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 №5 з дисципліни

«Алгоритми структури даних»

«Дослідження складних циклічних алгоритмів »

Варіант 34

Виконав студент ІП-1134 Шамков Іван Дмитрович

( прізвище, ім'я, по батькові)

Перевірив викладач Мартинова Оксана Петрівна

( прізвище, ім'я, по батькові)

Київ 2021

Лабораторна робота №5

Дослідження складних циклічних алгоритмів

Лабораторна робота 5

Дослідження складних циклічних алгоритмів

*Мета* –дослідити особливості роботи складних циклів та набути практичних навичок їх використання під час складання програмних специфікацій.

Варіант: 34

*Умова задачі:*



*Математична модель:*

| Змінна | Тип | Ім’я | Призначення |
| --- | --- | --- | --- |
| Кількість простих чисел | Цілий | N | Початкове дане |
| Число, яке перевіряємо | Цілий | num\_now | Проміжне значення |
| Число, на яке ділимо | Цілий | dil | Проміжне значення |
| Кількість уже виведених простих чисел | Цілий | kilk | Проміжне значення |
| Збережене значення кількості виведених простих чисел | Цілий | kilk1 | Проміжне значення |
| Логічна змінна для перевірки умови ділення націло | Логічний | Bool | Проміжне значення |

Постановка задачі:

Отже, математичне формулювання нашої задачі полягає в тому, щоб отримати значення n, яке є кількістю простих чисел, що нам потрібно вивести. Через арифметичний цикл ми пробігаємо по значенням непарних чисел, які перевіряємо, чи є вони простими. Для кожного такого числа працює ітераційний цикл, який перевіряє дві умови:

1. Чи ділиться число націло?
2. Чи є дільник меншим за остачу від ділення?

Якщо перша умова ні одного разу не виконується для перевіряємого числа, то рано чи пізно виконається друга умова. У такому випадку наше число буде простим. Усе через те, що ми перевірили усі можливі варіанти утворення числа. Розглянемо це на прикладі числа 30 та простого числа 131:

30/2=15; 30/3=10; 30/5=6; 30/6=5; 30/10=3; 30/15=2;

Як бачимо, дільники та повторилися з результатами від ділення. Числа 5 та 6 є перехідною точкою. Так, щоб утворити число 30, достатньо помножити деяке число з проміжку (0;6) на інше певне число з проміжку [6;30]. Це працює в обидві сторони.

Тепер розглянемо число 131. Перевіримо, чи буде воно простим.

131/3=43.6 3<43.6

131/5=26.2 5<26.2

131/7=18.7 7<18.7

131/9=14.5 9<14.5

131/11=11.9 11<11.9

131/13=10.07 13>10.07

Виконавши ці дії ми точно можемо сказати, що 131 - просте число, адже воно може утворитися з чисел, одне з яких належить проміжку (0;13), а інше - 13;131. Серед першого проміжку ми не зустрілили ні одного цілого числа, що поділило б 131 націло.

*Псевдокод:*

Крок 1. Визначимо основні дії.

Крок 2. Деталізуємо значення N.

Крок 3. Перевіряємо, чи N==1.

Крок 4. Перевіряємо, чи N >=2.

Крок 5.Знаходимо прості числа.

**Крок 1:**

Почато

Деталізуємо значення N

Перевірка значення N

Знаходимо прості числа

Кінець

**Крок 2:**

Початок

Введення N

Перевірка значення N

Знаходимо прості числа

Кінець

**Крок 3:**

Початок

Введення N

**якщо N==1**

то

kilk=1

Виведення 2

Перевірка, чи N>=2

**все якщо**

Знаходимо прості числа

Кінець

**Крок 4:**

Початок

Введення N

**якщо N==1**

то

kilk=1

Виведення 2

**інакше якщо N>=2**

то

kilk=2

Виведення 2 та 3

**все якщо**

Знаходимо прості числа

Кінець

**Крок 5:**

Початок

Введення N

**якщо N==1**

то

kilk=1

Виведення 2

**інакше якщо N>=2**

то

kilk=2

Виведення 2 та 3

**все якщо**

**Для num\_now від 5 з кроком 2 поки kilk!=n**

dil=3

Bool=True

kilk1=kilk

**поки Bool та kilk==kilk1**

то

**якщо num\_now/dil != num\_now//dil**

то

Bool=False

**інакше якщо num\_now/dil<dil**

то

kilk+=1

Виведення num\_now

**інакше**

то

dil+=2

**все якщо**

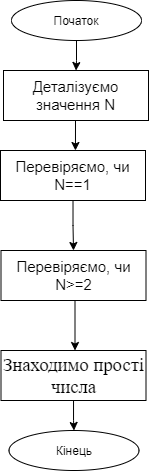
**все повторити**

**все повторити**

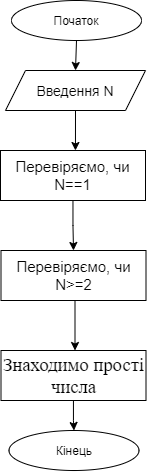
Кінець

*Блок схема:*

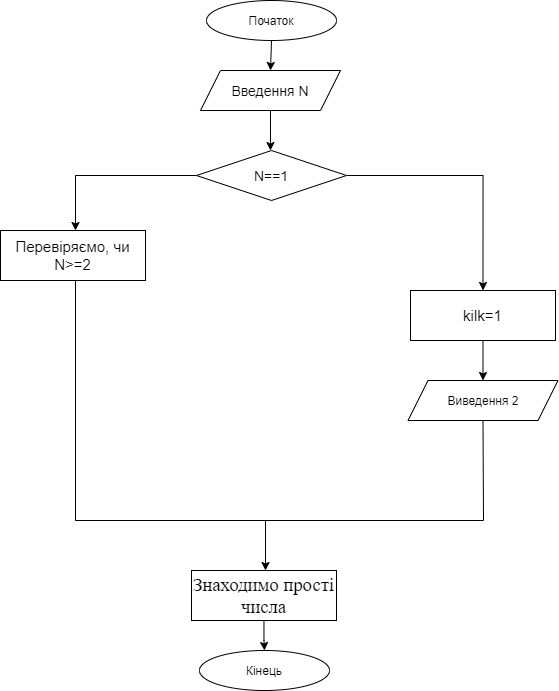
Крок 1



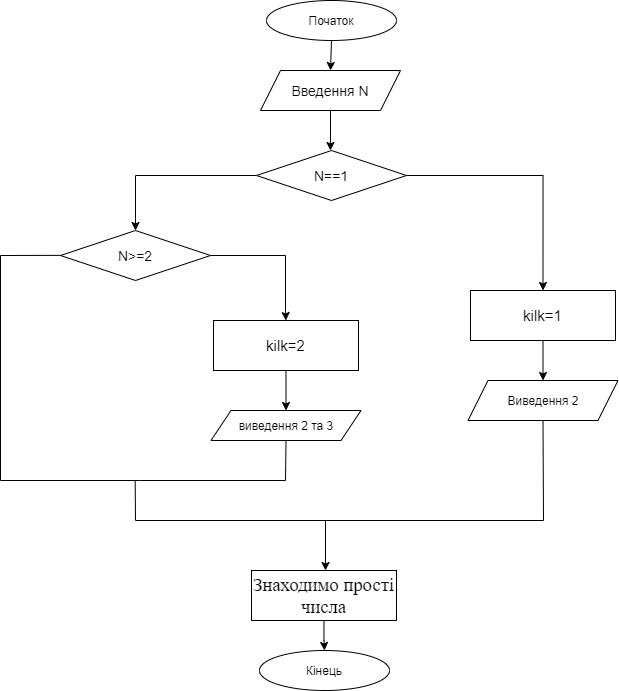
Крок 2



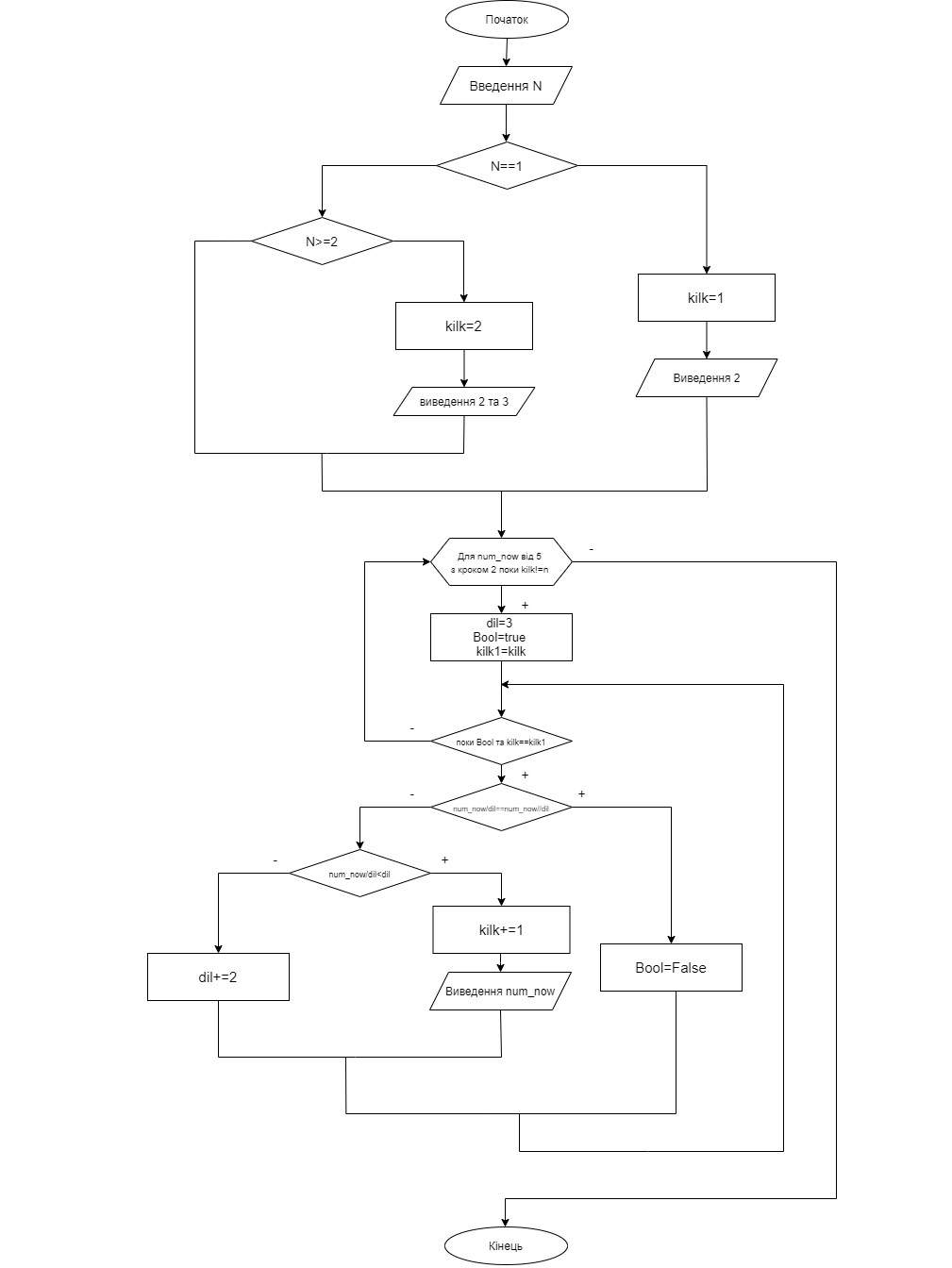
Крок 3



Крок 4



Крок 5

**

*Випробування алгоритму*

| Блок | Дія |
| --- | --- |
|  | Початок |
| 1 | Введення N=5, kilk=2 |
| 2 | Цикл: для num\_now від 5 з кроком 2 поки kilk!=N |
| 3 | Bool=true, dil=3, kilk1=kilk |
| 4 | Цикл: поки Bool та kilk1 == kilk=2 |
| 5 | Якщо num\_now/dil== num\_now//dil: 5/3 не дорівнює 5//3 |
| 6 | Якщо num\_now/dil<dil: 5/3 менше за 3, тому це є просте число  kilk+=1=3 |
| 7 | Виходимо з ітераційного циклу, num\_now+=2=7 |
| 8 | Якщо num\_now/dil== num\_now//dil: 7/3 не дорівнює 7//3 |
| 9 | Якщо num\_now/dil<dil: 7/3 менше за 3, тому це є просте число  kilk+=1=4 |
| 10 | Виходимо з ітераційного циклу, num\_now+=2=9 |
| 11 | Якщо num\_now/dil== num\_now//dil: 9/3 дорівнює 9//3 |
| 12 | Виходимо з ітераційного циклу, num\_now+=2=11 |
| 13 | Якщо num\_now/dil== num\_now//dil: 11/3 не дорівнює 11//3 |
| 14 | Якщо num\_now/dil<dil: 11/3 не менше за 3, тому це не просте число |
| 15 | dil+=2=5 |
| 16 | Якщо num\_now/dil== num\_now//dil: 11/5 не дорівнює 11//5 |
| 17 | Якщо num\_now/dil<dil: 11/5 менше за 5, тому це просте число  kilk+=1=5 |
|  | Кінець |

*Висновок*

Отже, виконавши цю лабораторну роботу, ми навчилися створювати складні цикли. У процесі виконання ми сформулювали задачу, побудували математичну модель та псевдокод алгоритму, що допомогло нам краще її зрозуміти. Основною частиною алгоритму є перевірка умови простоти числа за допомогою складного циклу та виведення такого числа. Алгоритм працює за принципом перебору непарних чисел та перевірки їхнього ділення на інші непарні числа. Також перевіряємо, чи дільник менший за результат від ділення. Арифметичний цикл використовує лічильник num\_now, який набуває значень непарних чисел до того моменту, поки не виведеться потрібна кількість простих чисел. Ітераційний цикл, що знаходиться в арифметичному, надає змінній dil значення непарних чисел, на які ділиться num\_now. У такий спосіб ми перевіряємо, чи є простим число. Підбираємо лише непарні числа, адже простим і парним числом є тільки двійка. У кінці кожного обрахунку виводимо просте число.