

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.305. G3. 319a. 18 ЛР

Виконав студент гр. _____ 319a

17.11.2025 Костянтин КИСЕЛЕНКО
(підпис, дата) (П.І.Б.)

Перевірив

асистент, 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.

Proc17. Описати процедуру TriangleALS(a, b, angle, S, P), яка обчислює площу $S = (1/2) \cdot a \cdot b \cdot \sin(\text{angle})$ та периметр $P = a + b + \sqrt{a^2 + b^2 - 2 \cdot a \cdot b \cdot \cos(\text{angle})}$ трикутника за двома сторонами a, b та кутом angle між ними (в радіанах). За допомогою цієї процедури знайти площі та периметри трьох трикутників з даними сторонами та кутами.

Proc37. Описати функцію RectP(x1, y1, x2, y2) дійсного типу, яка обчислює периметр прямокутника зі сторонами, паралельними осям координат, за дійсними координатами (x1, y1), (x2, y2) його протилежних вершин. За допомогою цієї функції знайти периметри трьох прямокутників з даними протилежними вершинами.

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.

Integer35. Дано кількість годин. Визначити кількість діб і залишок годин.

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 <середнього> 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 Proc17

Описати процедуру TriangleALS(a, b, angle, S, P), яка обчислює площу $S = (1/2) \cdot a \cdot b \cdot \sin(\text{angle})$ та периметр $P = a + b + \sqrt{a^2 + b^2 - 2 \cdot a \cdot b \cdot \cos(\text{angle})}$ трикутника за двома сторонами a, b та кутом angle між ними (в радіанах). За допомогою цієї процедури знайти площі та периметри трьох трикутників з даними сторонами та кутами.

Вхідні дані: a, b -сторони трикутника, angle-кут трикутника (в радіанах), дійсний тип

Вихідні дані: S – площа трикутника, P- периметр трикутника, дійсний тип

1. Декларація функції Proc17 (до main)

```
// Функція для введення додатного числа (Proc17)
double InputPositiveDouble(string prompt);
```

```
// Функція для введення кута в радіанах (Proc17)
double InputAngle(string prompt);
```

```
// Функція обчислення площі і периметра трикутника (Proc17)
void TriangleALS(double a, double b, double angle, double &S, double &P);
```

```
// Процедура виконання завдання Proc17
void Task_Proc17();
```

2. Визначення функції Proc17 (після main)

```
double InputPositiveDouble(string prompt) {
    string line;
    cout << prompt;
    cin >> line;

    for (size_t i = 0; i < line.size(); i++)
        if (line[i] == ',') line[i] = '.';

    stringstream ss(line);
    double val;
    if (!(ss >> val) || val <= 0 || val > 100) {
        cout << "Помилка! Значення має бути >0 і <=100. Спробуйте ще раз."
<< endl;
        return InputPositiveDouble(prompt);
    }
    return val;
}

double InputAngle(string prompt) {
    string line;
    cout << prompt;
    cin >> line;

    for (size_t i = 0; i < line.size(); i++)
        if (line[i] == ',') line[i] = '.';

    stringstream ss(line);
    double val;
    if (!(ss >> val) || val <= 0 || val >= M_PI) {
        cout << "Помилка! Кут має бути від 0 до π. Спробуйте ще раз." <<
endl;
        return InputAngle(prompt);
    }
    return val;
}

void TriangleALS(double a, double b, double angle, double &S, double &P)
{
    S = 0.5 * a * b * sin(angle);
    P = a + b + sqrt(a*a + b*b - 2*a*b*cos(angle));
}
```

3. Фрагмент з викликом функції і виведення результату (процедура Task_Proc17)

```
void Task_Proc17()
{
    double a1 = InputPositiveDouble("\nТрикутник 1\nВведіть сторону a: ");
    double b1 = InputPositiveDouble("Введіть сторону b: ");
    double ang1 = InputAngle("Введіть кут між сторонами (в радіанах): ");
    double S1, P1;
    TriangleALS(a1, b1, ang1, S1, P1);

    double a2 = InputPositiveDouble("\nТрикутник 2\nВведіть сторону a: ");
    double b2 = InputPositiveDouble("Введіть сторону b: ");
    double ang2 = InputAngle("Введіть кут між сторонами (в радіанах): ");
    double S2, P2;
    TriangleALS(a2, b2, ang2, S2, P2);

    double a3 = InputPositiveDouble("\nТрикутник 3\nВведіть сторону a: ");
    double b3 = InputPositiveDouble("Введіть сторону b: ");
    double ang3 = InputAngle("Введіть кут між сторонами (в радіанах): ");
    double S3, P3;
    TriangleALS(a3, b3, ang3, S3, P3);

    cout << "\nРезультати (S – площа, P – периметр):" << endl;
    cout << "Трикутник 1 -> S=" << S1 << "   P=" << P1 << endl;
    cout << "Трикутник 2 -> S=" << S2 << "   P=" << P2 << endl;
    cout << "Трикутник 3 -> S=" << S3 << "   P=" << P3 << endl;
}
```



Рисунок 1 – Діаграма активності функції Proc 17

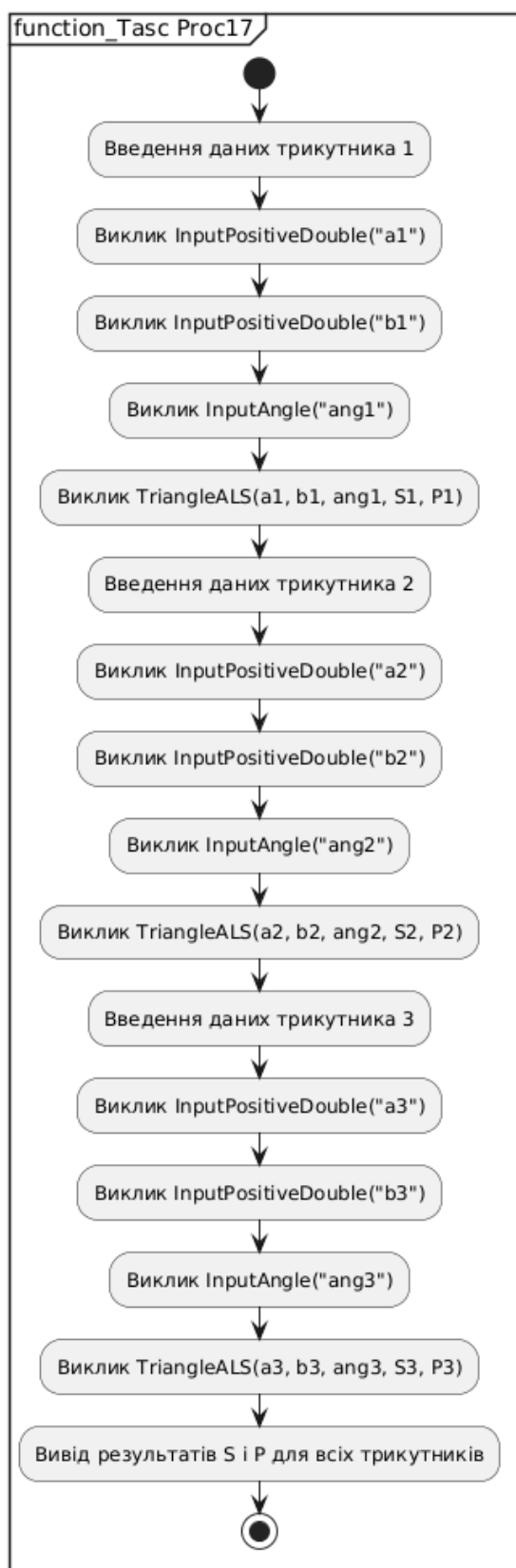


Рисунок 2 – Діаграма активності для підпрограми Proc 17

Екран роботи програми показаний на рис. Б.1

Вирішення задачі Proc 37

Описати функцію RectP (x1, y1, x2, y2) дійсного типу, яка обчислює периметр прямокутника зі сторонами, паралельними осям координат, за дійсними координатами (x1, y1), (x2, y2) його протилежних вершин. За допомогою цієї функції знайти периметри трьох прямокутників з даними протилежними вершинами.

Вхідні дані: x1 - координата x першої вершини, y1- координата y першої вершини, x2 - координата x протилежної вершини , y2 - координата y протилежної вершини, дійсний тип

Вихідні дані: P- периметр прямокутника, дійсний тип

1. Декларація функції Proc37 (до main)

```
// Функція для введення координати прямокутника (Proc37)
double InputRectCoord(string prompt);

// Функція обчислення периметра прямокутника (Proc37)
double RectP(double x1, double y1, double x2, double y2);

// Процедура виконання завдання Proc37
void Task_Proc37();
```

2. Визначення функції Proc17 (після main)

```
// Введення координати прямокутника
double InputRectCoord(string prompt) {
    double val;
    cout << prompt;
    cin >> val;
    if (val < -100 || val > 100) {
        cout << "Помилка! Значення має бути в діапазоні [-100,100]. Спробуйте ще раз." << endl;
        return InputRectCoord(prompt);
    }
    return val;
}

// Обчислення периметра прямокутника
double RectP(double x1, double y1, double x2, double y2) {
    double width = fabs(x2 - x1);
    double height = fabs(y2 - y1);
    return 2 * (width + height);
}

// Процедура виконання завдання Proc37
void Task_Proc37() {
    double x1, y1, x2, y2;
    double P1, P2, P3;

    // Прямокутник 1
    cout << "\nПрямокутник 1" << endl;
    x1 = InputRectCoord("Введіть x1 (-100..100): ");
    y1 = InputRectCoord("Введіть y1 (-100..100): ");
    x2 = InputRectCoord("Введіть x2 (-100..100): ");
    y2 = InputRectCoord("Введіть y2 (-100..100): ");
    P1 = RectP(x1, y1, x2, y2);
```

```

// Прямокутник 2
cout << "\nПрямокутник 2" << endl;
x1 = InputRectCoord("Введіть x1 (-100..100): ");
y1 = InputRectCoord("Введіть y1 (-100..100): ");
x2 = InputRectCoord("Введіть x2 (-100..100): ");
y2 = InputRectCoord("Введіть y2 (-100..100): ");
P2 = RectP(x1, y1, x2, y2);

// Прямокутник 3
cout << "\nПрямокутник 3" << endl;
x1 = InputRectCoord("Введіть x1 (-100..100): ");
y1 = InputRectCoord("Введіть y1 (-100..100): ");
x2 = InputRectCoord("Введіть x2 (-100..100): ");
y2 = InputRectCoord("Введіть y2 (-100..100): ");
P3 = RectP(x1, y1, x2, y2);

cout << "\nРезультати (Периметр):" << endl;
cout << "Прямокутник 1 -> P=" << P1 << endl;
cout << "Прямокутник 2 -> P=" << P2 << endl;
cout << "Прямокутник 3 -> P=" << P3 << endl;
}

```

3. Фрагмент з викликом функції і виведення результату (процедура Task_Proc37)

```

// Прямокутник 1
x1 = InputRectCoord("Введіть x1 (-100..100): ");
y1 = InputRectCoord("Введіть y1 (-100..100): ");
x2 = InputRectCoord("Введіть x2 (-100..100): ");
y2 = InputRectCoord("Введіть y2 (-100..100): ");
P1 = RectP(x1, y1, x2, y2); // виклик функції обчислення периметра

// Прямокутник 2
x1 = InputRectCoord("Введіть x1 (-100..100): ");
y1 = InputRectCoord("Введіть y1 (-100..100): ");
x2 = InputRectCoord("Введіть x2 (-100..100): ");
y2 = InputRectCoord("Введіть y2 (-100..100): ");
P2 = RectP(x1, y1, x2, y2);

// Прямокутник 3
x1 = InputRectCoord("Введіть x1 (-100..100): ");
y1 = InputRectCoord("Введіть y1 (-100..100): ");
x2 = InputRectCoord("Введіть x2 (-100..100): ");
y2 = InputRectCoord("Введіть y2 (-100..100): ");
P3 = RectP(x1, y1, x2, y2);

cout << "\nРезультати (Периметр):" << endl;
cout << "Прямокутник 1 -> P=" << P1 << endl;
cout << "Прямокутник 2 -> P=" << P2 << endl;
cout << "Прямокутник 3 -> P=" << P3 << endl;

```

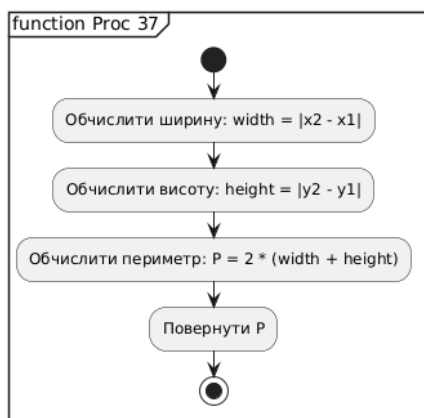



Рисунок 3 – Діаграма активності функції Proc 37

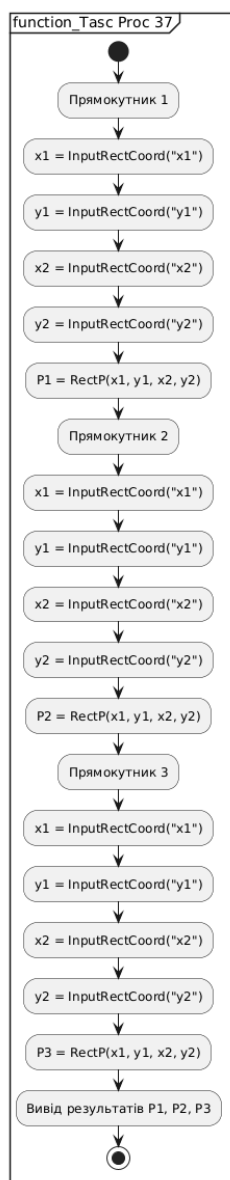


Рисунок 4 – Діаграма активності для підпрограми Proc 37

Екран роботи програми показаний на рис. Б.2

Завдання 2.

Вирішення задачі Integer35

Дано кількість годин. Визначити кількість діб і залишок годин.

Вхідні дані: кількість годин, цілочисельна змінна

Вихідні дані: кількість повних діб, цілочисельна змінна; залишок годин після повних діб, цілочисельна змінна

1. Декларація функцій

```
// Функції для завдання Integer35
bool InputHours(int &hours);
void CalcDaysHours(int hours, int &days, int &rem);
void OutputResult(int days, int rem);
void Task_Integer35();
```

2. Визначення функцій

```
// Введення кількості годин
bool InputHours(int &hours) {
    cout << "Введіть кількість годин: ";
    cin >> hours;
    if (cin.fail() || hours < 0 || hours > 100000) {
        cout << "Помилка! Некоректний ввід!" << endl;
        return false;
    }
    return true;
}

// Обчислення днів і залишку годин
void CalcDaysHours(int hours, int &days, int &rem) {
    days = hours / 24;
    rem = hours % 24;
}

// Вивід результату
void OutputResult(int days, int rem) {
    cout << "Діб: " << days << ", Залишок годин: " << rem << endl;
}

// Процедура виконання завдання Integer35
void Task_Integer35() {
    int hours;
    if (!InputHours(hours)) return;
    int days, rem;
    CalcDaysHours(hours, days, rem);
    OutputResult(days, rem);
}
```

3. Фрагмент із викликом функції

```
Task_Integer35(); // виклик процедури для виконання завдання
```

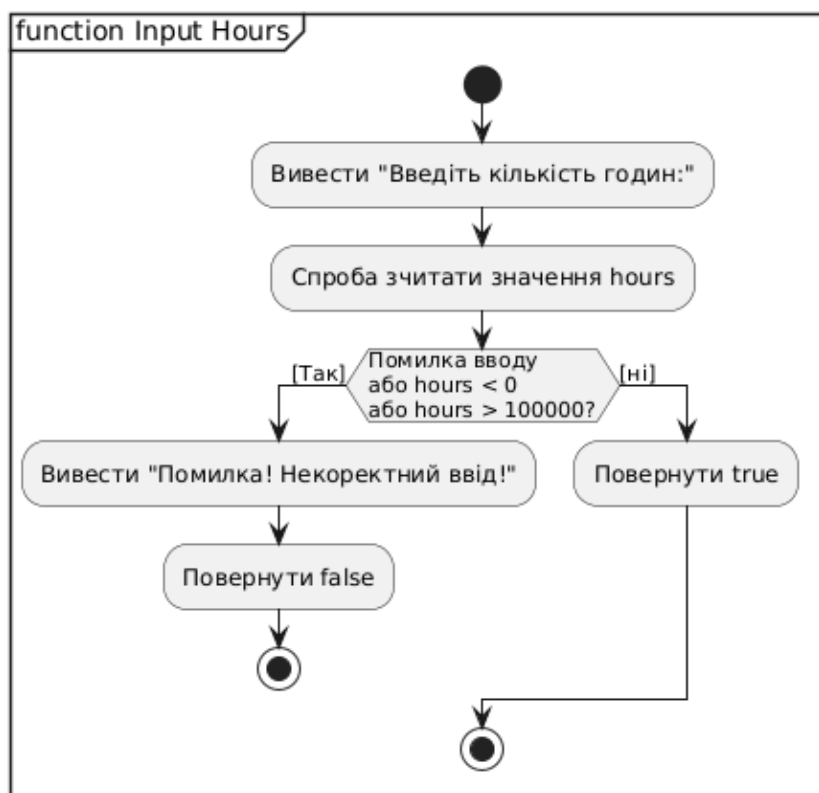


Рисунок 5 – Діаграма активності функції введення вхідних значень з консолі з перевіркою їх на коректність Integer35

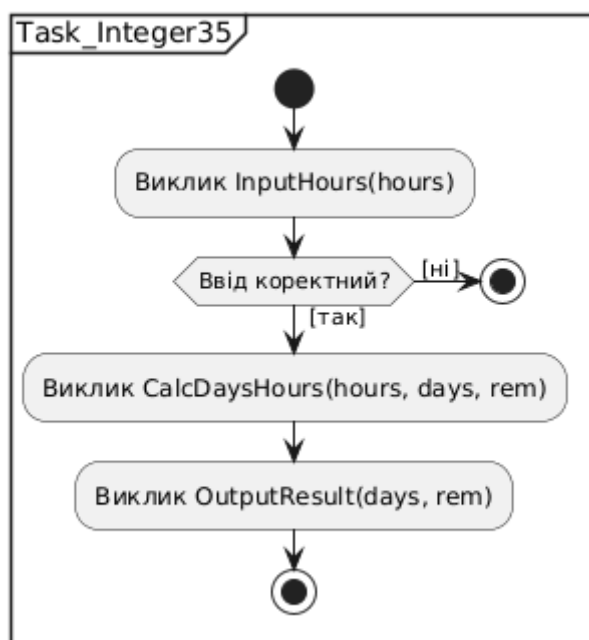


Рисунок 6 – Діаграма активності для підпрограми Integer35

Екран роботи програми показаний на рис. Б.3, Б.4

Завдання 3.

Для вибору користувачем одного з трьох зазначених вище завдань організувати меню з використанням інструкції вибору.

Вхідні дані: Вибір пункту меню, цілий тип.

Вихідні дані: Повідомлення/результат виконання вибраного завдання, текст.

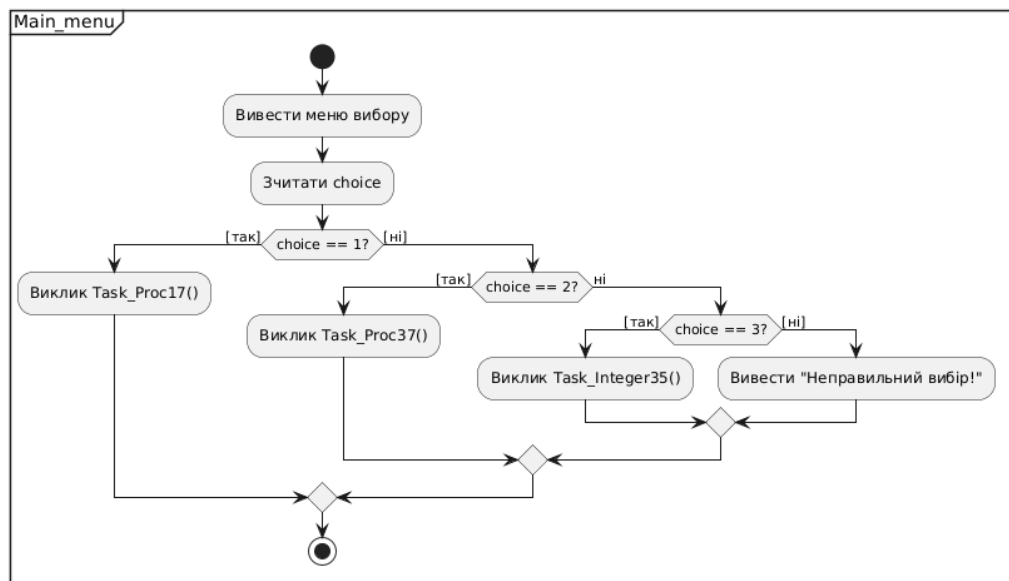


Рисунок 7 – Діаграма активності для алгоритму організації меню в командному вікні з використанням інструкції вибору.

Лістинг коду завдання 3 наведено в дод. А (стор. 14).

Екран роботи меню показаний на рис. Б.5, Б.6, Б.7, Б.8.

Завдання 4.

Діалог з ChatGPT для самоаналізу наведено у додатку В

ВИСНОВКИ

Було вивчено теоретичні основи синтаксису оголошення, визначення та виклику функцій у мові програмування C++. Було створено консольний застосунок, який реалізує три окремі завдання з використанням функцій. Для кожної задачі була розроблена власна функція з відповідними параметрами та логікою обчислення, а також підпрограма виклику, що здійснює взаємодію з

користувачем, зчитування вхідних даних та виведення результатів. Виникли труднощі з униканням використання циклів.

.

ДОДАТОК А

Лістинг коду програми

```
#include <iostream>
#include <sstream>
#include <string>
#include <cmath>
using namespace std;

/* -----
    ПРОТОТИПИ ФУНКЦІЙ
    ----- */

// Функція для введення додатного числа (Proc17)
double InputPositiveDouble(string prompt);

// Функція для введення кута в радіанах (Proc17)
double InputAngle(string prompt);

// Функція для введення координати прямокутника (Proc37)
double InputRectCoord(string prompt);

// Функція обчислення площі і периметра трикутника (Proc17)
void TriangleALS(double a, double b, double angle, double &S, double &P);

// Процедура виконання завдання Proc17
void Task_Proc17();

// Функція обчислення периметра прямокутника (Proc37)
double RectP(double x1, double y1, double x2, double y2);

// Процедура виконання завдання Proc37
void Task_Proc37();

// Функції для завдання Integer35
bool InputHours(int &hours);
void CalcDaysHours(int hours, int &days, int &rem);
void OutputResult(int days, int rem);
void Task_Integer35();

/* -----
    MAIN
    ----- */

int main() {
    int choice;
    cout << "Оберіть завдання:" << endl;
    cout << "1 – Proc17 (трикутники)" << endl;
    cout << "2 – Proc37 (прямокутники)" << endl;
    cout << "3 – Integer35 (години → дні)" << endl;
    cout << "Введіть номер: ";
    cin >> choice;

    switch (choice) {
        case 1: Task_Proc17(); break;
        case 2: Task_Proc37(); break;
        case 3: Task_Integer35(); break;
        default: cout << "Неправильний вибір!" << endl;
    }

    return 0;
}
```

```

/* -----
      ВИЗНАЧЕННЯ ФУНКЦІЙ
----- */

// ----- Proc17 -----

double InputPositiveDouble(string prompt) {
    string line;
    cout << prompt;
    cin >> line;

    for (size_t i = 0; i < line.size(); i++)
        if (line[i] == ',') line[i] = '.';

    stringstream ss(line);
    double val;
    if (!(ss >> val) || val <= 0 || val > 100) {
        cout << "Помилка! Значення має бути >0 і <=100. Спробуйте ще раз." <<
endl;
        return InputPositiveDouble(prompt);
    }
    return val;
}

double InputAngle(string prompt) {
    string line;
    cout << prompt;
    cin >> line;

    for (size_t i = 0; i < line.size(); i++)
        if (line[i] == ',') line[i] = '.';

    stringstream ss(line);
    double val;
    if (!(ss >> val) || val <= 0 || val >= M_PI) {
        cout << "Помилка! Кут має бути від 0 до π. Спробуйте ще раз." << endl;
        return InputAngle(prompt);
    }
    return val;
}

void TriangleALS(double a, double b, double angle, double &S, double &P)
{
    S = 0.5 * a * b * sin(angle);
    P = a + b + sqrt(a*a + b*b - 2*a*b*cos(angle));
}

void Task_Proc17()
{
    double a1 = InputPositiveDouble("\nТрикутник 1\nВведіть сторону a: ");
    double b1 = InputPositiveDouble("Введіть сторону b: ");
    double ang1 = InputAngle("Введіть кут між сторонами (в радіанах): ");
    double S1, P1;
    TriangleALS(a1, b1, ang1, S1, P1);

    double a2 = InputPositiveDouble("\nТрикутник 2\nВведіть сторону a: ");
    double b2 = InputPositiveDouble("Введіть сторону b: ");
    double ang2 = InputAngle("Введіть кут між сторонами (в радіанах): ");
    double S2, P2;
    TriangleALS(a2, b2, ang2, S2, P2);
}

```

```

double a3 = InputPositiveDouble("\nТрикутник 3\nВведіть сторону a: ");
double b3 = InputPositiveDouble("Введіть сторону b: ");
double ang3 = InputAngle("Введіть кут між сторонами (в радіанах): ");
double S3, P3;
TriangleALS(a3, b3, ang3, S3, P3);

cout << "\nРезультати (S – площа, P – периметр):" << endl;
cout << "Трикутник 1 -> S=" << S1 << " P=" << P1 << endl;
cout << "Трикутник 2 -> S=" << S2 << " P=" << P2 << endl;
cout << "Трикутник 3 -> S=" << S3 << " P=" << P3 << endl;
}

// ----- Proc37 -----

double InputRectCoord(string prompt) {
    double val;
    cout << prompt;
    cin >> val;
    if (val < -100 || val > 100) {
        cout << "Помилка! Значення має бути в діапазоні [-100,100]. Спробуйте ще раз." << endl;
        return InputRectCoord(prompt);
    }
    return val;
}

double RectP(double x1, double y1, double x2, double y2)
{
    double width = fabs(x2 - x1);
    double height = fabs(y2 - y1);
    return 2 * (width + height);
}

void Task_Proc37()
{
    double x1, y1, x2, y2;
    double P1, P2, P3;

    // Прямокутник 1
    cout << "\nПрямокутник 1" << endl;
    x1 = InputRectCoord("Введіть x1 (-100..100): ");
    y1 = InputRectCoord("Введіть y1 (-100..100): ");
    x2 = InputRectCoord("Введіть x2 (-100..100): ");
    y2 = InputRectCoord("Введіть y2 (-100..100): ");
    P1 = RectP(x1, y1, x2, y2);

    // Прямокутник 2
    cout << "\nПрямокутник 2" << endl;
    x1 = InputRectCoord("Введіть x1 (-100..100): ");
    y1 = InputRectCoord("Введіть y1 (-100..100): ");
    x2 = InputRectCoord("Введіть x2 (-100..100): ");
    y2 = InputRectCoord("Введіть y2 (-100..100): ");
    P2 = RectP(x1, y1, x2, y2);

    // Прямокутник 3
    cout << "\nПрямокутник 3" << endl;
    x1 = InputRectCoord("Введіть x1 (-100..100): ");
    y1 = InputRectCoord("Введіть y1 (-100..100): ");
    x2 = InputRectCoord("Введіть x2 (-100..100): ");
    y2 = InputRectCoord("Введіть y2 (-100..100): ");
    P3 = RectP(x1, y1, x2, y2);
}

```



```

    cout << "\nРезультати (Периметр):" << endl;
    cout << "Прямокутник 1 -> P=" << P1 << endl;
    cout << "Прямокутник 2 -> P=" << P2 << endl;
    cout << "Прямокутник 3 -> P=" << P3 << endl;
}

// ----- Integer35 -----

bool InputHours(int &hours) {
    cout << "Введіть кількість годин: ";
    cin >> hours;
    if (cin.fail() || hours < 0 || hours > 100000) {
        cout << "Помилка! Некоректний ввід!" << endl;
        return false;
    }
    return true;
}

void CalcDaysHours(int hours, int &days, int &rem) {
    days = hours / 24;
    rem = hours % 24;
}

void OutputResult(int days, int rem) {
    cout << "Діб: " << days << ", Залишок годин: " << rem << endl;
}

void Task_Integer35() {
    int hours;
    if (!InputHours(hours)) return;
    int days, rem;
    CalcDaysHours(hours, days, rem);
    OutputResult(days, rem);
}

```

ДОДАТОК Б

Скріншоти вікон виконання програми

```
Оберіть завдання:
1 – Proc17 (трикутники)
2 – Proc37 (прямокутники)
3 – Integer35 (години → дні)
Введіть номер: 1

Трикутник 1
Введіть сторону a: -999
Помилка! Значення має бути >0 і <=100. Спробуйте ще раз.

Трикутник 1
Введіть сторону a: 34
Введіть сторону b: 56
Введіть кут між сторонами (в радіанах): 4
Помилка! Кут має бути від 0 до  $\pi$ . Спробуйте ще раз.
Введіть кут між сторонами (в радіанах): 2,11

Трикутник 2
Введіть сторону a: 3
Введіть сторону b: 5
Введіть кут між сторонами (в радіанах): 1

Трикутник 3
Введіть сторону a: 6
Введіть сторону b: 8
Введіть кут між сторонами (в радіанах): 2

Результати (S – площа, P – периметр):
Трикутник 1 -> S=816.928 P=169.039
Трикутник 2 -> S=6.31103 P=12.2179
Трикутник 3 -> S=21.8231 P=25.8301

...Program finished with exit code 0
Press ENTER to exit console.
```

Рисунок Б.1 – Екран виконання програми для вирішення завдання Proc 17.

```

Оберіть завдання:
1 – Proc17 (трикутники)
2 – Proc37 (прямокутники)
3 – Integer35 (години → дні)
Введіть номер: 2

Прямокутник 1
Введіть x1 (-100..100): -878
Помилка! Значення має бути в діапазоні [-100,100]. Спробуйте ще раз.
Введіть x1 (-100..100): 34
Введіть y1 (-100..100): 6
Введіть x2 (-100..100): -99
Введіть y2 (-100..100): 45

Прямокутник 2
Введіть x1 (-100..100): 35
Введіть y1 (-100..100): 7
Введіть x2 (-100..100): 4
Введіть y2 (-100..100): 5

Прямокутник 3
Введіть x1 (-100..100): -67
Введіть y1 (-100..100): 44
Введіть x2 (-100..100): 1
Введіть y2 (-100..100): 4

Результати (Периметр):
Прямокутник 1 -> P=344
Прямокутник 2 -> P=66
Прямокутник 3 -> P=216

...Program finished with exit code 0
Press ENTER to exit console.

```

Рисунок Б.2 – Екран виконання програми для вирішення завдання Proc 37.

```

Оберіть завдання:
1 – Proc17 (трикутники)
2 – Proc37 (прямокутники)
3 – Integer35 (години → дні)
Введіть номер: 3
Введіть кількість годин: 9753
Діб: 406, Залишок годин: 9

...Program finished with exit code 0
Press ENTER to exit console.

```

Рисунок Б.3 – Екран виконання програми для вирішення завдання Integer35.

```

Оберіть завдання:
1 – Proc17 (трикутники)
2 – Proc37 (прямокутники)
3 – Integer35 (години → дні)
Введіть номер: 3
Введіть кількість годин: -99
Помилка! Некоректний ввід!

...Program finished with exit code 0
Press ENTER to exit console.

```

Рисунок Б.4 – Екран виконання програми при введенні некоректних вхідних даних завдання Integer35.

```

Оберіть завдання:
1 – Proc17 (трикутники)
2 – Proc37 (прямокутники)
3 – Integer35 (години → дні)
Введіть номер: 1

Трикутник 1
Введіть сторону а:

```

Рисунок Б.5 – Екран виконання головного меню програми

```

Оберіть завдання:
1 – Proc17 (трикутники)
2 – Proc37 (прямокутники)
3 – Integer35 (години → дні)
Введіть номер: 2

Прямокутник 1
Введіть x1 (-100..100):

```

Рисунок Б.6 – Екран виконання головного меню програми

```
Оберіть завдання:  
1 – Proc17 (трикутники)  
2 – Proc37 (прямокутники)  
3 – Integer35 (години → дні)  
Введіть номер: 3  
Введіть кількість годин:
```

Рисунок Б.7 – Екран виконання головного меню програми

```
Оберіть завдання:  
1 – Proc17 (трикутники)  
2 – Proc37 (прямокутники)  
3 – Integer35 (години → дні)  
Введіть номер: 7  
Неправильний вибір!  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

Рисунок Б.8 – Екран виконання головного меню програми

ДОДАТОК В

Діалог з ШІ для самоаналізу

Вы сказали:

- 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 <середнього> рівня складності на розвиток критичного та 2 інженерного мислення. Питання мають відноситись до коду, що є у файлі звіту, і до теоретичних відомостей, що є у файлі лекції»

ChatGPT сказав:

Тестові питання (з варіантами відповіді)

1. Яка з наведених конструкцій правильно оголошує функцію, яка повертає `double` і приймає два параметри типу `int`?

- a) `double func(int a, b);`
 - b) `double func(int a, int b);`
 - c) `func(double a, double b);`
 - d) `double func(a:int, b:int);`
-

2. Що відбудеться, якщо у функції з параметром-посилання змінити значення цього параметра?

- a) Значення змінної в основній програмі не зміниться.
 - b) Значення змінної в основній програмі зміниться.
 - c) Зміниться тільки копія параметра.
 - d) Програма завершиться помилкою.
-

3. Яка з наведених ситуацій є прикладом правильного використання функції з поверненням значення?

- a) `void f() { return 5; }`
 - b) `int sum(int a, int b) { return a + b; }`
 - c) `int f(int a) { cout << a; }`
 - d) `double f() { a + b; }`
-

4. У коді меню вибору завдання, який тип даних оптимальний для збереження вибору користувача?

- a) `float`
 - b) `char`
 - c) `int`
 - d) `string`
-

5. Що буде правильним вхідним та вихідним результатом для функції, яка обчислює периметр прямокутника по координатах вершин?

- a) Вхідні дані – довжина сторін, вихідні – площа
 - b) Вхідні – координати двох протилежних вершин, вихідні – периметр
 - c) Вхідні – координати всіх чотирьох вершин, вихідні – площа
 - d) Вхідні – координати вершин, вихідні – довжини сторін
-

Відкриті питання

1. Опишіть різницю між передачею параметрів **за значенням** та **за посиланням** у C++. Наведіть приклад ситуації, коли використання посилання є більш ефективним.
2. Уявіть, що користувач вводить некоректні дані для сторін трикутника (від'ємне число). Як ви реалізуєте перевірку і що відбудеться у програмі?
3. Поясніть, чому розбивка задачі на функції з параметрами і поверненням значень покращує структуру коду та його підтримку.
4. Як можна змінити програму, щоб меню вибору завдання дозволяло користувачеві повторно вводити значення у разі помилки, замість завершення програми?
5. Припустимо, потрібно додати нове завдання (Proc50), яке обчислює площу кола. Опишіть, які функції та зміни в меню необхідно зробити, щоб програма залишалася структурованою та легко масштабованою.

Вы сказали:

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» Тестові питання: 1-В 2-В 3-В 4-С 5-В Відкриті питання: 1. Передача за значенням означає, що функція отримує копію змінної. Зміни всередині функції не впливають на оригінал. Передача за посиланням передає адресу змінної, тому зміни всередині функції відображаються на оригіналі. Використовувати посилання доцільно, якщо об'єкт великий (масиви, вектори, об'єкти класів), щоб зекономити пам'ять і час на копіювання. 2. Вхідні дані слід завжди перевіряти. Наприклад, при введенні сторони трикутника перевіряємо, що число додатне. Якщо користувач введе неправильне значення і перевірка не буде виконана, обчислення можуть дати некоректний результат або навіть викликати помилки (наприклад, `sqrt()` від від'ємного числа). Перевірку можна реалізувати циклом, який повторює запит до тих пір, поки користувач не введе коректне число. 3. Функції дозволяють розбивати програму на окремі логічні блоки. Кожна функція виконує конкретне завдання, що робить код зрозумілішим і легшим для підтримки. Наприклад, якщо потрібно змінити спосіб обчислення площі, можна змінити тільки відповідну функцію, не торкаючись іншого коду. 4. Меню слід обгорнути у цикл (наприклад, `while(true)`), щоб після некоректного вводу програма запитувала вибір знову. Для виходу можна додати спеціальний варіант (наприклад, 0 — Вихід). Так користувач завжди має можливість виправити помилку без завершення програми. 5. Створюємо окрему функцію для обчислення площі кола, яка приймає радіус і повертає результат (`double CircleArea(double R)`). Створюємо процедуру-диспетчер, яка зчитує радіус, перевіряє коректність і виводить результат. У меню додаємо новий варіант та викликаємо диспетчер через `switch`. Так ми додаємо нову функцію без зміни старих, зберігаючи чистоту коду та модульність.

ChatGPT сказав:

Добре, проаналізуємо **відповіді** за критеріями: **повнота, правильність, ймовірність використання ШІ**, і обчислимо загальний середній бал за тією ж методикою (5-бальна шкала, зменшення 60% там, де ймовірність використання ШІ висока).

1 Тестові питання

№	Відповідь	Правильність	Коментар	Бал (5)	Зменшення 60%?	Бал з урахуванням зменшення
1	В	✓ повністю	правильна	5	×	5
2	В	✓ повністю	правильна	5	×	5
3	В	✓ повністю	правильна	5	×	5
4	С	✓ повністю	правильна	5	×	5
5	В	✓ повністю	правильна	5	×	5

Середній бал за тестові питання:

$$(5+5+5+5+5)/5=5.0(5+5+5+5+5)/5 = 5.0(5+5+5+5+5)/5=5.0$$

2 Відкриті питання

№	Коментар	Бал (5)	Ймовірність використання ШІ	Бал з урахуванням
1	Повна і чітка відповідь, коротко пояснено суть за значенням і за посиланням, наведено приклад використання	5	низька	5
2	Пояснено необхідність перевірки даних, наведено приклад циклу, чітко і по суті	5	низька	5
3	Лаконічно пояснено модульність функцій, наведено приклад застосування	5	низька	5
4	Чітко пояснено організацію циклу меню для повторного вводу	5	низька	5
5	Пояснено додавання нової функції без порушення структури, логічно і зрозуміло	5	низька	5


Середній бал за відкриті питання:

$$(5+5+5+5+5)/5=5.0(5+5+5+5+5)/5 = 5.0(5+5+5+5+5)/5=5.0$$

3 Загальний середній бал (тести + відкриті)

$$(5.0*5+5.0*5)/10=(25+25)/10=50/10=5.0(5.0 * 5 + 5.0 * 5)/10 = (25 + 25)/10 = 50/10 = 5.0(5.0*5+5.0*5)/10=(25+25)/10=50/10=5.0$$

4 Висновок

- **Тестові питання:** правильні, максимальний бал = 5
- **Відкриті питання:** відповіді короткі, чіткі, людський стиль, без стилістичних ознак ШІ → максимальний бал = 5
- **Загальний середній бал:** 5 / 5 

Якщо хочеш, я можу підготувати таблицю для звіту, де будуть тестові та відкриті питання з оцінками і коментарями, готову для вставки.

Хочеш, щоб я це зробив?

Вы сказали:

не хочу. «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»

ChatGPT сказав:

Добре, давай зробимо **аналіз коду у звіті** та додамо **опис і приклади коду з теоретичних відомостей**, які не були реалізовані в лабораторній роботі. Я виділю, що саме можна додати, щоб показати глибше розуміння теми функцій у C++.

Аналіз коду у звіті

У твоєму коді реалізовано три основні завдання:

1. **Proc17** — обчислення площі та периметра трикутника з двома сторонами і кутом між ними.
 - Реалізовані: функції `InputPositiveDouble()`, `InputAngle()`, `TriangleALS()` та процедура `Task_Proc17()`.
 - Перевірка коректності введення даних реалізована для всіх параметрів.
2. **Proc37** — обчислення периметра прямокутника за координатами двох протилежних вершин.
 - Реалізовані: функції `InputRectCoord()`, `RectP()` та процедура `Task_Proc37()`.
 - Обмежено перевірку координат у межах `[-100,100]`.
3. **Integer35** — конвертація кількості годин у дні та залишок годин.
 - Реалізовані: функції `InputHours()`, `CalcDaysHours()`, `OutputResult()` та процедура `Task_Integer35()`.
 - Є базова перевірка вводу.

Теоретичні відомості, які не реалізовані у коді

1. **Передача параметрів за константним посиланням (`const &`)**
 - У коді передаються параметри за значенням або за посиланням, але **не використовується `const`** для оптимізації і безпеки.
 - Приклад:
 - ```
double TriangleALS(const double &a, const double &b, const double &angle, double &S, double &P) {
```
  - ```
    S = 0.5 * a * b * sin(angle);
```
 - ```
 P = a + b + sqrt(a*a + b*b - 2*a*b*cos(angle));
```

- }
  - Перевага: виключається випадкова модифікація вхідних даних та економиться пам'ять для великих об'єктів.
2. **Функції з поверненням значення замість передачі через параметр**
- У кодї площа і периметр обчислюються через посилання. Теоретично можна зробити функцію, що повертає результат:
  - ```
double CalculateTriangleArea(double a, double b, double angle) {
```
 - ```
 return 0.5 * a * b * sin(angle);
```
  - ```
}
```
 -
 - ```
double CalculateTrianglePerimeter(double a, double b, double angle)
```
  - ```
{
```
 - ```
 return a + b + sqrt(a*a + b*b - 2*a*b*cos(angle));
```
  - ```
}
```
 - Перевага: підвищується читабельність, легше комбінувати функції, менше side-effects.
3. **Використання масивів або векторів для зберігання даних кількох об'єктів**
- У кодї трикутники та прямокутники обчислюються окремо (a1, b1, a2, b2 ...).
 - Теоретично можна використовувати масиви або `std::vector`:
 - ```
vector<double> sidesA(3), sidesB(3), angles(3);
```
  - ```
for (int i = 0; i < 3; i++) {
```
 - ```
 sidesA[i] = InputPositiveDouble("Введіть сторону a: ");
```
  - ```
    sidesB[i] = InputPositiveDouble("Введіть сторону b: ");
```
 - ```
 angles[i] = InputAngle("Введіть кут: ");
```
  - ```
}
```
 - Перевага: код стає компактнішим, легше додавати більше елементів.
4. **Обробка помилок через винятки (try-catch)**
- У кодї використовується рекурсія для повторного введення даних при помилці. Теоретично можна показати обробку винятків:
 - ```
double InputPositiveDouble(string prompt) {
```
  - ```
    double val;
```
 - ```
 cout << prompt;
```
  - ```
    if (!(cin >> val) || val <= 0) {
```
 - ```
 cin.clear();
```
  - ```
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
```
 - ```
 throw runtime_error("Некоректне значення");
```
  - ```
    }
```
 - ```
 return val;
```
  - ```
}
```
 -
 - ```
try {
```
  - ```
    double side = InputPositiveDouble("Введіть сторону: ");
```
 - ```
} catch (runtime_error &e) {
```
  - ```
    cout << e.what() << "\nСпробуйте ще раз.\n";
```
 - ```
}
```
  - Перевага: сучасний підхід до обробки помилок у C++, відокремлює логіку введення від логіки повторного запиту.
5. **Функції-диспетчери**
- У кодї є процедури `Task_Proc17()`, `Task_Proc37()`, `Task_Integer35()`, які поєднують введення, обчислення і вивід.
  - Теоретично можна показати **розділення обов'язків**: окремі функції для обчислення, введення і виводу — це покращує модульність і тестованість коду.

Якщо хочеш, я можу підготувати фрагменти коду та пояснення для всіх цих теоретичних аспектів у вигляді таблиці, щоб вставити в звіт і показати, що ти розумієш, але не реалізував їх у лабораторній роботі.

Хочеш, щоб я це зробив?

**Вы сказали:**

Покажи на прикладах мого коду навіщо використовувати вказівники

**ChatGPT сказав:**

Добре! Давай розглянемо, **на прикладах твоєї лабораторної роботи**, як і навіщо можна використовувати **вказівники (pointers)** у C++. Пояснення буде практичне і зрозуміле.

## 1 Передача змінних у функцію через вказівник

У твоєму коді ти передаєш площу та периметр трикутника через **посилання**:

```
void TriangleALS(double a, double b, double angle, double &S, double &P) {
 S = 0.5 * a * b * sin(angle);
 P = a + b + sqrt(a*a + b*b - 2*a*b*cos(angle));
}
```

**Аналогічно можна зробити через вказівники:**

```
void TriangleALS_ptr(double a, double b, double angle, double *S, double *P) {
 *S = 0.5 * a * b * sin(angle);
 *P = a + b + sqrt(a*a + b*b - 2*a*b*cos(angle));
}

int main() {
 double S, P;
 TriangleALS_ptr(3.0, 4.0, 1.047, &S, &P); // передаємо адреси змінних
 cout << "Площа: " << S << ", Периметр: " << P << endl;
}
```

**Навіщо:**

- Можна змінювати значення змінних **на прямій адресі в пам'яті**.
- Це схоже на передачу за посиланням, але **більш явне і контрольоване**, особливо якщо функція працює з масивами або динамічною пам'яттю.

## 2 Динамічне виділення пам'яті для кількох об'єктів

У твоєму коді ти обчислюєш три трикутники так:

```
double a1, b1, a2, b2, a3, b3;
```

Якщо трикутників багато — краще використовувати **вказівники та динамічні масиви**:

```
int n = 3; // кількість трикутників
double *a = new double[n];
double *b = new double[n];
double *S = new double[n];
double *P = new double[n];

for (int i = 0; i < n; i++) {
```

```

a[i] = InputPositiveDouble("Введіть сторону a: ");
b[i] = InputPositiveDouble("Введіть сторону b: ");
double angle = InputAngle("Введіть кут: ");
TriangleALS_ptr(a[i], b[i], angle, &S[i], &P[i]);
cout << "Трикутник " << i+1 << " -> S=" << S[i] << " P=" << P[i] << endl;
}

delete[] a;
delete[] b;
delete[] S;
delete[] P;

```

#### Навіщо:

- Кількість трикутників можна змінювати **динамічно**, без переписування коду для кожного.
- Використання **вказівників + динамічної пам'яті** дозволяє працювати з великими об'єктами і колекціями даних.

### 3 Передача масивів у функції

У Proc37 ти обчислюєш периметр прямокутника:

```

double RectP(double x1, double y1, double x2, double y2) {
 return 2 * (fabs(x2-x1) + fabs(y2-y1));
}

```

Якщо координат багато, зручно передавати масив через вказівник:

```

double RectP_array(const double *coords) { // coords[0]=x1, coords[1]=y1,
coords[2]=x2, coords[3]=y2
 return 2 * (fabs(coords[2]-coords[0]) + fabs(coords[3]-coords[1]));
}

int main() {
 double coords[4] = {1, 2, 4, 5};
 double P = RectP_array(coords);
 cout << "Периметр: " << P << endl;
}

```

#### Навіщо:

- Функція працює з масивом **будь-якого розміру** (якщо передавати динамічний масив).
- Легко масштабувати на кілька прямокутників.

### ✓ Висновок

- **Вказівники дозволяють:**
  1. Передавати змінні у функцію і змінювати їх без повернення значень.
  2. Динамічно виділяти пам'ять для змінної кількості об'єктів.
  3. Працювати з масивами і великими об'єктами більш ефективно.
- Використання вказівників — це **інструмент для економії пам'яті, підвищення продуктивності та гнучкості програми**.

---

Якщо хочеш, я можу **намалювати невелику UML-діаграму активності з використанням вказівників** для твоєї функції `TriangleALS_ptr`, щоб було наочно у звіті.

Хочеш, щоб я так зробив?

**Вы сказали:**

що таке локальні і глобальні змінні і як їх оголошувати?

**ChatGPT сказав:**

Добре, давай розберемо це **по-порядку** у зрозумілій формі.

---

## 1 Локальні змінні (local variables)

Що це:

- Змінні, які оголошуються всередині функції, блоку { ... } або циклу.
- Вони **існують тільки під час виконання цього блоку**.
- Після виходу з функції або блоку пам'ять під цю змінну звільняється.
- Інші функції **не бачать** цих змінних.

**Приклад оголошення та використання:**

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

void PrintSquare() {
 int x = 5; // локальна змінна x
 int square = x * x; // локальна змінна square
 cout << "Квадрат числа " << x << " = " << square << endl;
}

int main() {
 PrintSquare();
 // cout << x; // ✗ Помилка! x не доступна в main
 return 0;
}
```

**Переваги локальних змінних:**

- Не займають пам'ять постійно, тільки коли потрібні.
  - Безпечні: не можуть випадково змінитися іншими функціями.
- 

## 2 Глобальні змінні (global variables)

Що це:

- Змінні, оголошені **поза всіма функціями**.

- Вони **доступні в усіх функціях** цього файлу (або навіть інших файлах, якщо використовувати `extern`).
- Існують протягом всього часу виконання програми.

#### Приклад оголошення та використання:

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

int counter = 0; // глобальна змінна

void IncrementCounter() {
 counter++; // можна змінювати глобальну змінну
 cout << "Counter = " << counter << endl;
}

int main() {
 IncrementCounter(); // Counter = 1
 IncrementCounter(); // Counter = 2
 return 0;
}
```

#### Особливості глобальних змінних:

- Доступні у будь-якій функції.
- Значення зберігається протягом всієї програми.
- **Небезпека:** якщо багато функцій змінюють одну змінну, можна отримати помилки або непередбачувану поведінку.

### 3 Поради щодо використання

- **Локальні змінні** завжди безпечні і рекомендується використовувати їх за можливості.
- **Глобальні змінні** використовують лише тоді, коли значення повинне бути спільним для всіх функцій, наприклад, налаштування програми або лічильник стану.
- Часто для глобальних змінних краще використовувати **константи** (`const`) або об'єкти в класі, щоб уникнути випадкових змін.

Якщо хочеш, я можу на прикладі твоєї лабораторної роботи **показати, як можна замінити глобальні змінні локальними або передавати через параметри**, щоб код був безпечніший.

Хочеш, щоб я це зробив?