Міністерство освіти і науки України

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

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації

і управління

Звіт

з лабораторної роботи № 5 з дисципліни

«Алгоритми та структури даних-1.

Основи алгоритмізації»

«Дослідження складних циклічних алгоритмів»

Варіант 25

Виконав студент ІП-01 Гагарін Артем Олексійович

 (шифр, прізвище, ім'я, по батькові)

Перевірив Вітковська Ірина Іванівна

  ( прізвище, ім'я, по батькові)

Київ 2020

**Лабораторна робота 5**

**Дослідження складних циклічних алгоритмів**

**Мета** – дослідити особливості роботи складних циклів та набути практичних навичок їх

використання під час складання програмних специфікацій.

**Задача 25.** Дано число а. Знайти найближче до нього просте число.

**1. Постановка задачі.** Результатом розв’язку є знайдене найближче до а просте число.

**2**. **Побудова математичної моделі**. Складемо таблицю імен змінних.

|  |  |  |  |
| --- | --- | --- | --- |
| Змінна | Тип | Ім’я | Призначення |
| a | Дійсне | a | Початкове дане |
| Дільник | Цілий | і | Значення дільника |
| Число після а | Цілий | after\_a | Збереження значення цілого числа, більшого за а, що може бути простим |
| Число перед а | Цілий | before\_a | Збереження значення цілого числа, меншого за а, що може бути простим |
| Прапорець для after\_a | Логічний | p | Вказує, чи after\_a є простим числом |
| Прапорець для before\_a | Логічний | q | Вказує, чи before\_a є простим числом |
| Прапорець для a | Логічний | z | Вказує, чи є округлене а простим числом |

Для початку визначимо, що є простим числом. Просте число — це **натуральне** число, яке має рівно два різні **натуральні** дільники: одиницю і саме число. З цього можна зробити висновок, що числа 0 та 1 не є простими, отже найменше просте число рівне 2. Таким чином, 2 буде найближчим простим числом при будь-яких а < 2 та 2 < a < 2,5.

Найближче просте число до а може бути менше або більше за нього. Наприклад, для числа 8 найближче просте число – 7, а для числа 10 – 11. Тобто потрібно враховувати, яке число ближче до а. Також можливий випадок, коли найближче просте число, що більше а, та найближче просте число, що менше а, знаходяться на однаковій відстані від а. Наприклад, число 4 має найближчими простими числами 3 і 5.

Якщо округлене а = а, то число ціле. Якщо округлене а є простим числом, то воно є найближчим простим числом для а.

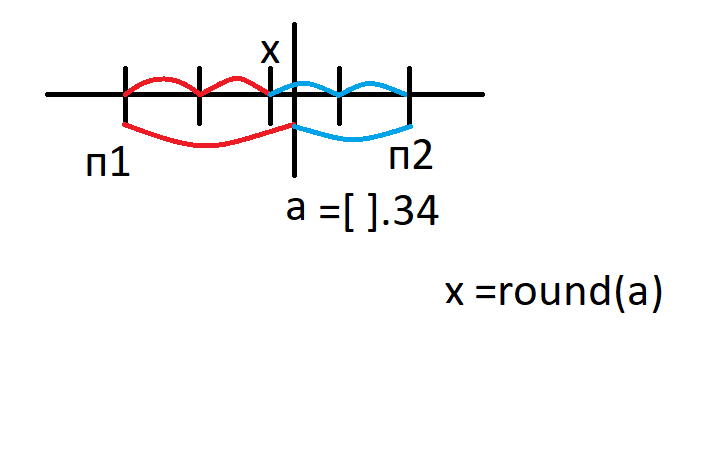
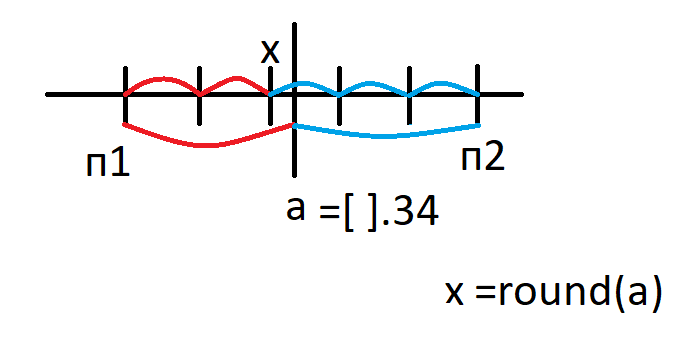
Число будемо вважати простим, якщо воно має хоча б один дільник. Доведено, що будь-яке число a має дільник на інтервалі [2; sqrt(a)]. Отже, можна перебирати дільники лише в цьому діапазоні, а не від 2 до а/2.

Прапорці будуть використані для умов входу або виходу з циклу та означатимуть чи є число простим.

Якщо округлене дійсне число а має два простих числа на однаковій відстані, то виберемо те, до якого ближче дійсне а, тобто потрібно порівняти а з округленим а та на основі результату вибрати найближче до а число.

Прості числа зручно шукати при цілому значенні а, оскільки прості числа є цілими та саме поняття дільника означає ділення націло без остачі. Тому якщо введено дійсне а, воно буде переведене у ціле. А після підрахунку з врахуванням округлення буде виведено найближче число.

Приклад:

Результатом має бути п2. Результатом має бути п1.

Прості числа п1 та п2 шукаємо відносного цілого х, що дорівнює окрегленому а.

**3. Програмні специфікації запишемо у псевдокоді та графічній формі у вигляді блок-схеми.**

Крок 1. Визначимо основні дії.

Крок 2. Деталізуємо пошук найближчого простого числа при а < 3.

Крок 3. Деталізуємо пошук найближчого простого числа, якщо округлене а є простим числом.

Крок 4. Деталізуємо пошук найближчого простого числа, якщо a >= 3 та округлене а не є простим числом.

*крок 1*

**початок**

Пошук найближчого простого числа при а < 3.

Пошук найближчого простого числа, якщо округлене а є простим числом.

Пошук найближчого простого числа, якщо a >= 3 та округлене а не є простим числом.

**кінець**

**Псевдокод**

*крок 3*

**початок**

**введення а**

**якщо** a < 3

**то**

**виведення** «3»

**інакше**

z = true

i = 2

**якщо** (round(a) == a)

**то**

z = false

**все якщо**

**повторити поки** z && i <= sqrt(a)

**якщо** int(round(a)) % i == 0

**то**

z = false;

**все** **якщо**

i++

**все повторити**

Пошук найближчого простого числа, якщо a >= 3 та округлене а не є простим числом.

**якщо**  z = true

**то**

**виведення** int(round(a))

**все якщо**

**все якщо**

**кінець**

*крок 2*

**початок**

**введення а**

**якщо** a < 3

**то**

**виведення** «3»

**інакше**

Пошук найближчого простого числа, якщо округлене а є простим числом.

Пошук найближчого простого числа, якщо a >= 3 та округлене а не є простим числом.

**все якщо**

**кінець**

**якщо** after\_a <= 1

**то** p = false

**все якщо**

**якщо** before\_a <= 1

**то** q = false

**все якщо**

**поки** !p && !q

**все повторити**

**якщо** (p && q)

**то**

**якщо** a == round(a)

**то**

**виведення** before\_a, after\_a

**інакше**

**якщо** a > round(a)

**то** **виведення** after\_a

**інакше виведення** before\_a

**все якщо**

**все якщо**

**інакше**

**якщо** p = true

**то виведення** after\_a

**інакше виведення** before\_a

**все** **якщо**

**все якщо**

**якщо**  z = true

**то**

**виведення** int(round(a))

**все якщо**

**все якщо**

**кінець**

*крок 4*

**початок**

**введення а**

**якщо** a < 3

**то**

**виведення** «3»

**інакше**

z = true

i = 2

**якщо** (round(a) == a)

**то**

z = false

**все якщо**

**повторити поки** z && i <= sqrt(a)

**якщо** int(round(a)) % i == 0

**то**

z = false;

**все** **якщо**

i++

**все повторити**

**якщо** z = false

**то**

after\_a = int(round(a))

before\_a = after\_a

**повторити**

after\_a++;

i = 2;

p = true;

**повторити**

**якщо** after\_a % i == 0

**то** p = false

**все якщо**

i++

**поки** p && i <= sqrt(after\_a)

**все повторити**

before\_a--;

i = 2;

q = true

**повторити**

**якщо** before\_a % i == 0

**то** q = false

**все якщо**

і++

**поки** q && i <= sqrt(before\_a)

**все повторити**

**Блок-схема**



Крок 1

Крок 2



Крок 3





Крок 4

**4. Перевіримо правильність алгоритму на довільних конкретних значеннях початкових даних:**

|  |  |
| --- | --- |
| Блок | Дія |
|  | Початок |
| 1 | Введення а = 12,4 |
| 2 | !(a < 3) |
| 3 | z = true, i = 2 |
| 4 | !(round(a) == a) 12,4 != 12 z == true |
| 5 | **повторити поки** z && i <= sqrt(a)  **якщо** int(round(a)) % i == 0  **то**  z = false  **все** **якщо**  i++  **все повторити**  z == false |
| 6 | (z == false) |
| 7 | after\_a = int(round(a)) = 12  before\_a = after\_a = 12 |
| 8 | **повторити**  after\_a++  i = 2  p = true  **повторити**  **якщо** after\_a % i == 0  **то** p = false  **все якщо**  i++  **поки** p && i <= sqrt(after\_a)  **все повторити**  before\_a--  i = 2  q = true  **повторити**  **якщо** before\_a % i == 0  **то** q = false  **все якщо**  і++  **поки** q && i <= sqrt(before\_a)  **все повторити**  **якщо** after\_a <= 1  **то** p = false  **все якщо**  **якщо** before\_a <= 1  **то** q = false  **все якщо**  **поки** !p && !q  **все повторити**  **p = true, q = true** |
| 9 | (p && q) |
| 10 | !(a == round(a)) **12,4 != 12** |
| 11 | (a > round(a)) 12,4 > 12 |
| 12 | !(z == true) z == false |
| 13 | Вивід: 13 (after\_a). |

|  |  |
| --- | --- |
| Блок | Дія |
|  | Початок |
| 1 | Введення а = 11,2 |
| 2 | !(a < 3) |
| 3 | z = true, i = 2 |
| 4 | !(round(a) == a) 11,2 != 11 z == true |
| 5 | **повторити поки** z && i <= sqrt(a)  **якщо** int(round(a)) % i == 0  **то**  z = false  **все** **якщо**  i++  **все повторити**  z == true |
| 6 | !(z == false) |
| 7 | (z == true) |
| 8 | Вивід: 11 ( round(a) ). |

|  |  |
| --- | --- |
| Блок | Дія |
|  | Початок |
| 1 | Введення а = 15 |
| 2 | !(a < 3) |
| 3 | z = true, i = 2 |
| 4 | (round(a) == a) 15 == 15: z = false |
| 5 | z == false  *(цикл не виконується)*  **повторити поки** z && i <= sqrt(a)  **якщо** int(round(a)) % i == 0  **то**  z = false  **все** **якщо**  i++  **все повторити** |
| 6 | (z == false) |
| 7 | after\_a = int(round(a)) = 15  before\_a = after\_a = 15 |
| 8 | **повторити**  after\_a++  i = 2  p = true  **повторити**  **якщо** after\_a % i == 0  **то** p = false  **все якщо**  i++  **поки** p && i <= sqrt(after\_a)  **все повторити**  before\_a--  i = 2  q = true  **повторити**  **якщо** before\_a % i == 0  **то** q = false  **все якщо**  і++  **поки** q && i <= sqrt(before\_a)  **все повторити**  **якщо** after\_a <= 1  **то** p = false  **все якщо**  **якщо** before\_a <= 1  **то** q = false  **все якщо**  **поки** !p && !q  **все повторити**  **p = true, q = true** |
| 9 | (p && q) |
| 10 | (a == round(a))15 == 15 |
| 11 | !(z == true) z == false |
| 12 | Вивід: 13 і 17 (after\_a and before\_a). |

|  |  |
| --- | --- |
| Блок | Дія |
|  | Початок |
| 1 | Введення а = 14,3 |
| 2 | !(a < 3) |
| 3 | z = true, i = 2 |
| 4 | !(round(a) == a) 14,3 !=14 z == true |
| 5 | **повторити поки** z && i <= sqrt(a)  **якщо** int(round(a)) % i == 0  **то**  z = false  **все** **якщо**  i++  **все повторити**  z == false |
| 6 | (z == false) |
| 7 | after\_a = int(round(a)) = 14  before\_a = after\_a = 14 |
| 8 | **повторити**  after\_a++  i = 2  p = true  **повторити**  **якщо** after\_a % i == 0  **то** p = false  **все якщо**  i++  **поки** p && i <= sqrt(after\_a)  **все повторити**  before\_a--  i = 2  q = true  **повторити**  **якщо** before\_a % i == 0  **то** q = false  **все якщо**  і++  **поки** q && i <= sqrt(before\_a)  **все повторити**  **якщо** after\_a <= 1  **то** p = false  **все якщо**  **якщо** before\_a <= 1  **то** q = false  **все якщо**  **поки** !p && !q  **все повторити**  **p = false, q = true** |
| 9 | !(p && q) |
| 10 | !(p == true) p == false |
| 11 | Вивід: 13 (before\_a). |

|  |  |
| --- | --- |
| Блок | Дія |
|  | Початок |
| 1 | Введення а = 1,3 |
| 2 | (a < 3) 1,3 < 3 |
| 3 | Вивід: 3 (“3”). |

**Висновок:** побудовано алгоритм розв’язання поставленої задачі, що включає послідовно поєднані цикл з передумовою та цикл з постумовою, який мітить два вкладені в нього цикли з постумовою. Цикл з постумовою використовуємо, коли дії потрібно виконати хоча б один раз, а з передумовою – коли дії відбуваються лише за виконання умови. Перевірено правильність виконання для кожної гілки розгалуження а = 12,4; 11,2; 15; 14,3.