Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний
інститут імені І. Сікорського»
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт
з лабораторної роботи № 6
«Алгоритми та структури даних-1.
Основи алгоритмізації»

«Дослідження рекурсивних алгоритмів»

Варіант 2

Виконав студент: ІП-13 Бабашев Олексій Дмитрович

Перевірив: Вечерковська Анастасія Сергіївна

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.

Варіант 2

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

1) **Постановка задачі:** Обчислити добуток послідовно виділених елементів геометричної прогресії. При початковому елементі 64 і кінцевому 1, з кроком 4, убуваюча прогресія.
2) **Побудова математичної моделі:**

Змінна	**Тип**	**Назва**	**Призначення**
Початкове значення геом. прог.	Цілий	bfirst	Вхідне дане
Кінцеве значення геом. прог.	Цілий	blast	Вхідне дане
Знаменник геом. прог.	Цілий	q	Вхідне дане
Результат добутку елементів геом. прог.	Цілий	res	Вихідне дане
Параметр підпрограми початковий елемент прогресії	Дійсний	first	Параметр функції
Параметр підпрограми знаменник прогресії	Дійсний	step	Параметр функції
Параметр підпрограми кінцевий елемент прогресії	Дійсний	last	Параметр функції
Добуток	Цілий	value	Проміжне значення

Математичне формулювання задачі зводиться до виділення елементів геометричної прогресії і обчислення їх добутку за допомогою виклику рекурсивної функції. Рекурсивна функція: product_elem(first , step, last)

first – початковий, step – крок, last – кінцевий. Для виділення усіх елементів даної прогресії у функції викликається сама функція з початком елементом зменшеним у step разів product_elem(first/step , step, last). Також всередені функції для обчислення добутку елементів обчислюємо добуток: first* product_elem(first/step , step, last) після чого функція повертає значення value. Так як прогресія спадна, то функція буде викликати саму себе доки умова first <= last не виконується. Коли умова виконається value = 1. Після остаточного обчислення з функції повертається значення value і присвоюється до значення res.

Розв’язання:
Крок 1. Визначимо основні дії.
Крок 2. Деталізуємо виклик підпрограми.
Крок 3. Деталізуємо підпрограму.

3) **Псевдокод:**

Крок 1

Початок

Введення bfirst blast q

Виклик підпрограми product_elem

Вивести res

Кінець

Крок 2

функція product_elem(first , step, last)
Реалізація підпрограми
Повернути value
вся функція

Початок
Введення bfirst blast step
res = product_elem(bfirst , q, blast)
Вивести res
Кінець

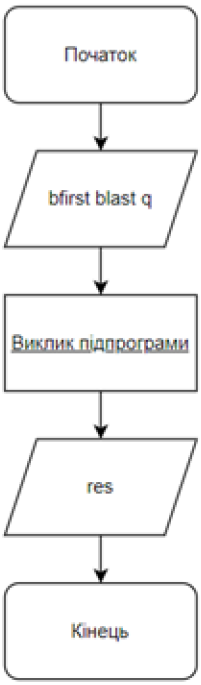
Крок 3

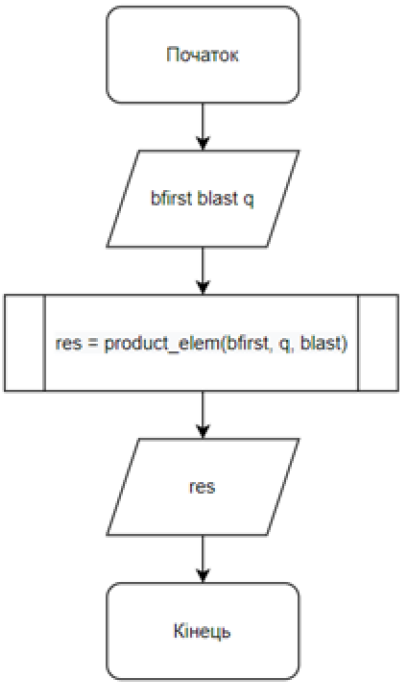
функція product_elem(first , step, last)
якщо (first <= last)
то
value = 1
інакше
value = first* product_elem(first/step , step, last)
все інакше
все якщо
Повернути value
вся функція

Початок
Введення bfirst blast step
res = product_elem(bfirst , q, blast)
Вивести res
Кінець

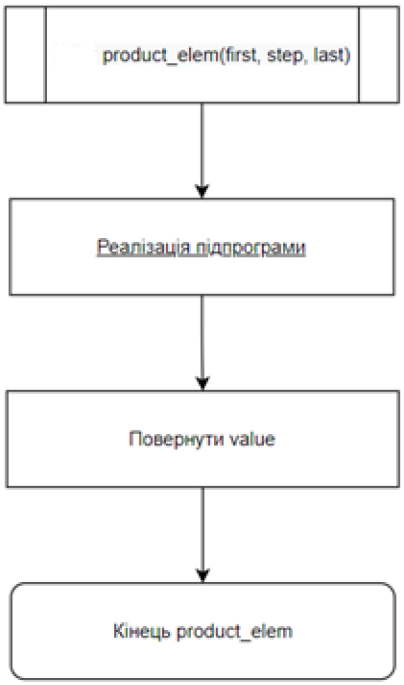
4) Блок-схема:

Крок 1

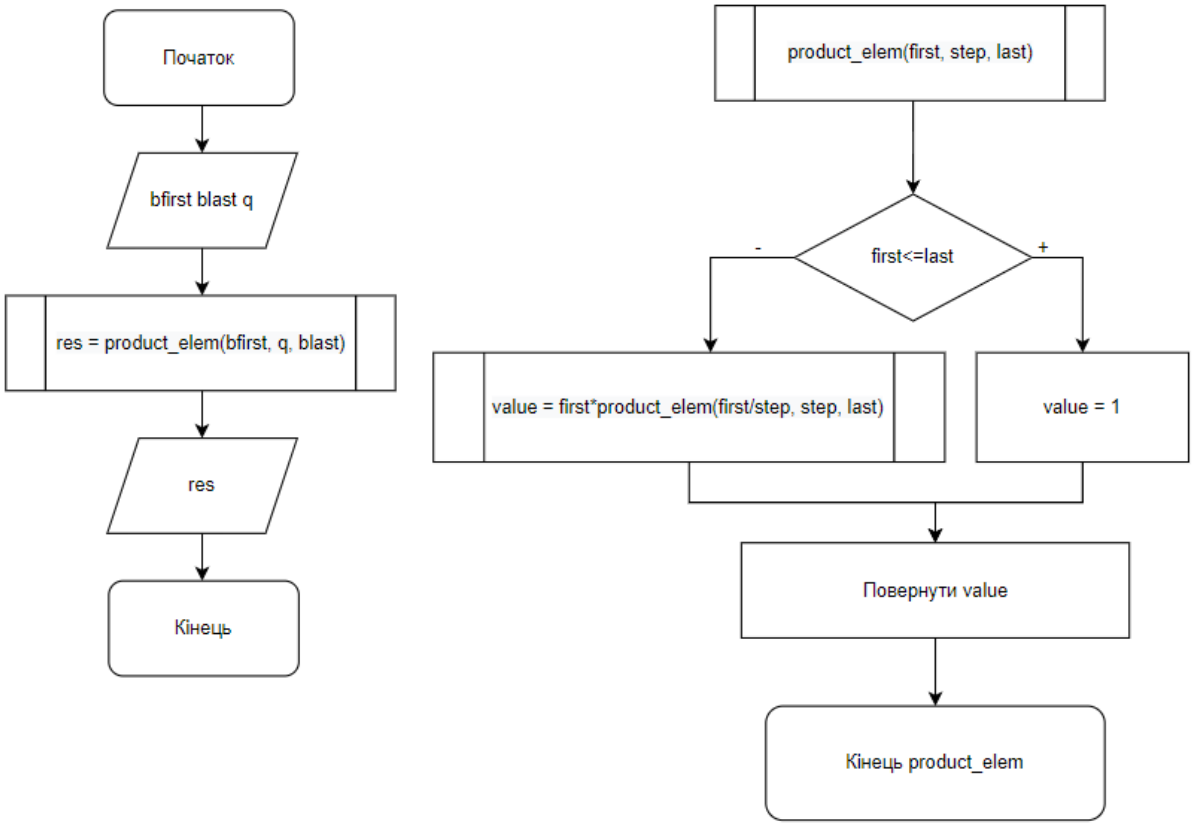




Крок 2



Крок 3



5) Код та результат

onsoleApplication1.cpp

ConsoleApplication1 (Global Scope)

```cpp
#include <iostream>
#include <cmath>

 using namespace std;

int product_elem(int first, int step, int last)
{
    int value;
    if (first <= last)
    {
    value = 1;

    }
    else {
    value = first * (product_elem(first / step, step, last));

    }

    return value;
}

int main()
{
    int bfirst, blast, q, res;
    cout << "enter first number,last number and step" << endl;
    cin >> bfirst;
    cin >> blast;
    cin >> q;
    res = product_elem(bfirst, q, blast);
    cout << "result equals= " << res;
}
```

```
enter first number,last number and step
64
1
4
result equals= 4096
```

6) Випробування алгоритму:

Блок	Дія
	Початок
1	Введення 64 1 4
2	Виклик функції product_elem(64 , 4, 1)
3	64<=1(ні) тому value = 64* product_elem(16 , 4, 1)
4	16<=1(ні) тому value = 16* product_elem(4 , 4, 1)
5	4<=1(ні) тому value = 4* product_elem(1 , 4, 1)
6	1<=1(так) тому value = 1
7	7) value = 4*1 5) value = 16*4 4) value = 64 * 64 = 4096
8	Повернення значення value від product_elem(64 , 4, 1)
9	res = product_elem(64 , 4, 1)
10	Вивести res
	Кінець

7) Висновок: дослідив особливості роботи рекурсивних алгоритмів та набув практичних навичок їх використання під час складання програмних специфікацій підпрограм.