіть оператори, які реалізують діалог з користувачем з метою введення вхідних даних.
- 9) Збережіть зміни програмного коду проекту.
- 10) Сформууйте новий набір тестових даних, що буде використовуватись у якості вхідних даних програми з метою отримання логічних значень true та false у якості результату обчислення кожної із простих складових логічного виразу.
- 11) Протестуйте роботу програми шляхом введення на запит програми кожного з варіантів вхідних даних нового тестового набору. Результати тестування додайте у звіт.

Самоконтроль:

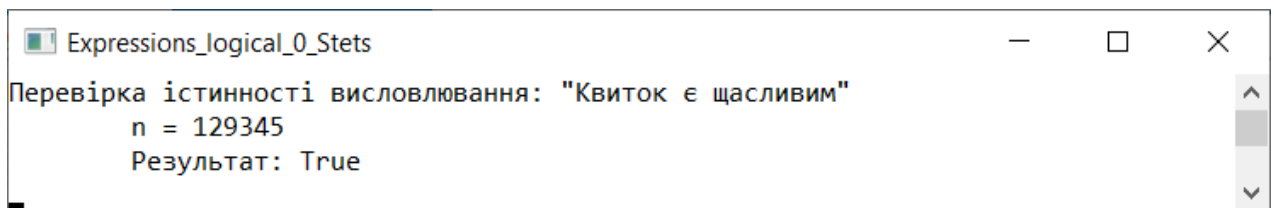


Рисунок 4.1 – Зразок результату для випадку, коли сума перших трьох цифр співпадає із сумою останніх трьох цифр

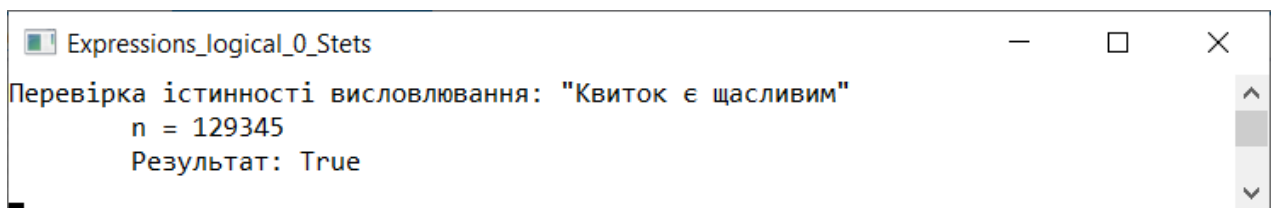


Рисунок 4.2 – Зразок результату для випадку, коли сума цифр в парних позиціях співпадає із сумою цифр в непарних позиціях

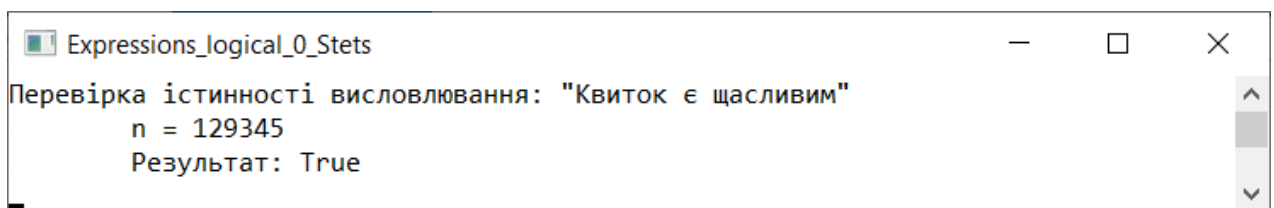


Рисунок 4.3 – Зразок результату для випадку, коли жодна з умов, за яких квиток вважається «щасливим», не виконується

Завдання 5. Створення проекту для виконання бітових операцій

- 1) Додайте у проектне рішення новий проект консольного застосування з назвою згідно формату:
`Expressions_bitwise_Номер_варіанту_Прізвище_виконавця`
- 2) Призначте новий проект стартовим проектом і протестуйте роботу програми.
- 3) Текст умови індивідуального варіанту задачі на виконання бітових операцій використайте у якості коментаря до методу Main класу Program створеного проекту, наприклад:
`// Дано число N цілого типу.`

// Переставити місцями другий і передостанній байти числа

- 4) Реалізуйте програмне і, за потреби, ручне налаштування параметрів консолі з метою відображення вікна з назвою проекту у рядку заголовку вікна та тексту завдання у робочій області вікна (рисунок 5.1). Протестуйте роботу програми.
- 5) Проаналізуйте умову індивідуального варіанту задачі на виконання бітових операцій, сформууйте тестовий набір значень, що буде використовуватись у якості вхідних даних програми з метою полегшення аналізу виконання операцій у шістнадцятковому та двійковому представленнях (рисунок 5.1).

- 6) У методі Main перед оператором виклику методу ReadKey класу Console опишіть:
 - a) оператори оголошення змінних, що призначені для збереження значень вхідних даних програми, та їх ініціалізації одним або, за потреби, декількома варіантами значень з тестового набору, залишаючи не закоментованими лише один варіант, наприклад:

```
uint n = 0x88dcfe11;
```

- b) оператор оголошення змінної format типу string та ініціалізації її посиланням на рядок, що визначає формат представлення вхідних даних, проміжних результатів обчислень та кінцевого результату у десятковому, шістнадцятковому та двійковому представленнях, наприклад:

```
string format = "\\t\\t {0,2}: {1:D12} 0x{1:X8} {2,32}";
```

- c) оператор оголошення змінної типу string, що призначена для збереження посилання на рядок з двійковим представленням одного з вхідних значень та її ініціалізації результатом виклику методу ToString класу Convert, другий аргумент якого визначає основу системи числення, наприклад:

```
string n_bin = Convert.ToString(n, 2);
```

- d) оператор виклику методу WriteLine класу Console з метою відображення імені змінної, що зберігає вхідне значення, та саме значення в різних системах числення у відповідності з форматом, що визначається рядком, на який посилається змінна format, наприклад:

```
Console.WriteLine(format, "n", n, n_bin);
```

- 7) Збережіть зміни програмного коду проекту. Протестуйте роботу програми.

- 8) Опишіть оператори які забезпечують:

- a) обчислення проміжних результатів обчислень та кінцевого результату та збереження їх у окремих змінних;
 - b) відображення текстів операторів, що реалізують обчислення;
 - c) створення рядків з двійковим представленням результатів обчислень;
 - d) відображення імен та значень змінних в процесі виконання обчислень у відповідності з форматом, що визначається рядком, на який посилається змінна format.

Приклад виконання проміжних обчислень та відображення їх результатів:

```
uint t1 = n & 0x00ff0000;  
Console.WriteLine("uint t1 = n & 0x00ff0000;");  
string t1_bin = Convert.ToString(t1, 2);  
Console.WriteLine(format, "t1", t1, t1_bin);
```

Приклад отримання та відображення кінцевого результату:

```
uint n2 = t1 | t2 | t3;  
Console.WriteLine("uint n2 = t1 | t2 | t3;");  
string n2_bin = Convert.ToString(n2, 2);
```

```
Console.WriteLine(format, "n2", n2, n2_bin);
```

- 9) Збережіть зміни програмного коду проекту. Протестуйте роботу програми для кожного з варіантів вхідних даних (якщо їх більше одного) (рисунки 5.1).
- 10) Перед оператором відображення вхідних даних опишіть оператори, які реалізують діалог з користувачем з метою введення вхідних даних.
- 11) Збережіть зміни програмного коду проекту.
- 12) Сформулюйте новий набір тестових даних, що буде використовуватись у якості вхідних даних у процесі діалогу програми з користувачем.
- 13) Протестуйте роботу програми шляхом введення на запит програми кожного з варіантів вхідних даних нового тестового набору. Результати тестування додайте у звіт.

Самоконтроль:

```
Expressions_bitwise_0_Stets
Переставлення місцями другого і передостаннього байтів числа
n: 002296184337 0x88DCFE11 100010001101110011111111000010001
uint t1 = n & 0x00ff0000;
t1: 000014417920 0x00DC0000 110111000000000000000000
t1 = t1 >> 8;
t1: 000000056320 0x0000DC00 1101110000000000
uint t2 = n & 0x0000ff00;
t2: 000000065024 0x0000FE00 1111111000000000
t2 = t2 << 8;
t2: 000016646144 0x00FE0000 111111100000000000000000
uint t3 = n & 0xff0000ff;
t3: 002281701393 0x88000011 100010000000000000000000000010001
uint n2 = t1 | t2 | t3;
n2: 002298403857 0x88FEDC11 10001000111111101101110000010001
```

Рисунок 5.1 – Зразок результатів виконання програми

Домашнє завдання: Оформити звіт з аналізом створеного коду та представленням результатів виконання програми. Зробити висновки.

Контрольні питання

1. Як створити проектне рішення з проектом консольного застосування?
2. Як додати в рішення ще один проект консольного застосування?
3. Яким чином можна призначити проект стартовим?
4. Що таке простір імен?
5. Для чого призначене ключове слово using?
6. Засоби яких просторів імен були використані в процесі виконання запропонованих завдань?
7. Що таке клас?
8. Які елементи може містити клас? Яке їх призначення?
9. Що таке метод?
10. Яким чином здійснюється виклик методів класу?
11. В чому полягає особливість використання статичних елементів класу?
12. В чому полягає особливість використання статичних класів?
13. Яким чином здійснюється доступ до значень властивостей класу?

14. Що таке точка входу в додаток?
15. Що є точкою входу консольного застосування?
16. Засоби якого класу призначені для підтримки роботи з вікном консолі?
17. Як змінити текст рядка заголовку вікна консолі?
18. Як встановити потрібний колір фону консолі?
19. Як вказати колір переднього плану консолі?
20. Яку команду використовують для виведення даних на екран?
21. В чому відмінність між методами Write та WriteLine класу Console?
22. У яких випадках використовується метод Write класу Console?
23. Яким чином можна здійснити форматування тексту, що виводиться?
24. Як вказати ширину поля виведення значення і спосіб вирівнювання значення?
25. Що таке керуючі послідовності і з якою метою вони використовуються?
26. Що визначає властивість OutputEncoding класу Console?
27. Що визначає властивість InputEncoding класу Console?
28. Використання яких шрифтів дозволяє правильно реалізувати введення та виведення тексту українською мовою? Яке кодування потрібно при цьому задати?
29. Які засоби для роботи з вікном консолі надає вікно його властивостей?
30. Що таке тип даних? Що визначає тип даних?
31. Які ви знаєте базові типи даних?
32. Як оголосити змінну?
33. Що таке ініціалізація змінної?
34. Яким чином можна описати оператор введення даних з клавіатури?
35. Як перетворити рядкове значення до значень базових типів даних?
36. Як отримати текстове представлення значень базових типів даних?
37. Як отримати шістнадцяткове представлення значень цілочислових типів даних?
38. Як отримати двійкове представлення значень цілочислових типів даних?
39. Яким чином можна проаналізувати значення елементів програми під час її виконання?
40. Який клас містить константи і методи для виконання математичних обчислень?