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 16

Виконав студент ІП-12 Ковинєв Владислав Андрійович

Перевірив \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(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.

Задача. Дано натуральне число n. Отримати всі прості дільники цього числа.

**I.** Постановка задачі. Розв`язати задачу може рекурсивна підпрограма, яка буде шукати перший дільник заданого числа, виведе його та поділить число на цей дільник. Якщо цей дільник поділить число не остаточно, то підпрограма викличе саму себе з більшим дільником. Так, ця функція знайде всі числа, які беруть участь у остаточному діленні числа.

**II.** Побудова математичної моделі. Складемо таблицю імен даних



Таким чином, математичне формулювання завдання зводиться до рекурсивної підпрограмизнаходження всіх дільників числа та їх виведення .

*III. Розв’язання*

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

*Крок 1. Визначимо основні дії*

*Крок 2. Деталізуємо дію вводу значень*

*Крок 3. Деталізуємо перевірку на те, чи може бути число поділеним*

*Крок 4. Деталізуємо виклик підпрограми Finding(number, 2) чи дію виведення причини виходу з програми*

*Підпрограма Finding(number, divider)*

*Крок 1. Деталізуємо пошук дільника*

*Крок 2. Деталізуємо виведення цього дільника*

*Крок 3. Деталізуємо ділення числа на цей дільник*

*Крок 4. Деталізуємо перевірку на те, чи менше число тотожного дільника*

*Крок 5. Деталізуємо дію виходу з підпрограми чи викликаємо підпрограму знову*

***Псевдокод***

***крок 1***

**початок**

**Введення значень**

Перевірка умови

Виклик підпрограми

Інакше

Виведення причини виходу з програми

Все якщо

**кінець**

***крок 2***

**початок**

**Введення number**

Перевірка умови

Виклик підпрограми

Інакше

Виведення причини виходу з програми

Все якщо

**кінець**

***крок 3***

**початок**

**Введення number**

Якщо number > 1

Виклик підпрограми

Інакше

Виведення причини виходу з програми

Все якщо

**кінець**

***крок 4***

**початок**

**Введення number**

Якщо number > 1

Виклик Finding(number, 2)

Інакше

Виведення причини виходу з програми

Все якщо

**кінець**

**Підпрограма Finding(number, divider)**

**начало**

Поки number % divider != 0

divider++

Все поки

**Виведення divider**

Робити поки number % divider == 0

number /= divider

Все робити

Якщо number <= divider

Вихід з підпрограми

Інакше

Виклик Finding(number, divider+1)

Все якщо

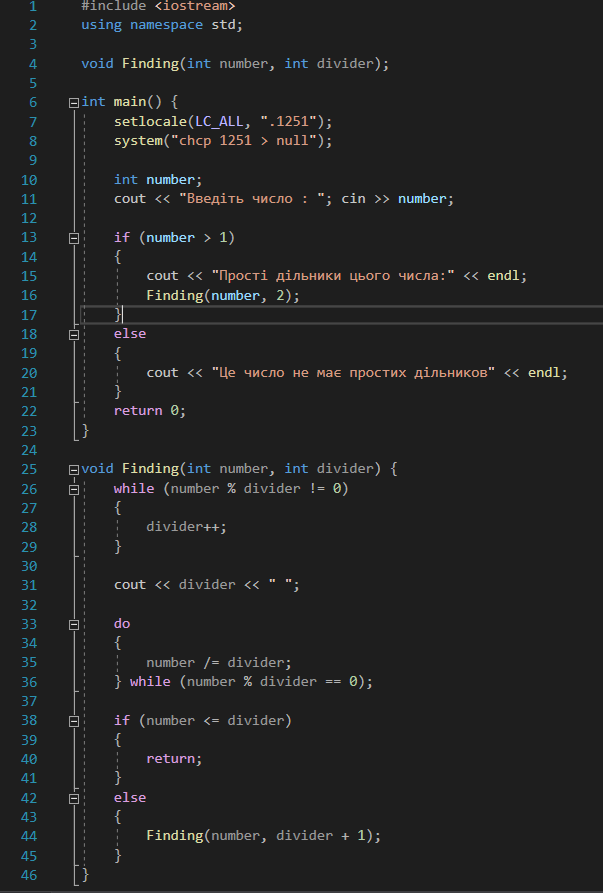
**кінець**

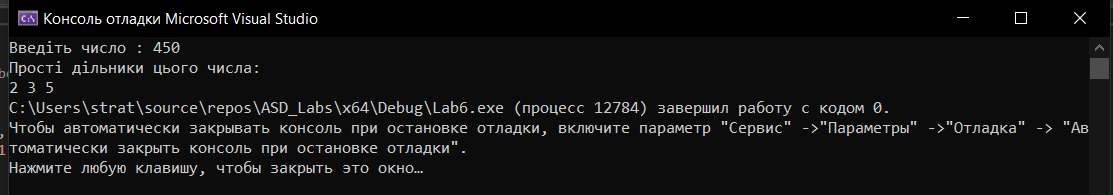








**IV.** Випробування алгоритму



**V.** Висновок

Було досліджено особливості роботи рекурсивних алгоритмів та набути практичних навичок їх використання під час складання програмних специфікацій підпрограм. В результаті виконання лабораторної роботи було знайдено прості дільники числа, розділивши задачу на 2 кроки: визначення основних дій, деталізація розрахування та виведення простих дільників числа за допомогою рекурсивної функції.