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 №4 з дисципліни
«Алгоритми та структури даних-1.
Основи алгоритмізації»

«Допоміжні алгоритми»

Варіант 13

Виконав студент ІП-43 Дутов Іван Андрійович

Перевірила Вітковська Ірина Іванівна

Київ 2024

Лабораторна робота №4
ДОПОМІЖНІ АЛГОРИТМИ

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

Задача. Розробити підпрограму для перевірки того, чи є задане натуральне число квадратом деякого натурального числа. Використовуючи її, порахувати кількість повних квадратів натуральних чисел, що належать заданому відрізку $[A; B]$.

Постановка задачі

Підпрограма перевірки, чи є число N повним квадратом, так чи інакше залучає знаходження **натурального** числа k на відрізку $[1; N]$, що $k^2 = N$. Оскільки послідовність $1, 2, \ldots, N$ можна вважати відсортованою, то доцільно застосувати алгоритм бінарного пошуку для знаходження k.

Алгоритм бінарного пошуку працює за допомогою двох вказівників: один, l, вказує на ліву межу (найменше значення), а інший, r, — на праву межу (найбільше значення). Процес виконується так:

1. **Поки** $l \leq r$:
   Визначаємо середнє значення за формулою $m = l + \frac{r-l}{2}$ (цілочисельне ділення)

2. Перевіряємо, чи є m^2 відповідним:

   - Якщо $m^2 > N$, то повними квадратами можуть бути значення між вказівниками l та $m - 1$, тому ми встановлюємо $r := m - 1$.
   - Якщо $m^2 < N$, то повними квадратами можуть бути значення між вказівниками $m + 1$ і r, тому встановлюємо $l := m + 1$.
   - Якщо $m^2 = N$, то ми знайшли шукане натуральне число.

Таким чином, алгоритм ефективно знаходить число k, якщо таке існує.

Для правильної роботи програми необхідно, щоб відрізок $[A; B]$ підходив, тобто $A \leq B$, для чого застосуємо альтернативну форму вибору. Також використаємо арифметичний цикл для чисел від A до B, аби порахувати кількість повних квадратів.

Табл. 1: Таблиця імен змінних

Програма			
Змінна	**Тип**	**Ім'я**	**Призначення**
Початок відрізка	Цілий	`A`	Ввід
Кінець відрізка	Цілий	`B`	Ввід
Кількість повних квадратів у відрізку	Цілий	`result`	Вивід
Підпрограма			
Змінна	**Тип**	**Ім'я**	**Призначення**
Число для перевірки N	Цілий	`num`	Формальний параметр
Ліва межа $\sqrt{N}$	Цілий	`l`	Допоміжна змінна
Права межа $\sqrt{N}$	Цілий	`r`	Допоміжна змінна
Можливе шукане значення $\sqrt{N}$	Цілий	`m`	Допоміжна змінна

Крок 1. Визначаємо основні дії.

Крок 2. Деталізуємо дію ініціалізації змінних A і B.

Крок 3. Деталізуємо дію перевірки на існування відрізка $[A; B]$.

Крок 4. Деталізуємо дію обчислення кількості повних квадратів на відрізку $[A; B]$.

Псевдокод (основний алгоритм)

Крок 1

1. **Початок**.
2. Ініціалізація A і B.
3. Перевірка на існування відрізка $[A; B]$.
4. Обчислення кількості повних квадратів на відрізку $[A; B]$.
5. **Кінець**.

Крок 2

1. **Початок**.
2. Ініціалізація A і B.
   2.1 **Ввід** A.
   2.2 **Ввід** B.
3. Перевірка на існування відрізка $[A; B]$.
4. Обчислення кількості повних квадратів на відрізку $[A; B]$.
5. **Кінець**.

Крок 3

1. **Початок**.
2. Ініціалізація A і B.
   2.1 **Ввід** A.
   2.2 **Ввід** B.
3. **Якщо** A <= B,
   то
   3.1 **Вивід** «ПОМИЛКА: А повинно бути меншим-рівним В.»
   інакше
4. Обчислення кількості повних квадратів на відрізку $[A; B]$.
5. **Кінець**.

Крок 4

1. **Початок**.

2. Ініціалізація `A` і `B`.

   2.1 **Ввід** `A`.

   2.2 **Ввід** `B`.

3. **Якщо** `A <= B`,
   то

   3.1 **Вивід** «ПОМИЛКА: A повинно бути меншим-рівним B.»

   інакше

4. Обчислення кількості повних квадратів на відрізку $[A; B]$.

   4.1 `result := 0`

   4.2 **Повторити для** `i := A` до `B`:

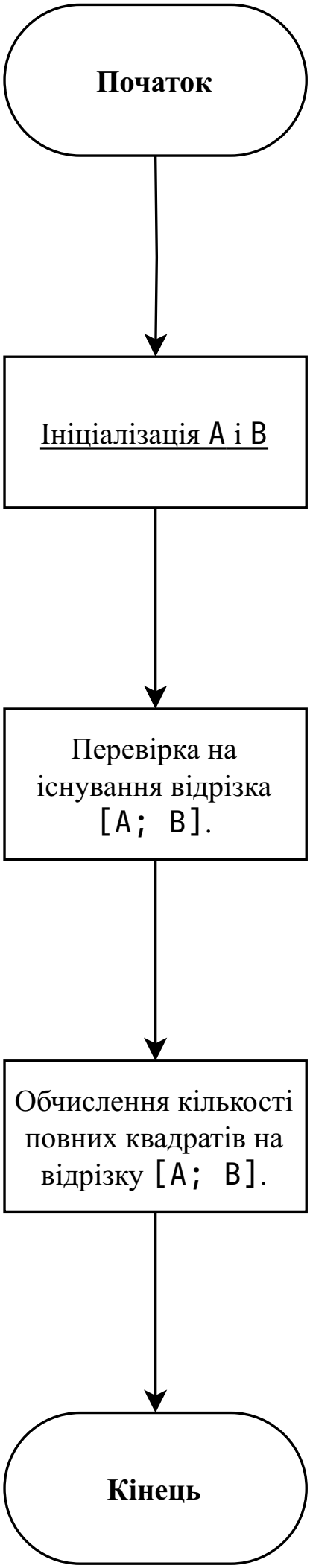
   4.2.1 **Якщо** `is_perfect_square(i) == 1`,
   то
   `result := result + 1`

   все повторити

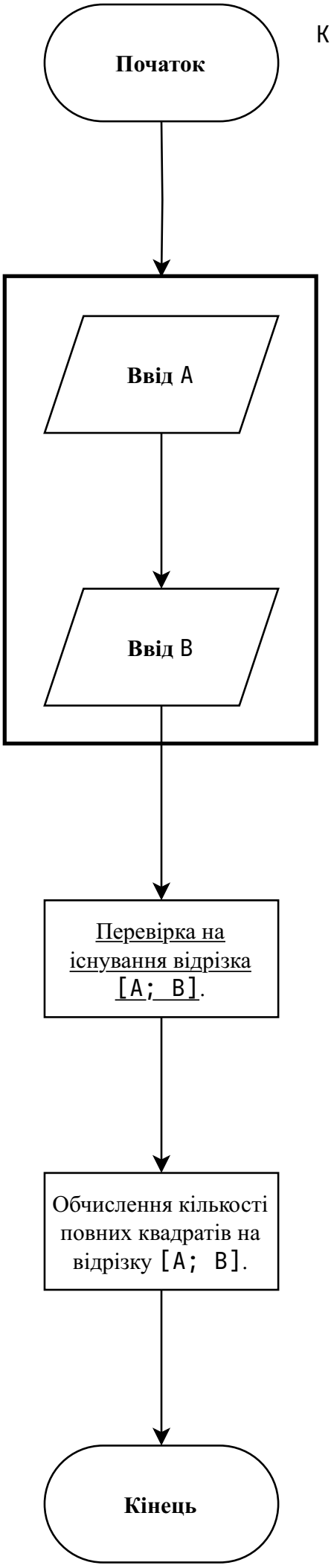
5. **Кінець**.

Блок-схеми (Основний алгоритм)

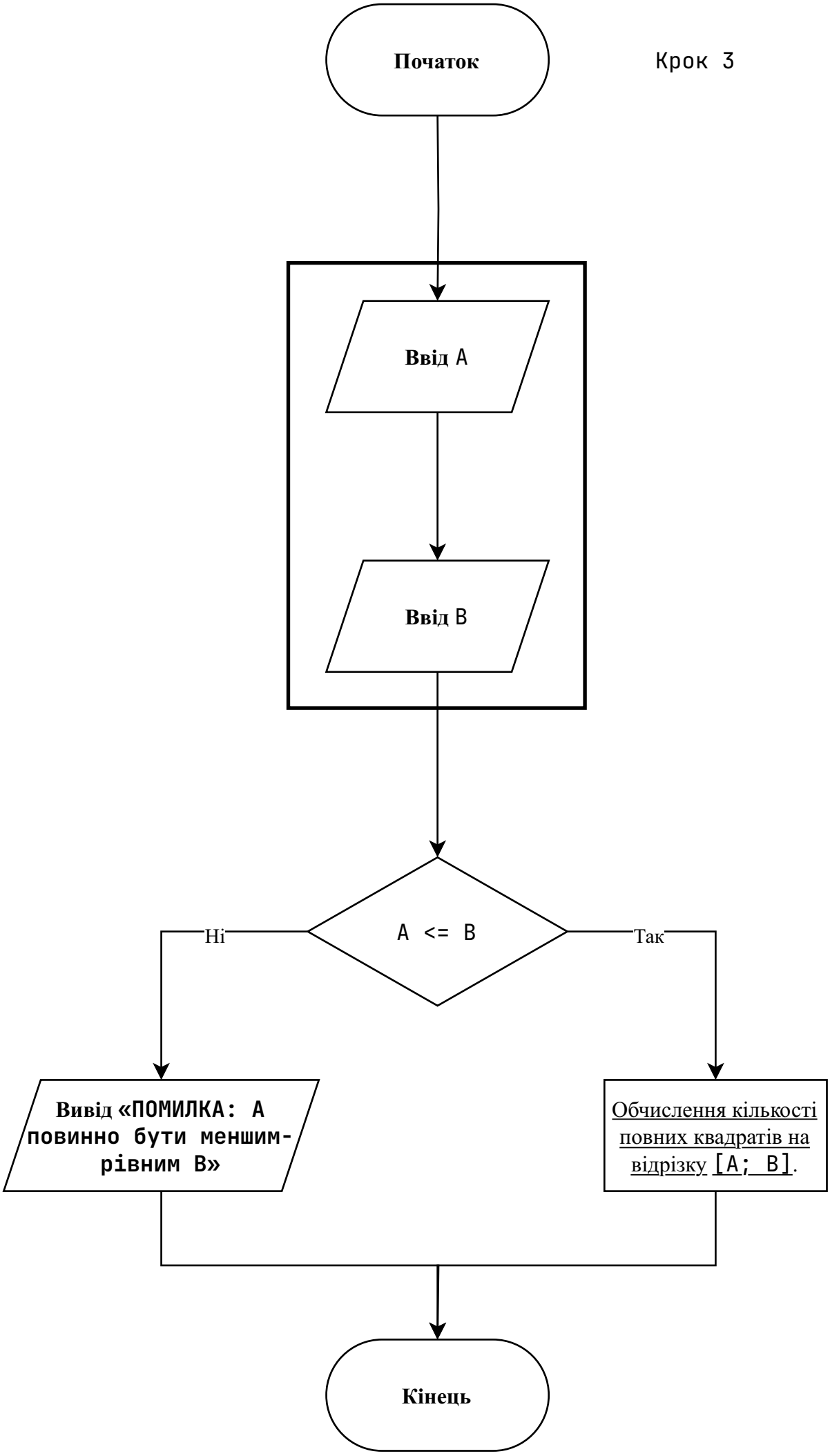
Крок 1

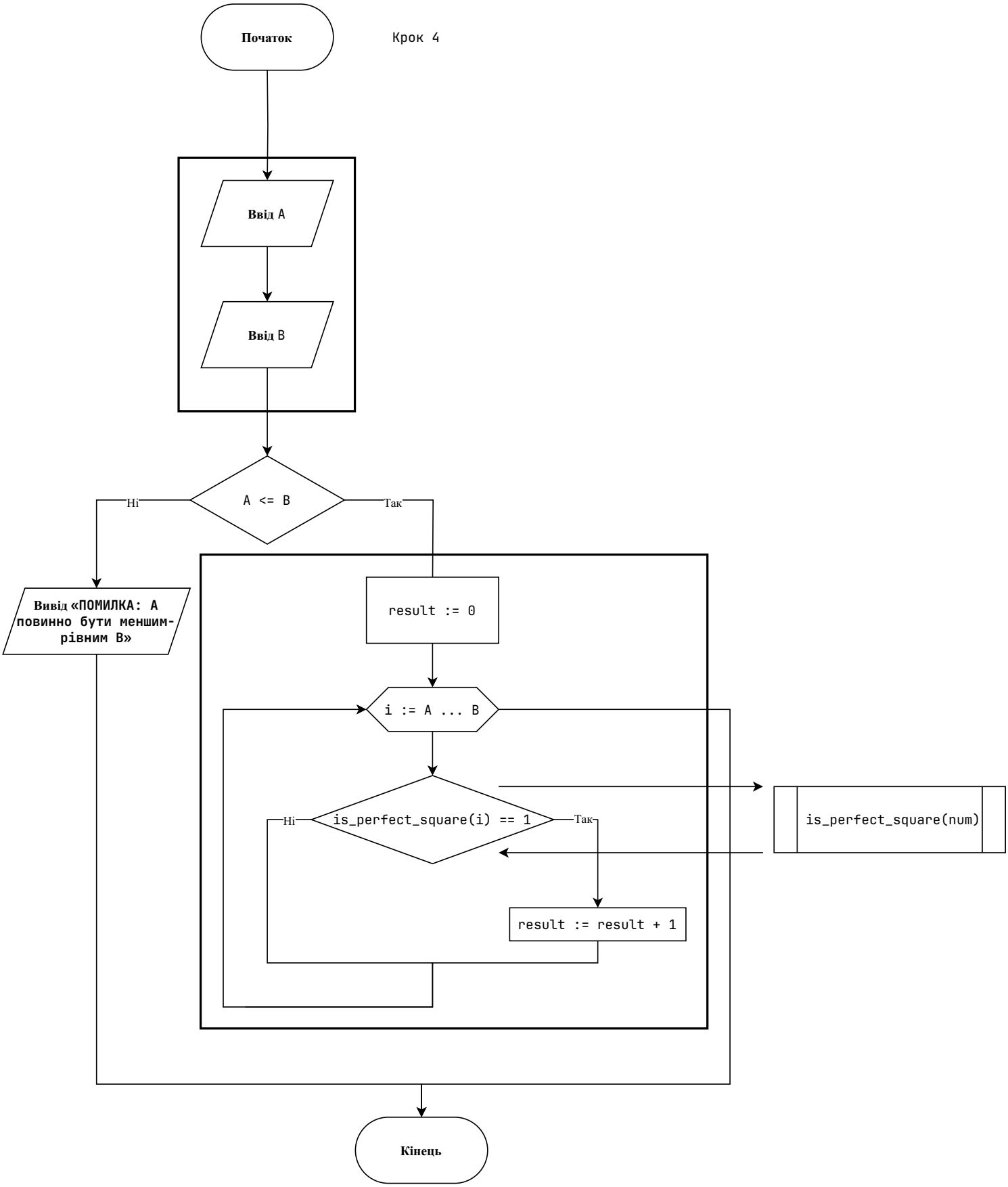

Початок
Ініціалізація A і B
Перевірка на існування відрізка [A; B].
Обчислення кількості повних квадратів на відрізку [A; B].
Кінець

Крок 2


Початок
Ввід A
Ввід B
Перевірка на існування відрізка [A; B].
Обчислення кількості повних квадратів на відрізку [A; B].
Кінець

Крок 3


Початок
Ввід A
Ввід B
A <= B
Ні
Так
Вивід «ПОМИЛКА: A повинно бути меншим-рівним B»
Обчислення кількості повних квадратів на відрізку [A; B].
Кінець


Початок
Крок 4
Ввід A
Ввід B
A <= B
Ні
Так
Вивід «ПОМИЛКА: A повинно бути меншим-рівним B»
result := 0
i := A ... B
is_perfect_square(i) == 1
Ні
Так
is_perfect_square(num)
result := result + 1
Кінець

Псевдокод (допоміжний алгоритм)

1. **Початок** `is_perfect_square(num)`.
2. `l := 1`
3. `r := num`
4. **Поки** `l <= r`,
   повторити

   4.1 `m := l + (r - l) / 2`

   4.2 **Якщо** `m * m > num`,
   то

   4.2.1 `r := m - 1`

   інакше

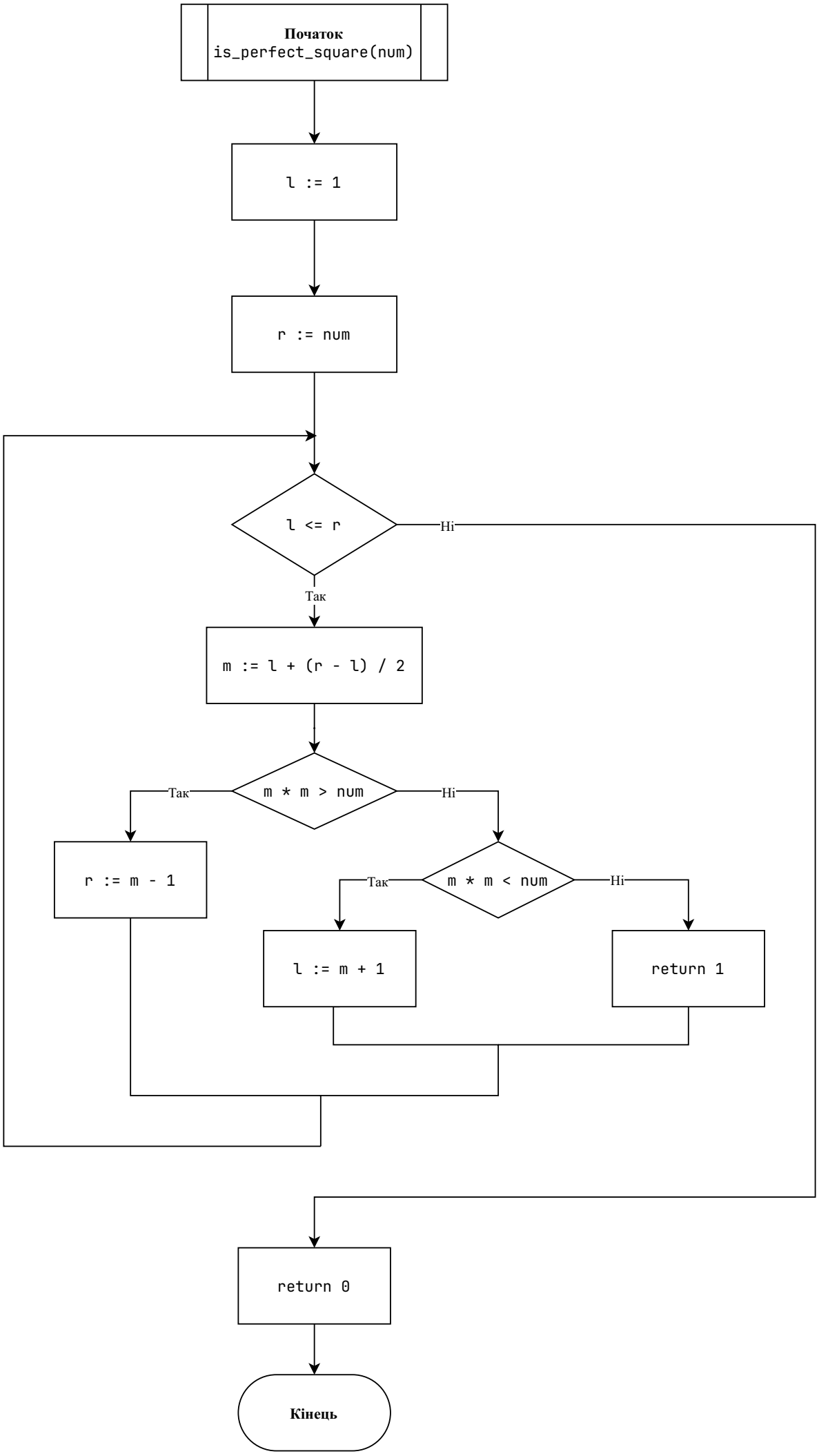
   4.3 **якщо** `m * m < num`,
   то

   4.3.1 `l := m + 1`

   інакше

   4.4 `return 1`

   все повторити
5. `return 0`
6. **Кінець**.


Початок
is_perfect_square(num)
l := 1
r := num
l <= r
Ні
Так
m := l + (r - l) / 2
m * m > num
Так
Ні
r := m - 1
m * m < num
Так
Ні
l := m + 1
return 1
return 0
Кінець

Код програми мовою C

```c
#include <stdio.h>

int is_perfect_square(int num);

int main() {
  int A, B;
  printf("Початок відрізка A: ");
  scanf("%d", &A);
  printf("Кінець відрізка B: ");
  scanf("%d", &B);

  if (A > B) {
    printf("ПОМИЛКА: A має бути меншим або рівним B.\n");
    return 1;
  }

  int result = 0;
  for (int i = A; i <= B; i++) {
    if (is_perfect_square(i))
      result++;
  }

  printf("Кількість повних квадратів на [A; B]: %d\n", result);

  return 0;
}

int is_perfect_square(int num) {
  int l = 1, r = num;
  int m;
  while (l <= r) {
    m = l + (r - l) / 2;
    if (m * m > num) {
      r = m - 1;
    } else if (m * m < num) {
      l = m + 1;
    } else {
      return 1;
    }
  }
  return 0;
}
```

Тестування програми

```
Початок відрізка A: 1
Кінець відрізка B: 100
Кількість повних квадратів на [A; B]: 10

Початок відрізка A: 255
Кінець відрізка B: 1024
Кількість повних квадратів на [A; B]: 17

Початок відрізка A: 87
Кінець відрізка B: 99
Кількість повних квадратів на [A; B]: 0

Початок відрізка A: 1000
Кінець відрізка B: 1
ПОМИЛКА: A має бути меншим або рівним B.
```

Висновок

Ми дослідили особливості роботи допоміжних алгоритмів та набули практичних навичок їх використання під час складання програмних специфікацій підпрограм на прикладі підпрограми для визначення, чи є число повним квадратом.