

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 30

Виконав студент ІІІ-12 Тарасюк Євгеній Сергійович

Перевірив _____

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 30.

Дано перший член і знаменник геометричної прогресії. Обчислити суму n перших членів прогресії та знайти n -й член прогресії.

Розв'язок.

1. Постановка задачі.

Початкові дані - це дійсні числа, додаткових змінних для розв'язку не потрібно. Додаткові умови: $n > 0$ (потрібно розрахувати хоча б один член геометричної прогресії), Для обчислення використовуватимемо рекурсивний цикл. Результатом розв'язку є два дійсних числа. Використовуватимемо стандартні логічні та арифметичні операції, а також функції.

2. Побудова математичної моделі

Таблиця змінних та функцій:

Змінні та функції	Тип	Ім'я	Призначення
b1	Дійсне число	b1	Перший член геометричної прогресії (вхідні дані)
q	Дійсне число	q	Знаменник геометричної прогресії (вхідні дані)
n	Натуральне число	n	Кількість членів прогресії в сумі (вхідні дані)
Сума	Дійсне число	gsum	Проміжне значення суми членів прогресії
Індекс	Натуральне число	ind	Індекс поточного елемента прогресії
Степінь	Функція, що повертає дійсне число	power(a, b)	Функція підведення числа до степеня
Основа показникової функції	Дійсне число	a	Число, що підводиться до степеня
Порядок	Ціле число	b	Порядок числа a
Рекурсивна сума	Функція	sumrec(gsum, b1, q, n, ind)	Рекурсивна функція для розрахунку суми членів геометричної прогресії (усі аргументи функції мають ролі, ідентичні ролям однойменних змінних в основній програмі)

Суму елементів прогресії можна розрахувати, викликаючи функцію для розрахунку нового елемента за формулою $b1 * q^{(ind-1)}$, поки не буде додано достатньо елементів.

3. Псевдокод алгоритму

Початок

$gsum = 0$

$ind = 1$

Введення $b1, q, n$

Якщо $n < 1$

Виведення “n must be > 0”

Інакше

$sumrec(gsum, b1, q, n, ind);$

Все якщо

Кінець.

Функція $sumrec(gsum, b1, q, n, ind)$

Початок

$elem = b1 * power(q, ind - 1)$

Якщо $ind < n$

$sumrec(gsum + elem, b1, q, n, ind + 1)$

Інакше

Виведення $gsum + elem, elem$

Все якщо

Повернення 0

Кінець

Функція $power(a, b)$

Початок

$res = 1$

Повторити для i **на проміжку** $[0; b)$

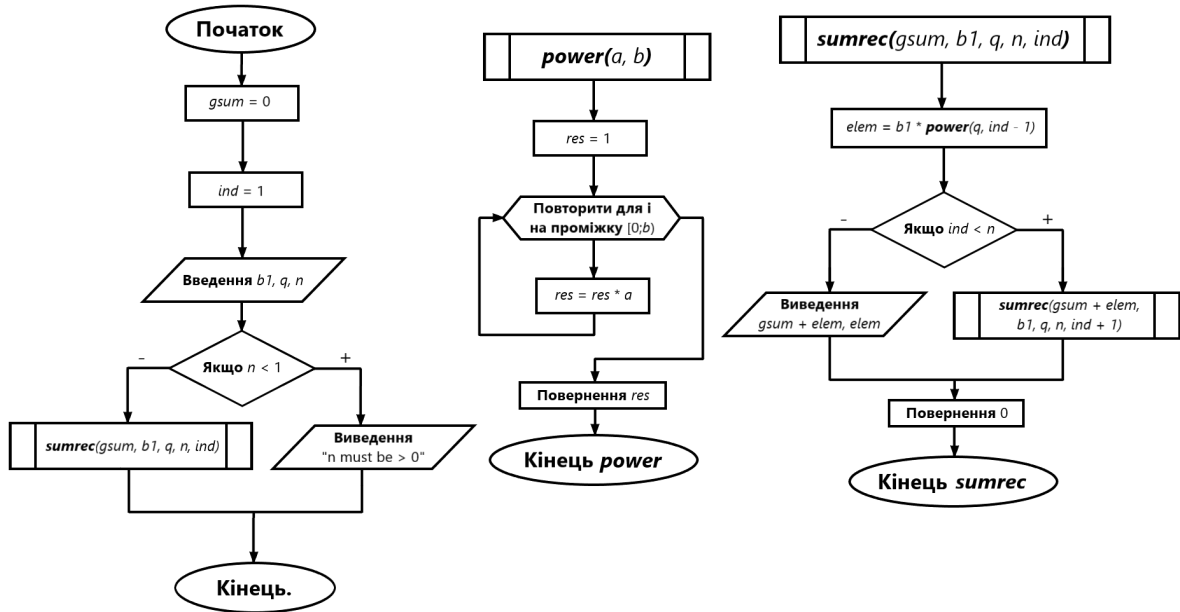
$res = res * a$

Все повторити

Повернення 0

Кінець

4. Блок-схема алгоритму



5. Випробування алгоритму.

Перевіримо правильність алгоритму для різних вхідних даних:

	Тест 1	Тест 2	Тест 3	Тест 4
gsum = 0; ind = 1; Введення <i>bl, q, n</i>	<i>bl</i> = 3.14 <i>q</i> = 2.71 <i>n</i> = 3	<i>bl</i> = 2 <i>q</i> = 2 <i>n</i> = 0	<i>bl</i> = 2 <i>q</i> = 0 <i>n</i> = 2	<i>bl</i> = 0 <i>q</i> = 2 <i>n</i> = 2
Якщо <i>n</i> < 1	<i>sumrec</i> (0, 3.14, 2.71, 3, 1)	Виведення “n must be > 0”	<i>sumrec</i> (0, 2, 0, 2, 1)	<i>sumrec</i> (0, 0, 2, 2, 1)
Результат виклику функції <i>sumrec</i> 1:	<i>elem</i> = 3.14; <i>ind</i> < <i>n</i> ; <i>sumrec</i> (3.14, 3.14, 2.71, 3, 2)	-	<i>elem</i> = 2; <i>ind</i> < <i>n</i> ; <i>sumrec</i> (2, 2, 0, 2, 2)	<i>elem</i> = 0; <i>ind</i> < <i>n</i> ; <i>sumrec</i> (0, 0, 2, 2, 2)
Виклик функції <i>sumrec</i> 2:	<i>elem</i> = 11.6494; <i>ind</i> < <i>n</i> ; <i>sumrec</i> (11.6494, 3.14, 2.71, 3, 3)	-	<i>elem</i> = 0; не (<i>ind</i> < <i>n</i>); Виведення sum = gsum + <i>elem</i> , <i>elem</i>	<i>elem</i> = 0; не (<i>ind</i> < <i>n</i>); Виведення sum = gsum + <i>elem</i> , <i>elem</i>
Виклик функції <i>sumrec</i> 3:	<i>elem</i> = 23.0605; не (<i>ind</i> < <i>n</i>); Виведення sum = gsum + <i>elem</i> , <i>elem</i>	-	-	-
Консоль після завершення роботи	sum = 34.7099 <i>elem</i> = 23.0605	n must be > 0	sum = 2 <i>elem</i> = 0	sum = 0 <i>elem</i> = 0

6. Код програми (C++)

```

#include <iostream>

using namespace std;

float power(float a, int b)
{
    float res = 1;
    for (int i = 0; i < b; i++)
    {
        res *= a;
    }
    return res;
}

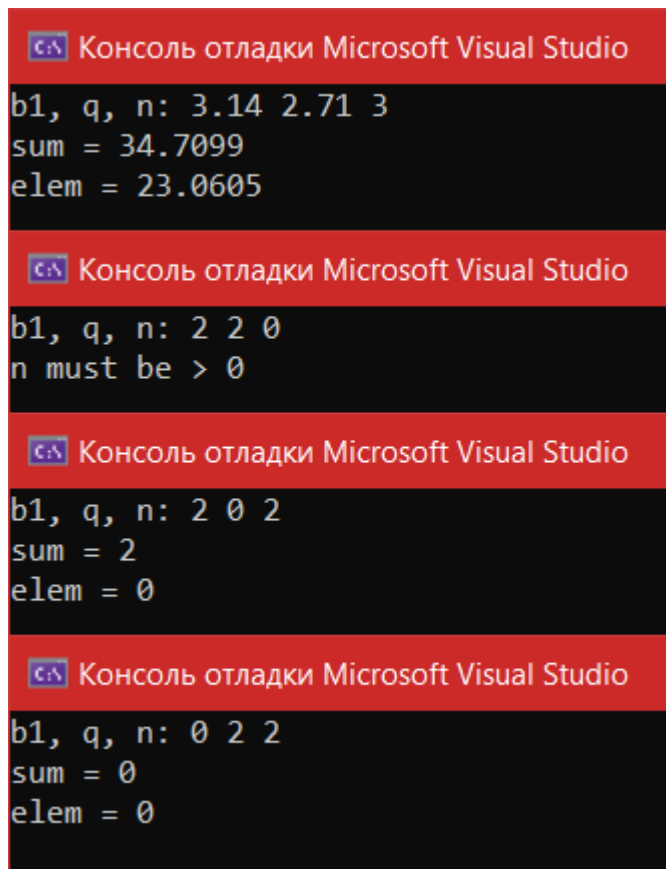
void sumrec(float gsum, float b1, float q, int n, int ind)
{
    int elem = b1 * power(q, ind - 1);
    if (ind < n)
    {
        sumrec(gsum + elem, b1, q, n, ind + 1);
    }
    else
    {
        cout << "sum = " << gsum + elem << endl;
        cout << "elem = " << elem << endl;
    }
    return 0;
}

int main()
{
    float b1, q, gsum = 0;
    int n, ind = 1;
    cout << "b1, q, n: "; cin >> b1 >> q >> n;
    if (n < 1)
    {
        cout << "n must be > 0" << endl;
    }
    else
    {
        sumrec(gsum, b1, q, n, ind);
    }
    return 0;
}

```

7. Випробування коду

Результати випробування для різних вхідних даних



```
Консоль отладки Microsoft Visual Studio
b1, q, n: 3.14 2.71 3
sum = 34.7099
elem = 23.0605

Консоль отладки Microsoft Visual Studio
b1, q, n: 2 2 0
n must be > 0

Консоль отладки Microsoft Visual Studio
b1, q, n: 2 0 2
sum = 2
elem = 0

Консоль отладки Microsoft Visual Studio
b1, q, n: 0 2 2
sum = 0
elem = 0
```

8. Висновки

Було досліджено особливості роботи рекурсивного алгоритму розрахування суми елементів геометричної прогресії та набуто практичних навичок його використання під час складання програмних специфікацій підпрограм. Випробування коду та алгоритму вручну дали однакові результати, що свідчить про правильність роботи програми.