

**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 №2
з дисципліни
«Алгоритми і структури даних»

Виконав:

студент групи ІМ-42
Туров Андрій Володимирович
номер у списку групи: 29

Перевірила:

Молчанова А. А.

Завдання

1. Задане натуральне число **n**. Вирахувати значення заданої формули за варіантом.
2. Для розв'язання задачі написати дві програми:
 - 1) перша програма повинна використовувати для обчислення формули вкладені цикли;
 - 2) друга програма повинна виконати обчислення формули за допомогою одного циклу з використанням методу динамічного програмування.
3. Виконати розрахунок кількості операцій для кожного з алгоритмів за методикою, викладеною на лекції, додавши до неї підрахунок кількості викликів стандартних функцій.
4. Програма має правильно розв'язувати поставлену задачу при будь-якому заданому **n**, для якого результат обчислення може бути коректно представлений типом **double**.
5. Результати вивести у форматі з сімома знаками після коми.

Варіант 29

$$S_n = \sum_{i=1}^n \frac{\prod_{j=1}^i (\ln(j+2))}{3 - \sin^2(i)}$$

Текст програми

A)

```
#include <math.h>
#include <stdio.h>

int main(void) {
    // Operations are: variable assignment, boolean logic, math operations,
    // external math functions,
    // jumps to the next loop iteration
    int op_counter;

    int n;
    scanf("%d", &n);

    double result = 0;
    op_counter = 2; // init of `n`, `result`
    for (int i = 1; i <= n; i++) {
        // `i` init or increment, `i <= n` check, jump
```

```

    op_counter += 3;

    double numerator = 1;
    op_counter += 1; // init of `numerator`

    for (int j = 1; j <= i; j++) {
        // `j` init or increment, `j <= i` check, jump
        op_counter += 3;

        numerator *= log(j + 2); // multiplication, log call, addition
        op_counter += 3;
    }
    // +final `j <= i` check, -unneeded jump on the first iteration

    // addition, division, subtraction, sin call (x2), multiplication
    result += numerator / (3 - sin(i) * sin(i));
    op_counter += 6;
}
// +final `i <= n` check, -unneeded jump on the first iteration (<- comment omitted in b.c)

printf("Result: %.7lf\n", result);
printf("Operations counter: %d\n", op_counter);
}

```

B)

```

#include <math.h>
#include <stdio.h>

int main(void) {
    int op_counter;

    int n;
    scanf("%d", &n);

    double result = 0;
    double last_numerator = 1;
    op_counter = 3; // init of `n`, `result`, `last_numerator`
    for (int i = 1; i <= n; i++) {
        // `i` init or increment, `i <= n` check, jump
        op_counter += 3;

        last_numerator *= log(i + 2); // multiplication, log call, addition
        op_counter += 3;

        result += last_numerator / (3 - sin(i) * sin(i));
        op_counter += 6; // addition, division, subtraction, sin call (x2),
multiplication
    }

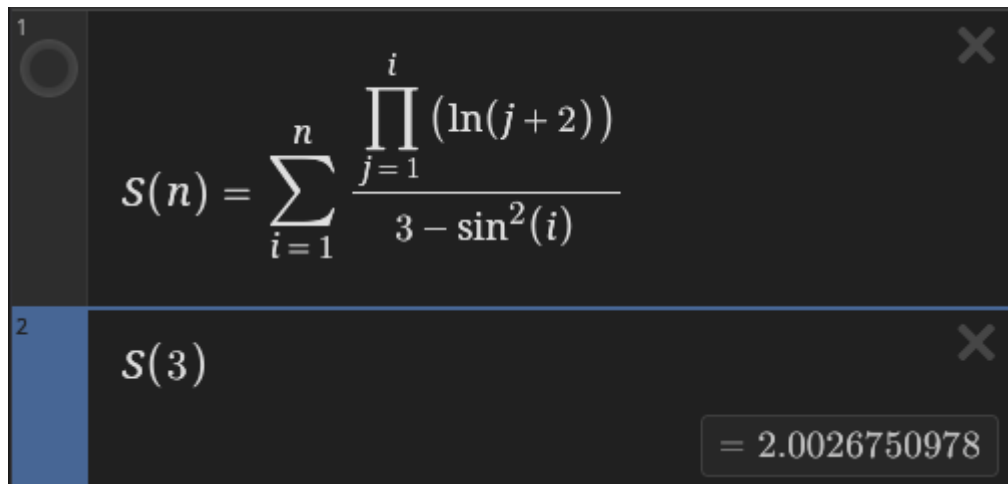
    printf("Result: %.7lf\n", result);
    printf("Operations counter: %d\n", op_counter);
}

```

}

Ручні розрахунки при $n = 3$:

$$S_3 = \frac{\ln(3)}{3 - \sin^2(1)} + \frac{\ln(3) \cdot \ln(4)}{3 - \sin^2(2)} + \frac{\ln(3) \cdot \ln(4) \cdot \ln(5)}{3 - \sin^2(3)} \approx 2$$



The screenshot shows a software interface with two panels. The top panel, labeled '1', displays the formula for $S(n)$ as a sum from $i=1$ to n of the product of $\ln(j+2)$ for $j=1$ to i , divided by $3 - \sin^2(i)$. The bottom panel, labeled '2', shows the calculation of $S(3)$ resulting in the value 2.0026750978.

$$S(n) = \sum_{i=1}^n \frac{\prod_{j=1}^i (\ln(j+2))}{3 - \sin^2(i)}$$
$$S(3) = 2.0026750978$$

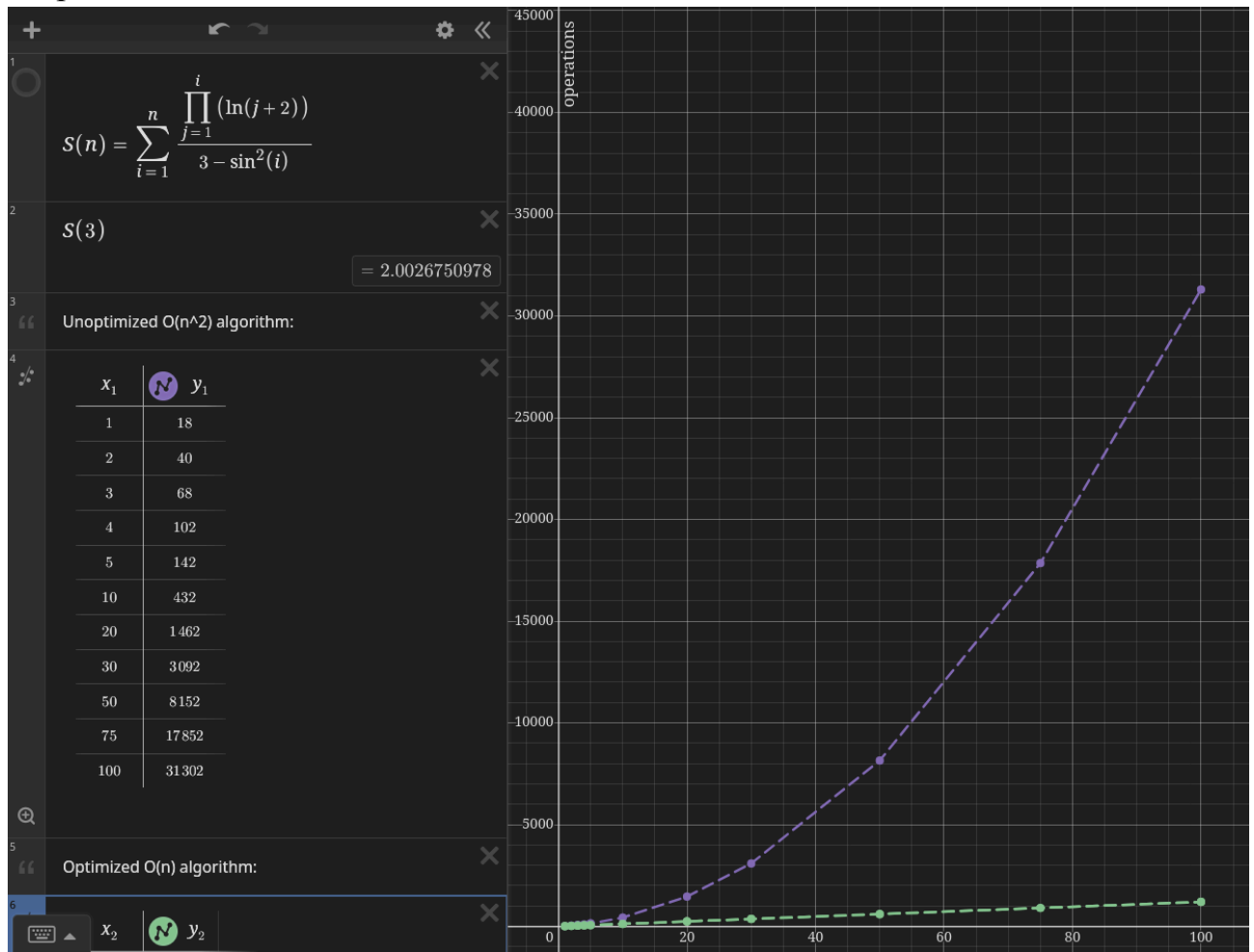
Результати тестування програми

```
theammir in ~/projects/asd/lab/2 λ ./a
1
Result: 0.4793401
Operations counter: 18
theammir in ~/projects/asd/lab/2 λ ./b
1
Result: 0.4793401
Operations counter: 15
```

```
theammir in ~/projects/asd/lab/2 λ ./a
2
Result: 1.1801570
Operations counter: 40
theammir in ~/projects/asd/lab/2 λ ./b
2
Result: 1.1801570
Operations counter: 27
```

```
theammir in ~/projects/asd/lab/2 λ ./a
3
Result: 2.0026751
Operations counter: 68
theammir in ~/projects/asd/lab/2 λ ./b
3
Result: 2.0026751
Operations counter: 39
```

На основі тестування обох програм побудував [графік](#) залежності кількості операцій від n :



Висновки

Протягом виконання лабораторної роботи написав дві програми різної часової складності, замірив кількість операцій, потрібних для їхнього виконання.

Виявилось, що цикл `for`, по своїй суті, це:

- 1 ініціалізація лічильника
- $i + 1$ перевірок умови
- $i - 1$ джампів до початку тіла циклу
- $i - 1$ інкрементів лічильника

всього $3i$ операцій (вище i - кількість ітерацій)

Оптимізував другу програму методом динамічного програмування, зберігаючи результати обчислень внутрішнього циклу для $1 \leq j \leq i - 1$. Таким чином, нове значення при кожному i вираховується виконанням однієї ітерації над збереженим.

Побудував графік залежності кількості операцій від n .

Оскільки внутрішній цикл виконував

$3i$ (власне цикл) + $(i - 1)$ (переініціалізація *numerator*) + $3(i - 1)$ (математика) =

$= 7i - 4$ зайвих операцій кожну ітерацію зовнішнього, можна наочно побачити перетворення квадратичної залежності на лінійну.

(раптові розрахунки – це не чат-гпт, це я, коли повертаюсь думками до лаби, яку зробив ~~кілька годин тому~~ вчора)