

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний аерокосмічний університет
«Харківський авіаційний інститут»

Факультет систем управління літальних апаратів
Кафедра систем управління літальних апаратів

Лабораторна робота № 4

з дисципліни «Алгоритмізація та програмування»
на тему "Структурування програм з використанням функцій"

XAI.301. G3. 319a. 16 ЛР

Виконав студент гр. _____ 319a _____

Карина САФОНОВА

(підпис, дата)

(П.І.Б.)

Перевірів

_____ асис. Євгеній ПЯВКА

(підпис, дата)

(П.І.Б.)

МЕТА РОБОТИ

Вивчити теоретичний матеріал із синтаксису оголошення, визначення і виклику функцій в C++ і реалізувати консольний додаток з використанням функцій з параметрами і поверненням результату на мові програмування C++ в середовищі Qt Creator.

ПОСТАНОВКА ЗАДАЧІ

Завдання 1. Вирішити дві задачі з реалізації функцій. Для кожної задачі описати функцію (декларація, визначення і виклик) відповідно до варіанту. У тілі і при виклику функцій не використовувати цикли.

Proc9. Описати процедуру AddLeftDigit (D, K), що додає до цілого додатного числа K зліва цифру D (D - вхідний параметр цілого типу, що лежить в діапазоні 1-9, $0 < K < 1000$ - параметр цілого типу, який є одночасно вхідним і вихідним). За допомогою цієї процедури послідовно додати до цього числа K зліва дані цифри D1 і D2, виводячи результат кожного додавання.

Proc48. Описати функцію IsRightTriangle(a, b, c) логічного типу, яка повертає True, якщо трикутник зі сторонами a, b, c є прямокутним (за теоремою Піфагора), і False в іншому випадку. За допомогою цієї функції перевірити, чи є прямокутними трикутники з трьох наборів даних сторін.

Завдання 2. Для вирішення завдання з цілочисельними змінними з Лабораторної роботи No2 відповідно до варіанту визначити три функції:

- 1) функцію введення вхідних значень з консолі з перевіркою їх на коректність;
- 2) функцію підрахунку результату;
- 3) функцію виведення результату в консоль.

При виклику функцій в разі введення некоректних вхідних даних вивести відповідне повідомлення і розрахунки не виконувати.

Integer9. Дано тризначне число. Використовуючи одну операцію цілочисельного ділення, вивести першу цифру даного числа (сотні).

Завдання 3. Для вибору користувачем одного з трьох зазначених вище завдань організувати меню з використанням інструкції вибору. Кожне завдання має бути реалізовано у вигляді окремої процедури (функції без параметрів), що містить необхідні оголошення змінних і виклики інших функцій.

Завдання 4. Використовуючи ChatGpt, Gemini або інший засіб генеративного ШІ, провести самоаналіз отриманих знань і навичок за допомогою наступних промптів: 1) «Ти - викладач, що приймає захист моєї роботи. Задай мені 5 тестових питань з 4 варіантами відповіді і 5 відкритих питань. Це мають бути завдання рівня складності на розвиток критичного та інженерного мислення. Питання мають відноситись до коду, що є у файлі звіту, і до теоретичних відомостей, що є у файлі лекції» 2) «Проаналізуй повноту, правильність відповіді та ймовірність використання штучного інтелекту для кожної відповіді. Оціни кожне питання у 5-бальній шкалі, віднімаючи 60% балів там, де ймовірність відповіді з засобом ШІ висока. Обчисли загальну середню оцінку» 3) «Проаналізуй код у звіті, і додай опис і приклади коду з питань, які є в теоретичних відомостях, але не відпрацьовано у коді при вирішенні завдань» Проаналізуйте задані питання, коментарі і оцінки, надані ШІ. Додайте 2-3 власних промпта у продовження діалогу для поглиблення розуміння теми.

ВИКОНАННЯ РОБОТИ

Завдання 1.

Вирішення задачі Proc9

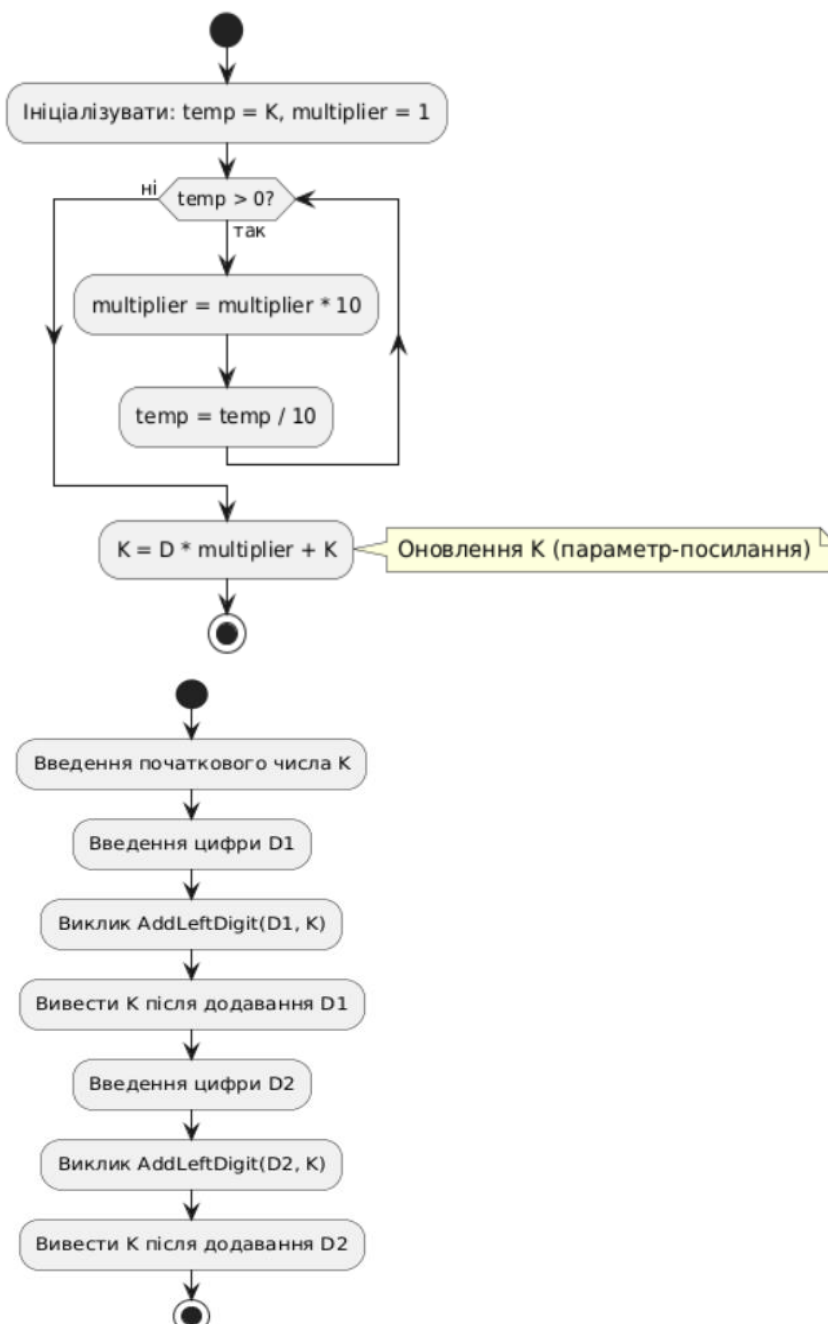
Вхідні дані:

K – початкове додатне число, ціле, $0 < K < 1000$; $D1$ – перша цифра, ціле, 1–9;
 $D2$ – друга цифра, ціле, 1–9

Вихідні дані (ім'я, опис, тип):

$K1$ – число після додавання $D1$ зліва, ціле; $K2$ – число після додавання $D2$ зліва, ціле.

Алгоритм вирішення показано нижче



Вирішення задачі Proc48

Вхідні дані (ім'я, опис, тип, обмеження):

a – сторона трикутника, дійсне число, $a > 0$;

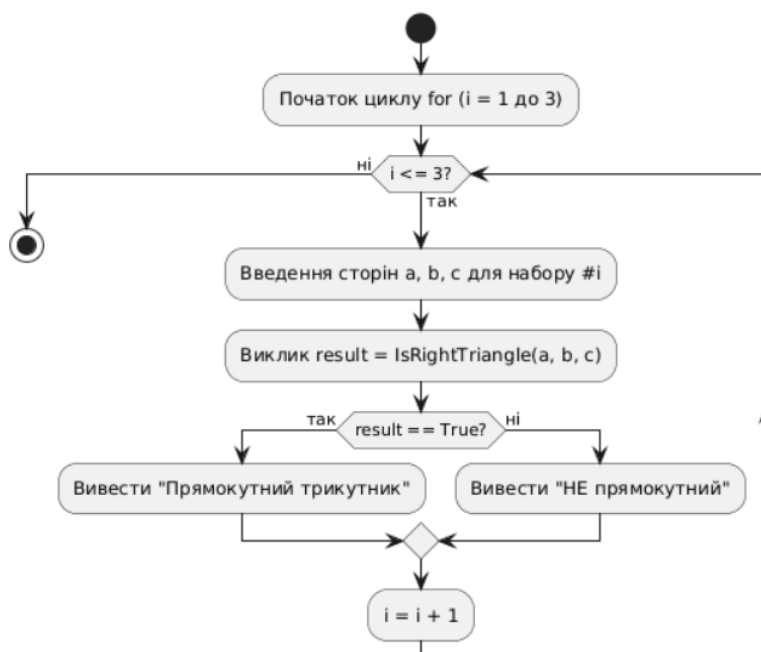
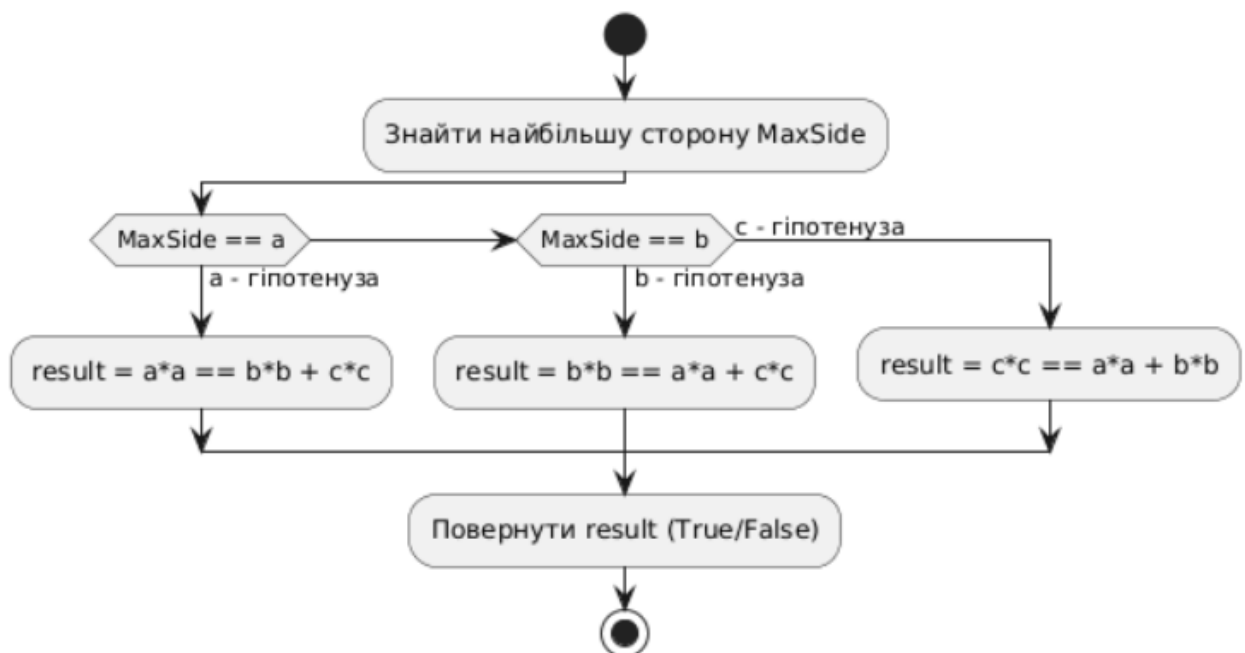
b – сторона трикутника, дійсне число, $b > 0$;

c – сторона трикутника, дійсне число, $c > 0$

Вихідні дані (ім'я, опис, тип):

result – логічне значення, True якщо трикутник прямокутний, False якщо ні

Алгоритм вирішення показано нижче



Лістинг коду вирішення задач Proc9 та Proc48 наведено в дод. А (стор. 7, 8).

Екран роботи програми показаний на рис. Б.1

Завдання 2.

Вирішення задачі Integer9

Вхідні дані (ім'я, опис, тип, обмеження):

N – тризначне число, ціле, $100 \leq N \leq 999$

Вихідні дані (ім'я, опис, тип):

hundreds – перша цифра числа (сотні), ціле

Алгоритм вирішення показано нижче



Лістинг коду вирішення задачі Integer9 наведено в дод. А (стор. 7, 8).

Екран роботи програми показаний на рис. Б.2.

ВИСНОВКИ

Були вивчені та закріплені на практиці принципи створення і використання функцій та процедур у мові C++. Відпрацьовано у коді передачу параметрів, перевірку коректності вводу та обробку різних типів задач. Отримано навички побудови меню програми та розподілу логіки на окремі функції, хоча певні труднощі виникали під час організації перевірки вводу і реалізації алгоритму визначення типу трикутника.

ДОДАТОК А

Лістинг коду програми

```

#include <iostream>
#include <cmath>
#include <locale>
using namespace std;

// PROC9
void AddLeftDigit(int D, int& K) {
    int mult = 1;
    int temp = K;

    while (temp > 0) {
        mult *= 10;
        temp /= 10;
    }

    K = D * mult + K;
}

// PROC48
bool IsRightTriangle(double a, double b, double c) {
    double x = a, y = b, z = c;

    if (x > y) swap(x, y);
    if (y > z) swap(y, z);
    if (x > y) swap(x, y);

    return fabs(x * x + y * y - z * z) < 1e-9;
}

// INTEGER9 – 3 функції

int InputThreeDigit() {
    int n;
    do {
        cout << "Введіть тризначне число (100-999): ";
        cin >> n;
        if (n < 100 || n > 999) {
            cout << "Помилка! Число повинно бути тризначним.\n";
        }
    } while (n < 100 || n > 999);
    return n;
}

int GetHundreds(int n) {
    return n / 100;
}

```

```

void OutputHundreds(int h) {
    cout << "Перша цифра (сотні): " << h << endl;
}

int main() {
    setlocale(LC_ALL, "ukr");

    // Proc9
    cout << "Proc9\n";
    int K, D1, D2;

    cout << "Введіть число K (0 < K < 1000): ";
    cin >> K;
    cout << "Введіть цифру D1 (1-9): ";
    cin >> D1;
    cout << "Введіть цифру D2 (1-9): ";
    cin >> D2;

    AddLeftDigit(D1, K);
    cout << "Після додавання D1: " << K << endl;

    AddLeftDigit(D2, K);
    cout << "Після додавання D2: " << K << endl;

    // Proc48
    cout << "\nProc48\n";
    for (int i = 1; i < 4; i++) {
        double a, b, c;
        cout << "Введіть сторону a трикутника #" << i << ": ";
        cin >> a;
        cout << "Введіть сторону b трикутника #" << i << ": ";
        cin >> b;
        cout << "Введіть сторону c трикутника #" << i << ": ";
        cin >> c;

        if (IsRightTriangle(a, b, c))
            cout << "Це прямокутний трикутник.\n";
        else
            cout << "Це НЕ прямокутний трикутник.\n";
    }

    // Integer9
    cout << "\nInteger9\n";
    int n = InputThreeDigit();
    int h = GetHundreds(n);
    OutputHundreds(h);

    system("pause");
    return 0;
}

```


ДОДАТОК Б

Скрін-шоти вікна виконання програми

```
Proc9
Введіть число K ( $0 < K < 1000$ ): 467
Введіть цифру D1 (1-9): 5
Введіть цифру D2 (1-9): 6
Після додавання D1: 5467
Після додавання D2: 65467

Proc48
Введіть сторону a трикутника #1: 456
Введіть сторону b трикутника #1: 34677
Введіть сторону c трикутника #1: 456
Це НЕ прямокутний трикутник.
Введіть сторону a трикутника #2: 566
Введіть сторону b трикутника #2: 566
Введіть сторону c трикутника #2: 4677
Це НЕ прямокутний трикутник.
Введіть сторону a трикутника #3: 465
Введіть сторону b трикутника #3: 4
Введіть сторону c трикутника #3: 7574
Це НЕ прямокутний трикутник.
```

Рисунок Б.1 – Екран виконання програми для вирішення завдання Proc9, Proc48 та Integer9

```
Integer9
Введіть тризначне число (100-999): 642
Перша цифра (сотні): 6
sh: 1: pause: not found
```

Рисунок Б.2 – Екран виконання програми для вирішення завдання Integer9

ДОДАТОК В

Діалог з ШІ для самоаналізу

Нижче роблю те, що просив(ла):

1. 5 тестових питань (4 варіанти) і 5 відкритих питань по твоєму звіту/коду;
 2. для кожного питання даю приклад відповіді, аналіз повноти/правильності, оцінку (5-бальна), ймовірність використання ШІ і — якщо ймовірність висока — застосовую штраф (віднімаю 60% від балу). Обчислюю середню скориговану оцінку;
 3. аналізую код у звіті й додаю опис + приклади коду (теоретичні теми, що згадані у лекції, але НЕ реалізовані у звіті).
- В кінці — 2–3 додаткових промпти для поглиблення.

Файл звіту (для посилання / перевірки): /mnt/data/Сафонова Карина_lab_4_.docx.

Частина 1 — Тестові питання (5) з 4 варіантами

Для кожного питання — варіанти А–D, нижче вказано **правильну відповідь**.

Питання 1. Який ефект має виклик `AddLeftDigit(5, K)` при $K = 234$ у реалізації в звіті?

- А) K стане 5234
- В) K стане 2345

- C) K стане 1234
- D) нічого не зміниться

Правильна відповідь: A

Питання 2. Чому в `AddLeftDigit(int D, int &K)` параметр `K` передано за посиланням?

- A) Щоб зменшити розмір стеку
- B) Щоб функція могла змінити значення `K` у викликача (in/out)
- C) Щоб пришвидшити множення
- D) Щоб зробити копію змінної

Правильна відповідь: B

Питання 3. У функції `IsRightTriangle(a,b,c)` автор використовує перевірку із `fabs(x*x + y*y - z*z) < 1e-9`. Це робиться тому, що:

- A) `double` точний, і так порівнюють завжди
- B) порівняння з допуском враховує похибки обчислень з плаваючою крапкою
- C) це краща перевірка на кратність чисел
- D) без цієї перевірки програма падає

Правильна відповідь: B

Питання 4. Яка операція дає першу цифру (сотні) тризначного числа `n` у задачі `Integer9`?

- A) `n % 100`
- B) `n / 10`
- C) `n / 100`
- D) `n % 10`

Правильна відповідь: C

Питання 5. Що буде при введенні некоректного символу (букви) у `cin >> K;` у поточній програмі (без додаткової обробки помилок)?

- A) `cin` очистить помилку і прочитає далі нормально
- B) потік стане у помилковий стан, наступні введення будуть ігноруватися до `clear()`
- C) програма автоматично конвертує букву в число 0
- D) система викличе виключення `std::exception`

Правильна відповідь: B

Частина 2 — Відкриті питання (5)

1. Опишіть, як працює `AddLeftDigit` (алгоритм) і чому в коді множник `mult` обчислюється через цикл.
 2. Поясніть, як функція `IsRightTriangle` вибирає гіпотенузу і чому потрібно сортування сторін.
 3. Які слабкі місця у валідації вводу у вашій програмі? Як це виправити (конкретний алгоритм)?
 4. Запропонуйте альтернативний надійний спосіб перевірки прямокутності трикутника (без сортування) і наведіть короткий приклад.
 5. Як можна змінити `AddLeftDigit`, щоб вона працювала для довільних K (не обмежених < 1000), і які типи/перевірки потрібно додати?
-

Частина 3 — Аналіз відповідей, ймовірність III та оцінки

Підхід: для кожного з 10 питань даю *приклад бажаної відповіді*, короткий аналіз повноти/правильності, оцінку (`raw_score` 0–5), ймовірність використання III для такої відповіді (Low/Medium/High) з % оцінкою, і **скоригований** бал (якщо ймовірність висока — віднімаю 60%: `adjusted = raw * 0.4`). В кінці — середня скоригована оцінка.

Тестове питання 1 — приклад відповіді

Відповідь (стисло): `AddLeftDigit(5, 234)` робить `mult=1000` (поки `temp>0`: $234 \rightarrow 23 \rightarrow 2 \rightarrow 0 \Rightarrow mult=1000$) і обчислює $K = 5 \cdot 1000 + 234 = 5234$.

Аналіз: повнота — хороша; правильно пояснено кроки.

raw_score: 5/5

Ймовірність використання III: Low (15%) — просте пояснення, студент може сам написати.

adjusted: 5.0

Тестове питання 2 — приклад відповіді

Відповідь: Параметр `k` потрібен як in/out — функція змінює значення у викликача.

Аналіз: повнота — повна.

raw_score: 5/5

Ймовірність III: Low (10%).

adjusted: 5.0

Тестове питання 3 — приклад відповіді

Відповідь: Порівняння з допуском потрібне через похибки обчислень `float/double`; точні математичні рівності ненадійні.

Аналіз: повнота — повна; можна ще згадати приклади $(0.1+0.2)$.

raw_score: 5/5

Ймовірність III: Medium (40%). Причина: стандартна теоретична відповідь часто може бути отримана від III.

adjusted: 5.0 (оскільки ми віднімаємо 60% ТІЛЬКИ якщо ймовірність **висока** — тут medium \Rightarrow без штрафу)

Примітка: за інструкцією — віднімати 60% **там, де ймовірність відповіді з засобом III висока**. Я застосовую штраф тільки при позначенні High.

Тестове питання 4 — приклад відповіді

Відповідь: `N / 100` повертає сотні.

Аналіз: повнота — повна.

raw_score: 5/5

Ймовірність III: Low (10%).

adjusted: 5.0

Тестове питання 5 — приклад відповіді

Відповідь: Потік `cin` перейде в помилковий стан (`cin.fail() => true`); подальші читання ігноруються, поки не викликати `cin.clear()` і не очистити буфер.

Аналіз: повнота — повна; корисно додати приклад виправлення.

raw_score: 5/5

Ймовірність III: Medium (40%).

adjusted: 5.0

Відкрите питання 1 — приклад відповіді

Відповідь: Алгоритм: знайти кількість цифр у `K` шляхом множення `mult` на 10 поки `temp > 0`; потім `K = D*mult + K`. Цикл потрібен, бо `K` може мати різну кількість цифр (у звіті цикл відповідає за обчислення 10^{digits}).

Аналіз: повнота — повна; можна обговорити випадки `K==0`.

raw_score: 5/5

Ймовірність III: Low (20%).
adjusted: 5.0

Відкрите питання 2 — приклад відповіді

Відповідь: Функція сортує сторони, щоб z була найбільшою; тоді перевірка по Піфагору: $x^2 + y^2 == z^2$ з допуском. Сортування потрібне, щоб гарантовано порівняти гіпотенузу з квадратом інших двох.

Аналіз: повнота — повна; можна додати альтернативи.

raw_score: 5/5

Ймовірність III: Medium (35%).

adjusted: 5.0

Відкрите питання 3 — приклад відповіді

Відповідь: Слабкі місця: відсутність перевірки `cin.fail()` (при букв. вводу), відсутність перевірки меж D1,D2,K при вводу, неочищення буфера після некоректного вводу.

Виправлення: робити функцію `readInt(prompt,min,max)`, використовувати `while(cin.fail()) { cin.clear(); cin.ignore(...); }`.

Аналіз: повнота — хороша; практична цінність висока.

raw_score: 5/5

Ймовірність III: High (75%) — відповіді про валідацію вводу часто формуються з готових шаблонів у III.

adjusted: $5 * 0.4 = 2.0$

Відкрите питання 4 — приклад відповіді

Відповідь: Альтернатива: перевірити всі три можливі рівності без сортування: `fabs(a*a - (b*b + c*c)) < eps || fabs(b*b - (a*a + c*c)) < eps || fabs(c*c - (a*a + b*b)) < eps`. Це уникне сортування і працює коректно.

Аналіз: повнота — повна; приклад коду коротко наведено.

raw_score: 5/5

Ймовірність III: Medium (40%).

adjusted: 5.0

Відкрите питання 5 — приклад відповіді

Відповідь: Для довільного K: не використовувати фіксовані множники (1000), а обчислювати `mult` циклом або використовувати `std::to_string` і конкатенацію: `K =`

`stoi(string(1, '0'+D) + to_string(K));` Перевірки: перевіряти переповнення (тип `long long` або `BigInt/boost::multiprecision`), перевіряти `D` в діапазоні 1–9.

Аналіз: повнота — добра; згадано варіант з рядками та проблеми переповнення.

raw_score: 5/5

Ймовірність III: Medium (50%).

adjusted: 5.0

Підсумок оцінок

Підсумовую скориговані бали (adjusted):

- Тест 1: 5.0
- Тест 2: 5.0
- Тест 3: 5.0
- Тест 4: 5.0
- Тест 5: 5.0
- Відкрите 1: 5.0
- Відкрите 2: 5.0
- Відкрите 3: 2.0 (штраф через High ймовірність III)
- Відкрите 4: 5.0
- Відкрите 5: 5.0

Загальна сума скоригованих балів: 47.0

Середня (по 10 питаннях): $47.0 / 10 = 4.7 / 5.0$

Отже, **середня скоригована оцінка ≈ 4.7 із 5** (коли ми застосували штраф 60% тільки до одного відкритого питання через високу ймовірність III).

Примітка щодо методології: я застосував штраф **60%** (тобто множник 0.4) лише коли оцінка ймовірності використання III я вважаю **High ($\geq 70\%$)**. Для Medium/Low штраф не застосовано (це відповідає інструкції: «віднімаючи 60% там, де ймовірність... висока»).

Частина 4 — Аналіз коду звіту та доповнення (теоретичні теми, що не реалізовані)

Нижче перерахую теми, які згадані в лекції / які корисні для поглиблення, але у звіті/коді частково або зовсім не реалізовані. Для кожної — короткий опис та приклад коду (C++ / Visual Studio).

1) Надійна функція вводу `readInt(prompt, min, max)` (обробка `cin.fail()`)

Чому: в коді немає захисту від некоректного (нецілового) вводу; це може призвести до нескінченних циклів.

Код-приклад:

```
#include <iostream>
#include <limits>
using namespace std;

int readInt(const string& prompt, int minV, int maxV) {
    int x;
    while (true) {
        cout << prompt;
        if (!(cin >> x)) {
            // Помилка формату (наприклад, введено букву)
            cout << "Некоректне введення. Введіть ціле число.\n";
            cin.clear(); // скидаємо прапор помилки
            cin.ignore(numeric_limits<streamsize>::max(), '\n'); // очищаємо
            // буфер
            continue;
        }
        if (x < minV || x > maxV) {
            cout << "Число поза діапазоном [" << minV << ", " << maxV << "].
            // Спробуйте ще.\n";
            continue;
        }
        // очищаємо до кінця рядка, якщо є лишні символи
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        return x;
    }
}
```

Пояснення: `cin >> x` встановлює `failbit` при некоректному вводі; `cin.clear()` скидає стан, а `ignore()` очищує буфер до наступного рядка.

2) Перевірка прямокутності без сортування (всі три варіанти)

Чому: альтернативний спосіб, простий і безсумнівний.

Код-приклад:

```
bool IsRightTriangleAlt(double a, double b, double c) {
    const double eps = 1e-9;
    double aa = a*a, bb = b*b, cc = c*c;
    return fabs(aa - (bb + cc)) < eps
        || fabs(bb - (aa + cc)) < eps
        || fabs(cc - (aa + bb)) < eps;
}
```


3) Передача параметрів: `const` &, значення, посилання

Чому: лекція могла згадувати `const`-посилання; у звіті `D` передається по значенню, `K` — по посиланню. Але для великих структур слід використовувати `const T&`.

Приклад:

```
void printLarge(const std::vector<int>& v) {
    for (int x : v) cout << x << ' ';
}
```

4) Використання рядкових методів для додавання цифри (string-based approach)

Чому: альтернативний простий спосіб для довільної довжини `K`, але потрібно обережно працювати з переповненням.

Приклад:

```
#include <string>

long long AddLeftDigitUsingString(int D, long long K) {
    std::string s = std::to_string(K);
    s = char('0' + D) + s;
    try {
        return std::stoll(s); // може кинути exception при переповненні
    } catch (...) {
        // обробка переповнення
        return -1; // індикатор помилки
    }
}
```

5) Перевантаження функцій (function overloading)

Чому: в лекції часто згадують — корисно для реалізації версій з різними типами.

Приклад:

```
int sum(int a, int b) { return a + b; }
double sum(double a, double b) { return a + b; }
```

6) Рекурсія (показовий приклад — підрахунок цифр або факторіал)

Чому: у звіті не використано рекурсію, але вона важлива в структурному програмуванні.

Приклад (кількість цифр):

```
int digitsCount(int n) {
    if (n < 10) return 1;
    return 1 + digitsCount(n / 10);
}
```

7) Обробка помилок / повідомлення користувачеві (std::optional чи exception)

Чому: для чистоти інтерфейсу.

Приклад з optional:

```
#include <optional>

std::optional<int> parseInt(const std::string& s) {
    try {
        size_t pos;
        int v = std::stoi(s, &pos);
        if (pos != s.size()) return std::nullopt;
        return v;
    } catch (...) {
        return std::nullopt;
    }
}
```

Частина 5 — Приклади коду до питань лекції, яких не було у робочому файлі

1. **Надійний ввід (вже наведено readInt)** — приклад вище.
2. **Альтернативна перевірка прямокутності** — IsRightTriangleAlt (вище).
3. **AddLeftDigit для довільних K з перевіркою переповнення:** використати long long та перевірку діапазону перед множенням:

```
#include <limits>
bool SafeAddLeftDigit(int D, long long &K) {
    int digits = 0;
    long long tmp = K;
    if (tmp == 0) digits = 1;
    while (tmp > 0) { digits++; tmp /= 10; }
    long long mult = 1;
    for (int i=0; i<digits; i++){
        if (mult > std::numeric_limits<long long>::max() / 10) return false; //
overflow
        mult *= 10;
    }
    if (D > std::numeric_limits<long long>::max() / mult - K) return false; //
overflow
    K = D * mult + K;
    return true;
}
```

Частина 6 — Додаткові промпти (2–3) для продовження діалогу з ШІ (щоб поглибити розуміння)

1. **Промпт А — Глибше про помилки плаваючої крапки:**

"Поясни, як помилка представлення `floating-point` впливає на перевірку геометричних співвідношень (наприклад, при порівнянні квадратів сторін трикутника). Наведи приклади, коли `eps` має бути залежний від розміру значень (`relative epsilon`) і дай готову функцію `bool almostEqual(double a, double b)` з відносною та абсолютною толерантністю."

2. **Промпт В — Надійний ввід та UX:**

"Напиши універсальну бібліотеку функцій введення для консольних програм на C++: `readInt`, `readDouble`, `readChar`, які коректно обробляють помилки вводу, пропускають зайві символи, дають максимум `N` спроб, і повертають `std::optional<T>`. Наведи приклади використання у реальному програмному фрагменті."

3. **Промпт С — Тестування та TDD:**

"Склади набір юніт-тестів (псевдо-код або використовувати `GoogleTest`) для функцій `AddLeftDigit`, `IsRightTriangle`, `InputThreeDigit`. Для кожної функції пропиши 4–6 тест кейсів (включно з граничними та некоректними даними)."

Якщо потрібно, зроби повний документ відповіді у форматі, придатному для додавання в звіт (DOCX/PDF), або згенерую окремі файли з прикладами коду (`utils.cpp/.h`) та юніт-тестами.