# 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 лінійних алгоритмів» Варіант 2

Виконав студент Боровков Іван Ігорович

(шифр, прізвище, ім'я, по батькові)

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.

**Умова задачі** – Обчислення добутку елементів геометричної прогресії, що убуває: початкове значення – 64, кінцеве значення – 1, крок – 4

**Постановка задачі**

За допомогою рекурсивного алгоритму визначити добуток елементів геометричної прогресії з заданим початковим та кінцевим значенням та кроком прогресії.

**Математична модель.**

**Складемо таблицю імен змінних:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Змінна** | **Тип** | **Ім’я** | **Призначення** |
| Початкове значення | Цілий | first\_value | Вхідні дані |
| Кінцеве значення | Цілий | last\_value | Вхідні дані |
| Крок | Цілий | step | Вхідні дані |
| Добуток прогресії | Цілий | result | Вихідні дані |
| Значення актуального елемента прогресії | Цілий | relevant\_value | Допоміжна змінна |

Використаємо рекурсію. Термінальна гілка рекурсії виконається за умови рівності значення актуального елемента прогресії та кінцевого значення. В цьому випадку функція поверне значення цього елемента без змін. В іншому випадку завжди виконуватиметься рекурсивна гілка. В гілці міститиметься рекурсивний виклик даної функції, аргументом якої (оскільки прогресія спадна) має бути актуальне значення елемента прогресії, розділене на крок прогресії. В такому випадку функція поверне добуток значення, що поверне рекурсивний виклик на актуальне значення елемента прогресії.

***Розв’язання:***

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

*Крок 1:* Визначимо основні дії ;

*Крок 2:* Задамо значення початкового та кінцевого значень та кроку прогресії;

*Крок 3:* Обчислимо добуток прогресії;

*Псевдокод*

*Крок 1*

**Початок**

Задамо значення початкового та кінцевого значень та кроку прогресії;

Обчислимо добуток прогресії;

**Кінець.**

*Крок 2*

**Початок**

first\_value = 64;

last\_value = 1;

step = 4**;**

Обчислимо добуток прогресії;

**Кінець.**

*Крок 3*

**Початок**

first\_value = 64;

last\_value = 1;

step = 4;

result = product(first\_value, last\_value, step);

**Кінець.**

**product** (relevant\_value, last\_value, step)

**Початок**

**Якщо (**relevant\_value == last\_value)

**То**

**Повернути** last\_value;

**Інакше**

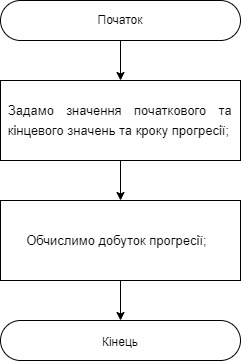
**Повернути (product** (relevant\_value / step, last\_value, step) \* last\_value);

**Все якщо;**

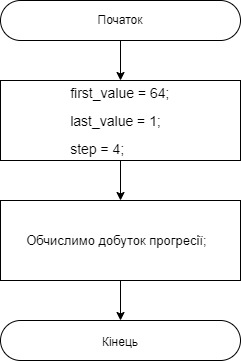
**Кінець.**

*Блок-схема:*

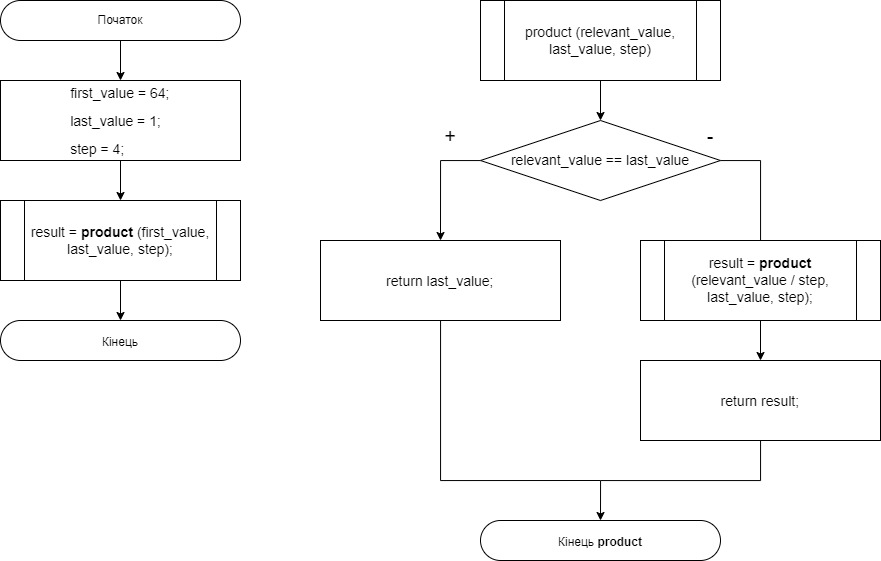
*Крок 1*

**

*Крок 2*

**

*Крок 3*

**

**Випробування алгоритму:**

|  |  |
| --- | --- |
| **Блок** | **Дія** |
|  | Початок |
| **1.** | Product(64, 1, 4) |
| **2.** | 64 ≠ 1, тому  Product(16, 1, 4) |
| **3.** | 16 ≠ 1, тому  Product(4, 1, 4) |
| **4.** | 4 ≠ 1, тому  Product(1, 1, 4) |
| **5.** | 1 = 1, тому  Повертаємо 1. |
| **6.** | Повертаємо 4\*1 = 4 |
| **7.** | Повертаємо 16\*4 = 64 |
| **8.** | Повертаємо 64\*64 = 4096 |
|  | Добуток заданої геометричної прогресії = 4096, отже алгоритм працює правильно. |
|  | Кінець |

**Код програми:**

**#include <iostream>**

using namespace std;

double product(double, double, double);

int main()

{

double first\_value = 64, last\_value = 1, step = 4, result;

result = product(first\_value, last\_value, step);

cout << "Product is: " << result;

return 1;

}

double product(double relevant\_value, double last\_value, double step)

{

if (relevant\_value == last\_value) return last\_value; //Термінальна гілка рекурсії:

//Повертаємо значення останнього елементу прогресії

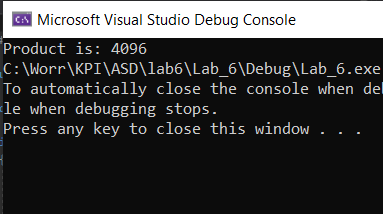
else return product((relevant\_value / step), last\_value, step) \* relevant\_value; //Рекурсивна гілка рекурсії:

//Повертаємо значення добутку всіх наступних елементів прогресії

// \* на теперішній аргумент

**}**

*Копії екранних форм:*



**Висновок:** В ході лабораторної роботи я навчився використовувати рекурсію, зображати рекурсивні алгоритми графічно та у вигляді псевдокоду, обчислювати за їх допомогою добуток геометричної прогресії.