

Міністерство освіти і науки України

Національний технічний університет України «Київський політехнічний  
інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 6 з дисципліни

«Алгоритми та структури даних-1.

Основи алгоритмізації»

«Дослідження рекурсивних алгоритмів»

Варіант 25

Виконав ПІ-15, Плугатирьов Дмитро Валерійович

студент (шифр, прізвище, ім'я, по батькові)

Перевірів Всечерковська Анастасія Сергіївна

(прізвище, ім'я, по батькові)

Київ 2021

## Лабораторна робота № 6

### Дослідження рекурсивних алгоритмів

**Мета** - дослідити особливості роботи рекурсивних алгоритмів та набути практичних навичок їх використання під час складання програмних специфікацій підпрограм.

#### Варіант 25

#### Завдання

25.) Отримати всі піфагорові трійки натуральних чисел, кожне з яких не перевищує  $n$ , тобто всі такі трійки натуральних чисел  $a, b, c$ , що  $a^2 + b^2 = c^2$  ( $a \leq, b \leq, c \leq n$ ).

#### 1. Постановка задачі

Оскільки нашою остаточною ціллю є отримання значень змінних, які утворюють піфагорові трійки, значення яких не може перевищувати задане користувачем, то можна знайти кожну трійку за допомогою використання рекурсії. Числа є натуральними та обраховуються наступним чином: якщо під час одного із рекурсивних викликів функції вийшло так, що третя змінна ( $c$ ) дорівнює введеному користувачем числу, то відбуватиметься присвоєння одиниці до  $c$  та збільшення значення змінної, яка розташовується лівіше у формулі Піфагора на одиницю.

Результатом є виведення усіх можливих піфагорових трійок з дублюванням (оскільки перша та друга змінні дають правильний результат, коли є взаємо обернені) та кількості трійок без урахування дублювання.

## 2. Побудова математичної моделі

Змінна	Тип	Ім'я	Призначення
Дане число	Цілочисельний	n	Початкові дані
Перша змінна	Цілочисельний	a	Проміжні дані
Друга змінна	Цілочисельний	b	Проміжні дані
Третя змінна	Цілочисельний	c	Проміжні дані
Кількість трійок	Цілочисельний	count	Проміжні дані
Пошук піфагорових трійок	Процедура	pythagorean_trio	Проміжні дані
Виведення трійок	Процедура	output	Результат

- Для вирішення даної задачі треба скористатися теоремою Піфагора: сума квадратів катетів прямокутного трикутника дорівнює квадрату гіпотенузи.
- Пошук піфагорових трійок відбувається шляхом послідовного перебору значень правої змінної до значення, заданого користувачем, після проходження якого значення зліва збільшується на одиницю і т.д.

Дія  $\text{row}(x, m)$  означає піднесення числа  $x$  до степені  $m$ .

Дії  $x += 1$  та  $++x$  означають  $x := x + 1$ .

Дія **вивести**  $x, v, t$  означає послідовне виведення  $x, v, t$ .

## 3. Розв'язання

Програмні специфікації записати у псевдокодi та графічній формi у вигляді блок-схеми.

*Крок 1.* Визначити основні дії.

*Крок 2.* Введення обмежувальної змінної та перевірка її значення на натуральність.

*Крок 3.* Виклик підпрограми, реалізація рекурсій в ній, складання умови виведення потрібних даних.

#### **4. П с е в д о к о д**

##### **Основна програма:**

*Крок 1*

**початок**

введення обмежувальної змінної та перевірка її значення на натуральність

виклик підпрограми, реалізація рекурсій в ній, складання умови виведення потрібних даних

**кінець**

*Крок 2*

**початок**

a := 1

b := 1

c := 1

count := 0

**ввести n**

**повторити**

**поки** n <= 0

**ввести n**

**все повторити**

виклик підпрограми, реалізація рекурсій в ній, складання умови виведення потрібних даних

**кінець**

*Крок 3*

**початок**

a := 1

b := 1

c := 1

count := 0

```
    ввести n
    повторити
    поки  $n \leq 0$ 
        ввести n
    все повторити
    pythagorean_trio(n, a, b, c, count)
    вивести count
кінєць
```

### Підпрограми:

```
output(a, b, c, count)
    вивести count, a, b, c
кінєць
```

```
pythagorean_trio(n, a, b, c, count)
    якщо  $\text{pow}(a, 2) + \text{pow}(b, 2) == \text{pow}(c, 2)$ 
        то
            ++count
            output(a, b, c, count)
    все якщо

    якщо  $c \neq n$ 
        то
            pythagorean_trio(n, a, b, ++c, count)
    інакше якщо  $c == n$  та  $b \neq n$ 
        то
             $c := 1$ 
            pythagorean_trio(n, a, ++b, c, count)
    інакше якщо  $c == n$  та  $b == n$  та  $a \neq n$ 
        то
```

$b := 1$

$c := 1$

$\text{pythagorean\_trio}(n, ++a, b, c, \text{count})$

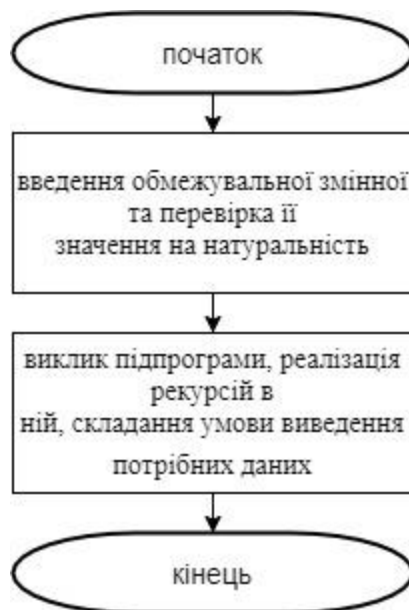
**все якщо**

**кінець**

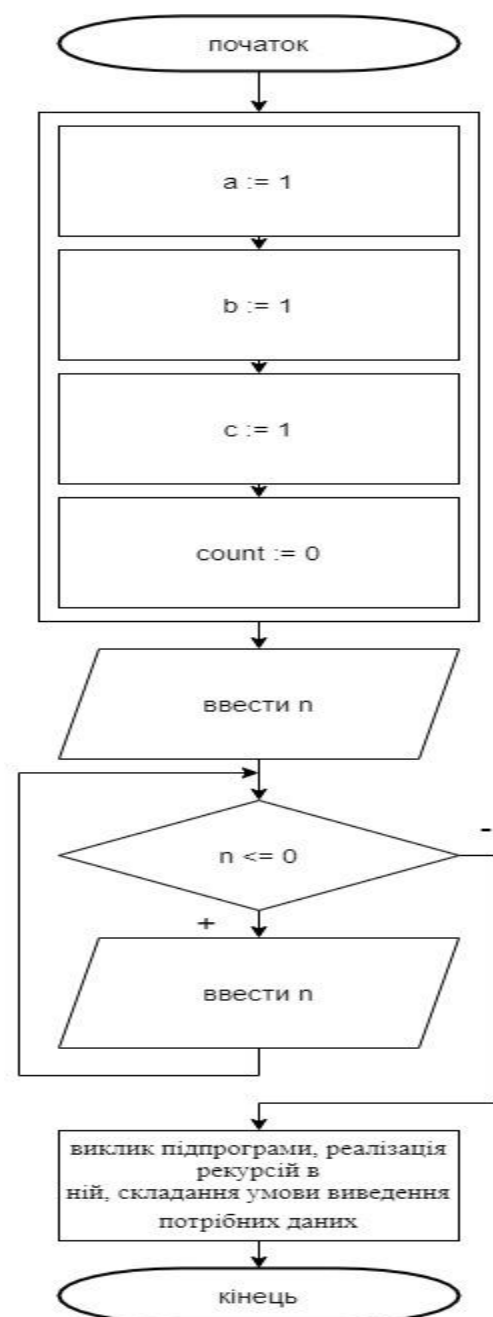
*Блок-схема*

**Основна програма:**

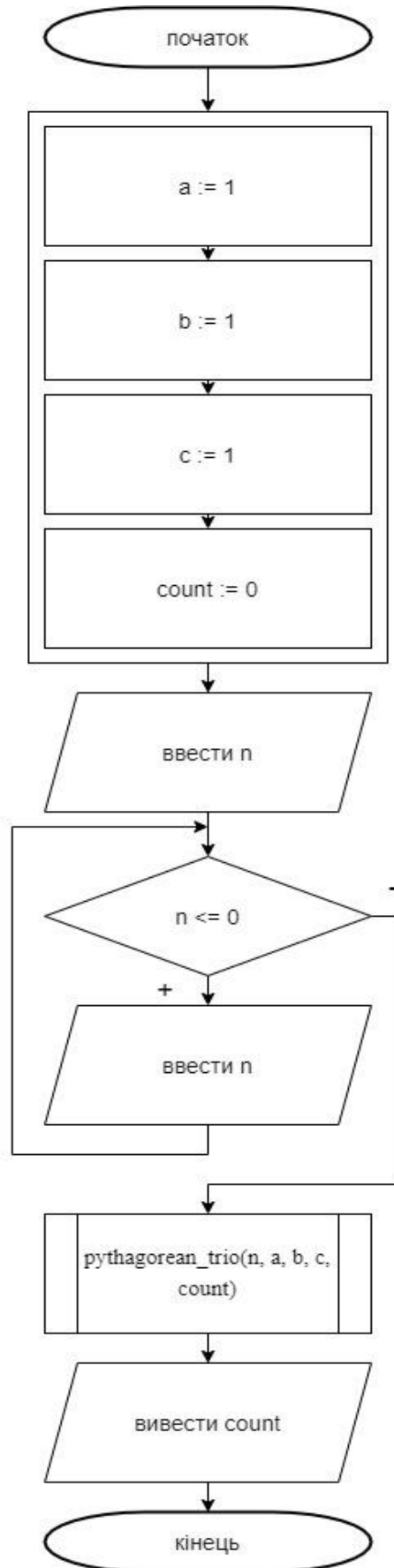
*Крок 1*



*Крок 2*



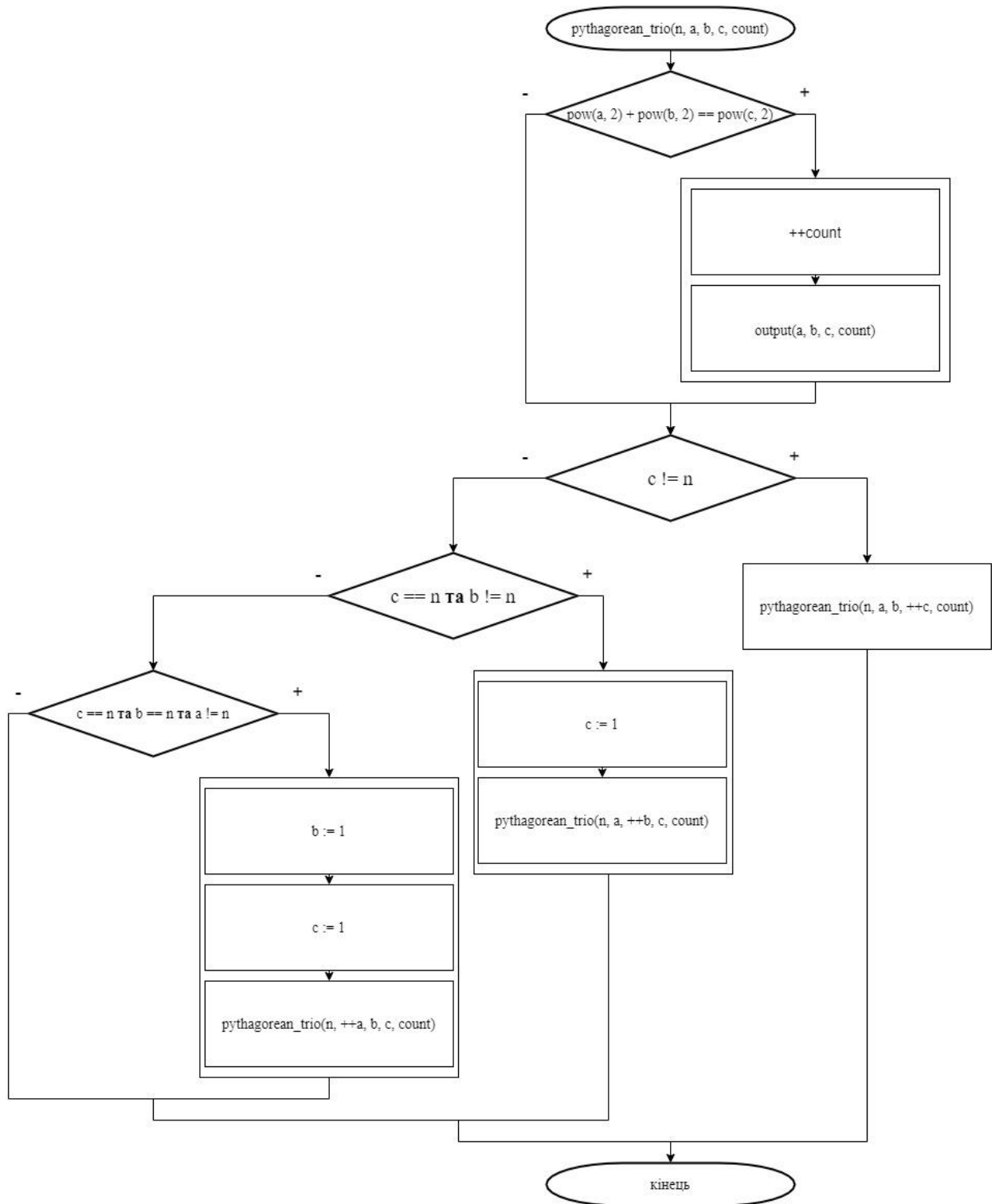
### Крок 3



## Підпрограми:







## Код програми

```
#include <iostream>
#include <cmath>

void pythagorean_trio(int n, int a, int b, int c, int& count);
void output(int a, int b, int c, int count);

// TASK 25

int main()
{
    int n, a = 1, b = 1, c = 1, count = 0;
    // Input of the numbers
    std::cout << "Please, enter the natural n: ";
    std::cin >> n;
    while (n <= 0)    // The condition of a natural number
    {
        std::cout << "n is less or equal to 0. Please, enter a natural one: ";
        std::cin >> n;
    }
    pythagorean_trio(n, a, b, c, count);    // Calculation of pythagorean trios
    std::cout << "The quantity of pythagorean trios is: " << count / 2 << std::endl;
    // Output of the count of trios
}

void pythagorean_trio(int n, int a, int b, int c, int& count)
{
    if (a <= n && b <= n && c <= n)
    {
        if (pow(a, 2) + pow(b, 2) == pow(c, 2)) // Completed pythagorean trio
        {
            count++;
            output(a, b, c, count);
        }

        if (c != n)
        {
            pythagorean_trio(n, a, b, ++c, count);
        }
        else if (c == n && b != n)
        {
            c = 1;
            pythagorean_trio(n, a, ++b, c, count);
        }
        else if (c == n && b == n && a != n)
        {
            b = 1;
            c = 1;
            pythagorean_trio(n, ++a, b, c, count);
        }
    }
}

void output(int a, int b, int c, int count) // Output of a, b and c
{
    std::cout << "There is the " << count << " pythagorean trio! It's consisted of: "
    << std::endl;
```

```

std::cout << "a, which equals to: " << a << std::endl;
std::cout << "b, which equals to: " << b << std::endl;
std::cout << "c, which equals to: " << c << std::endl;
}

```

## 5.) Т е с т у в а н н я

Блок	Дія 1		Дія 2
	Початок		Початок
1	a := 1, b := 1, c := 1, count := 0	1	a := 1, b := 1, c := 1, count := 0
2	n := 5	2	n := 10
3	pythagorean_trio(n, a, b, c, count)	3	pythagorean_trio(n, a, b, c, count)
4	pythagorean_trio(n, a, b, ++c, count)	4	pythagorean_trio(n, a, b, ++c, count)
5	pythagorean_trio(n, a, b, ++c, count)	5	pythagorean_trio(n, a, b, ++c, count)
...	...	...	...
22	c := 1	22	c := 1
23	pythagorean_trio(n, a, ++b, c, count)	23	pythagorean_trio(n, a, ++b, c, count)
24	pythagorean_trio(n, a, b, ++c, count)	24	pythagorean_trio(n, a, b, ++c, count)
...	...	...	...
44	c := 1	44	c := 1
45	pythagorean_trio(n, a, ++b, c, count)	45	pythagorean_trio(n, a, ++b, c, count)
...	...	...	...
111	b := 1	111	b := 1
112	c := 1	112	c := 1
113	pythagorean_trio(n, ++a, b, c, count)	113	pythagorean_trio(n, ++a, b, c, count)
...	...	...	...
310	count += 1	310	count += 1
311	a := 3, b := 4, c := 5, count := 1, output(a, b, c, count)	311	a := 3, b := 4, c := 5, count := 1, output(a, b, c, count)
...	...	...	...
409	count += 1	409	count += 1

	<b>Кінець</b>
--	---------------

<b>410</b>	<b>a := 4, b := 3, c := 5, count := 2, output(a, b, c, count)</b>	<b>410</b>	<b>a := 4, b := 3, c := 5, count := 2, output(a, b, c, count)</b>
		<b>...</b>	<b>...</b>
		<b>620</b>	<b>count += 1</b>
		<b>621</b>	<b>a := 6, b := 8, c := 10, count := 3, output(a, b, c, count)</b>
		<b>...</b>	<b>...</b>
		<b>820</b>	<b>count += 1</b>
		<b>821</b>	<b>a := 8, b := 6, c := 10, count := 4, output(a, b, c, count)</b>
			<b>Кінець</b>

## 6.) В и с н о в о к

В цій лабораторній роботі мені довелося зайнятися дослідженням особливостей роботи рекурсивних алгоритмів та закріпити навички на практиці, складаючи програмні специфікації підпрограм. А саме, реалізація виводу трьох чисел, які утворюють піфагорову трійку за допомогою рекурсії і певних умов виводу даних та виходу з неї.