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

Лабораторна робота № 1.2

з дисципліни
«Структури даних і алгоритми»

Виконав: Давидчук А.М.

студент групи ІО-41
Давидчук Артем Миколайович
номер у списку групи: 08

Перевірив:

Сергієнко А. М.

Київ 2024

Завдання:
Задано дійсне натуральне число n. Вирахувати значення заданої формули за своїм варіантом. Для розв'язання цієї задачі написати 2 програми:

1. Перша повинна використовувати для обчислення формули вкладені цикли
2. Друга програма повинна використовувати обчислення формули за допомогою одного циклу з використанням методу динамічного програмування

Також виконати розрахунок кількості операцій кожного алгоритму за методикою, викладеною на лекції, додавши до неї підрахунок кількості викликів стандартних функцій. Результуючі дані вивести у форматі з сімома знаками після крапки.

Мій варіант задачі:

$$8)\quad S = \sinh^2 x = \sum_{i=1}^{n} \frac{2^{2i-1}\, x^{2i}}{(2i)!} \quad \text{при } x = 1/3.$$

Код програми №1 (із вкладеними циклами):

```
#include <stdio.h>

int main() {

    int counter = 0;

    int n;
    double sum = 0;
    double first_value = 0.5;
    double second_value = 1;
    double factorial;

    printf("Enter n (2loops): ");
    scanf("%d", &n);

    counter += 7; // 5 операцій присвоювання та 2 виклика стандартних
функцій

    for (int i = 1; i <= n; i++) {

        first_value *= 4;
        second_value /= 9;
        factorial = 1;

        for (int j = 1; j <= 2*i; j++) {

            factorial *= j;

            counter += 6; //3 арифметичних операцій, 1 операцій
присвоювання, 1 проходження цикла, 1 порівняння
```

```
        }

        sum += first_value * second_value / factorial;

        counter += 2; // присвоювання j=1 та останнє порівняння j > 2*i
        counter += 13; // 6 арифметичних операцій, 4 операцій
присвоювання, 1 проходження цикла, 1 порівняння

    }

    counter += 3; // присвоювання i = 1, фінальне порівняння коли i > n,
фінальний вивід printf

    printf("S = sinh^2(x) = %lf, counter = %d\n", sum, counter);

    return 0;
}
```

Код програми №2 (динамічне програмування):

```
#include <stdio.h>

int main() {

    int counter = 0;

    int n;
    double Sum = 0;
    double first_value = 0.5;
    double second_value = 1;
    double factorial = 1;

    printf("Enter n (Dynamic Programming): ");
    scanf("%d", &n);

    counter += 7; //5 операцій присвоювання та 2 виклика стандартних
функцій

    for (int i = 1; i <= n; i++) {

        first_value *= 4;
        second_value /= 9;
        factorial *= (2*i-1) * 2*i;

        Sum += first_value * second_value / factorial;

        counter += 17; //11 арифметичних операцій, 4 операції
присвоювання, 1 проходження цикла, 1 порівняння

    }

    counter += 3; // присвоєння int i = 1; Ще одне фінальне порівняння
коли i > n; виклик фінального printf

    printf("S = sinh^2(x) = %lf, counter = %d\n", Sum, counter);

    return 0;
}
```

Тестування програм при n = 2, n = 3, n = 4:

Результати програми №1:

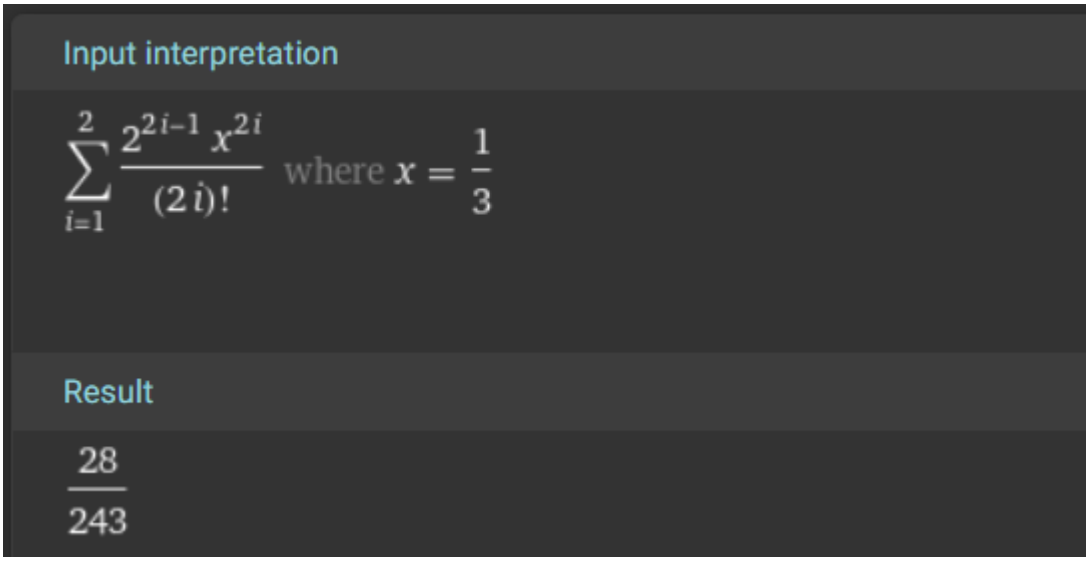
```
Enter n (2loops): 2
S = sinh^2(x) = 0.115226, counter = 76
Enter n (2loops): 3
S = sinh^2(x) = 0.115287, counter = 127
Enter n (2loops): 4
S = sinh^2(x) = 0.115288, counter = 190
```

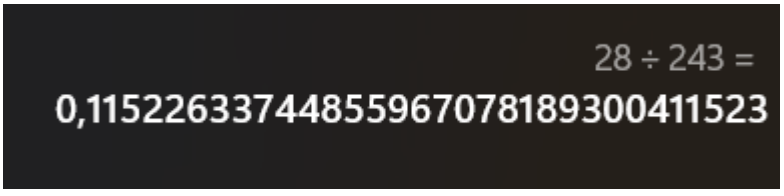
Результати програми №2:

```
Enter n (Dynamic Programming): 2
S = sinh^2(x) = 0.115226, counter = 44
Enter n (Dynamic Programming): 3
S = sinh^2(x) = 0.115287, counter = 61
Enter n (Dynamic Programming): 4
S = sinh^2(x) = 0.115288, counter = 78
```

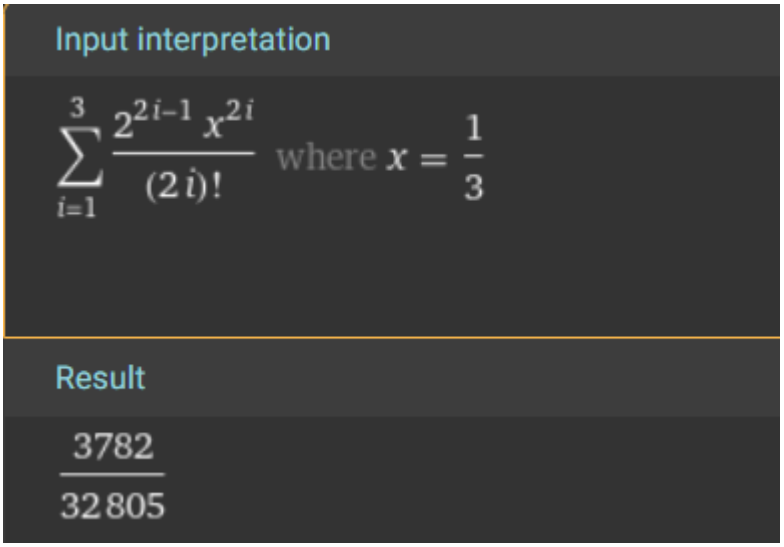
Результати калькулятора (Wolfram|Alpha)

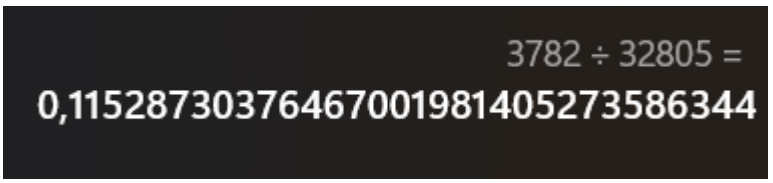
При n = 2:



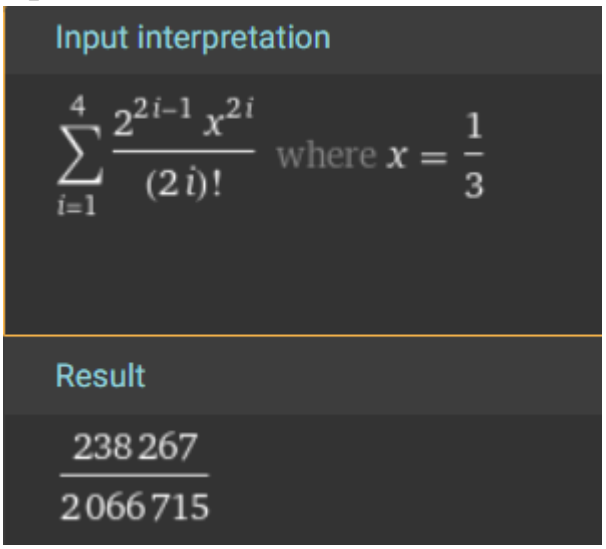


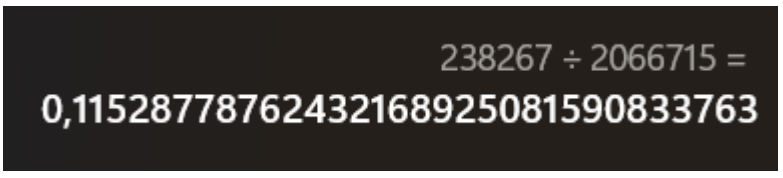
При n = 3:





При n = 4:

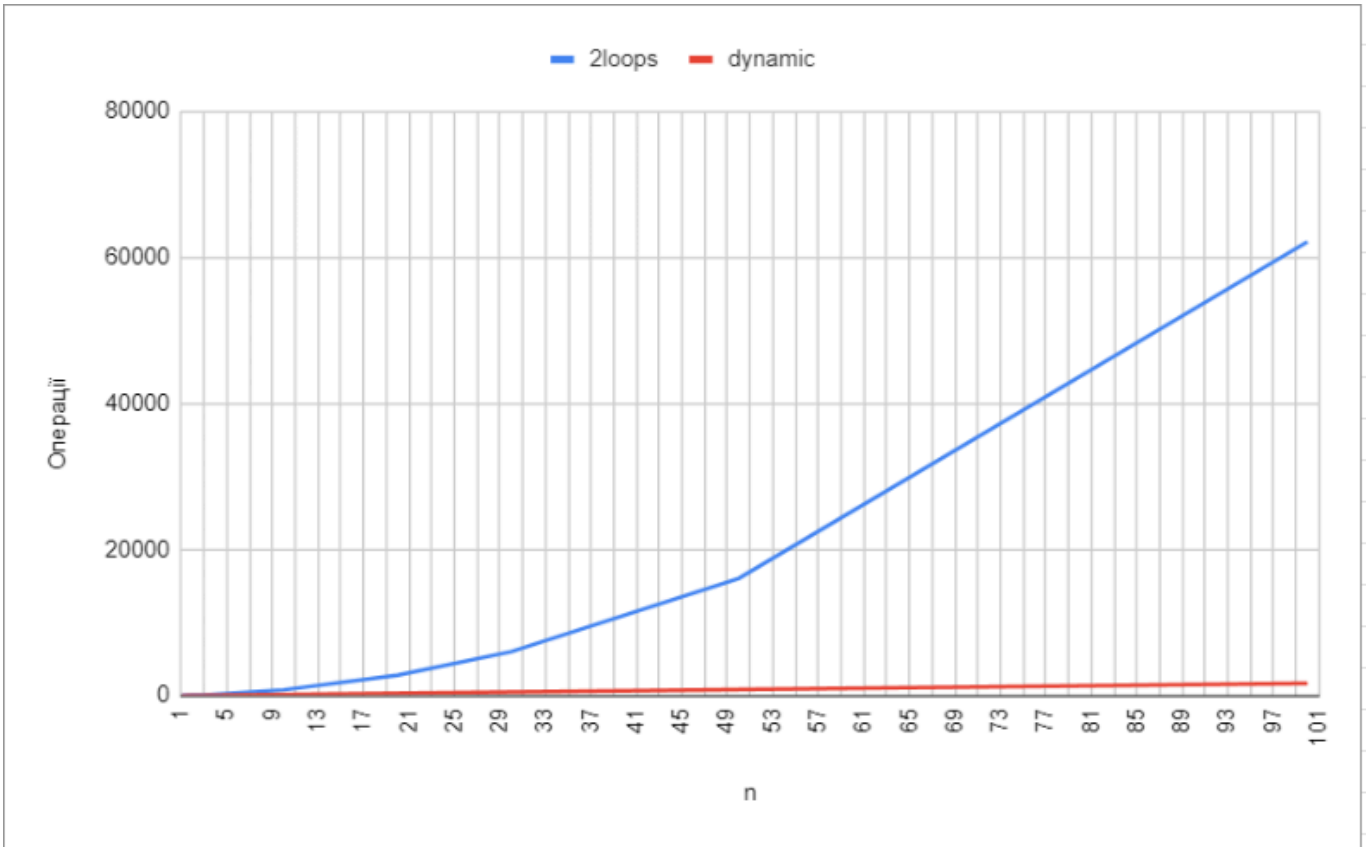




Кількість операцій для двох алгоритмів

n		1	2	3	10	20	30	50	100
Кількість операцій	1 спосіб	37	76	127	820	2830	6040	16060	62110
	2 спосіб	27	44	61	180	350	520	860	1710

Графік залежності кількості операцій від n:



Висновок:

Я зумів представити розрахунок математичної формули у двох різних алгоритмах з різними прийомами його реалізації, а саме за допомогою динамічного програмування та вкладеними циклами. Після аналізу алгоритмів, найшвидшим виявився перший алгоритм (динамічне програмування). Складність такого алгоритму лінійна, а із вкладеними циклами – квадратична. Програми написані правильно, і доказом до цього є результати обчислення онлайн-калькулятора Wolfram | Alpha.