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

Лобань Михайло Юрійович

номер у списку групи: 21

Перевірила:

Молчанова А. А.

Київ 2024

Завдання:

Задано натуральне число n. Вирахувати значення заданої формули за варіантом.

Варіант №21

21. $$S = \sum_{i=1}^{n} \frac{\prod_{j=1}^{i} \left(\frac{j+2}{10} \right)}{i \cdot 2^i}$$

Текст програми (спосіб 1)

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

int main() {
   int n;
   double product = 1;
   double sum = 0;
   int powers = 1;

   int counter = 0;

   printf("enter n: ");
   scanf("%d", &n);

   counter+=4; //n, product, sum, powers initialization

   for (int i = 1; i <= n; i++) {//2 comparison, increment
      product = 1; //1 assignment
      for (int j = 1; j <= i; j++) { //2 comparison, increment
         product *= ((j + 2)/10.0); //3 sum, division, multiplication
         counter += 6; //5+1 jump to the start of the inner for-loop
      }
      powers *= 2;//1 multiplication
```

```
    sum += product / (powers * i);//3 sum, division, multiplication
    counter += 11; //7+1 jump to the start of the outer for-loop+3 for each extra increment, extra comparison, initialization of j
  }

  counter+=3;//3 for extra increment, comparison, i initialization

  printf("sum: %.7lf\n", sum);
  printf("amount of operations: %d\n", counter);
  return 0;
}
```

Результат тестування програми

```
enter n: 3
sum: 0.1675000
amount of operations: 76

Process returned 0 (0x0)   execution time : 0.912 s
Press any key to continue.
```

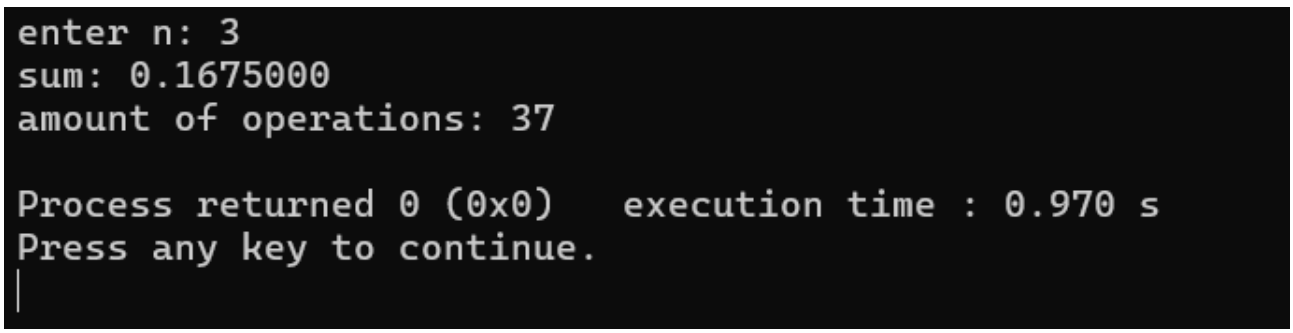
```
enter n: 4
sum: 0.1680625
amount of operations: 111

Process returned 0 (0x0)   execution time : 1.170 s
Press any key to continue.
```

Текст програми (спосіб 2)

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

int main() {
```

```
    int n;
    double product = 1;
    double sum = 0;
    int powers = 1;
    int counter = 0;

    printf("enter n: ");
    scanf("%d", &n);

    counter+=4; //n, product, sum, powers initialization

    for (int i = 1; i <= n; i++) { //2 comparison, increment
        product *= ((i + 2)/10.0); //3 sum, division, multiplication
        powers *= 2; //1 multiplication
        sum += product / (powers * i); //3 sum, division, multiplication
        counter+=10; //9+jump
    }

    counter+=3;//3 for extra increment, extra comparison, initialization of i

    printf("sum: %.7lf\n", sum);
    printf("amount of operations: %d\n", counter);
    return 0;
}
```

Результати тестування програми

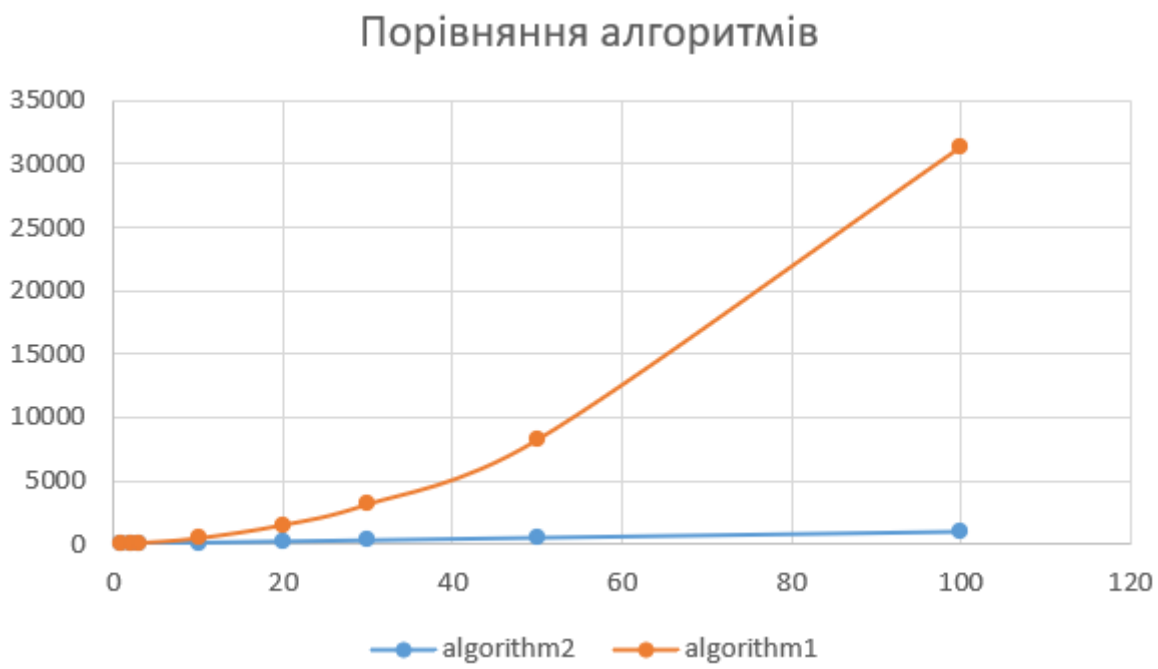


```
enter n: 5
sum: 0.1682200
amount of operations: 57

Process returned 0 (0x0)   execution time : 0.871 s
Press any key to continue.
```

Таблиці з результатами запуску

N:	1	2	3	10	20	30	50	100
Спосіб 1	***24***	***47***	***76***	***447***	***1487***	***3127***	***8207***	***31407***
Спосіб 2	***17***	***27***	***37***	***107***	***207***	***307***	***507***	***1007***

Графік за таблицею



Результати перевірки на калькуляторі

FROM THE MAKERS OF WOLFRAM LANGUAGE AND MATHEMATICA

WolframAlpha

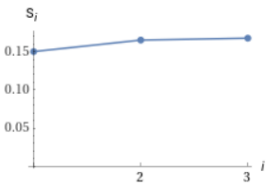
$$\sum_{i=1}^{3} \frac{\prod_{j=1}^{i} \left(\frac{j+2}{10}\right)}{i * 2^i}$$

NATURAL LANGUAGE | MATH INPUT

Sum

$$\sum_{i=1}^{3} \frac{\prod_{j=1}^{i} \frac{j+2}{10}}{i\,2^i} = \frac{67}{400}$$

Partial sums



POWERED BY THE WOLFRAM LANGUAGE

$$67 \div 400 =$$

$$0{,}1675$$

WolframAlpha

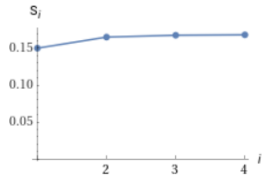
$$\sum_{i=1}^{4} \frac{\prod_{j=1}^{i} \left(\frac{j+2}{10}\right)}{i * 2^i}$$

NATURAL LANGUAGE | MATH INPUT

Sum

$$\sum_{i=1}^{4} \frac{\prod_{j=1}^{i} \frac{j+2}{10}}{i\,2^i} = \frac{2689}{16\,000}$$

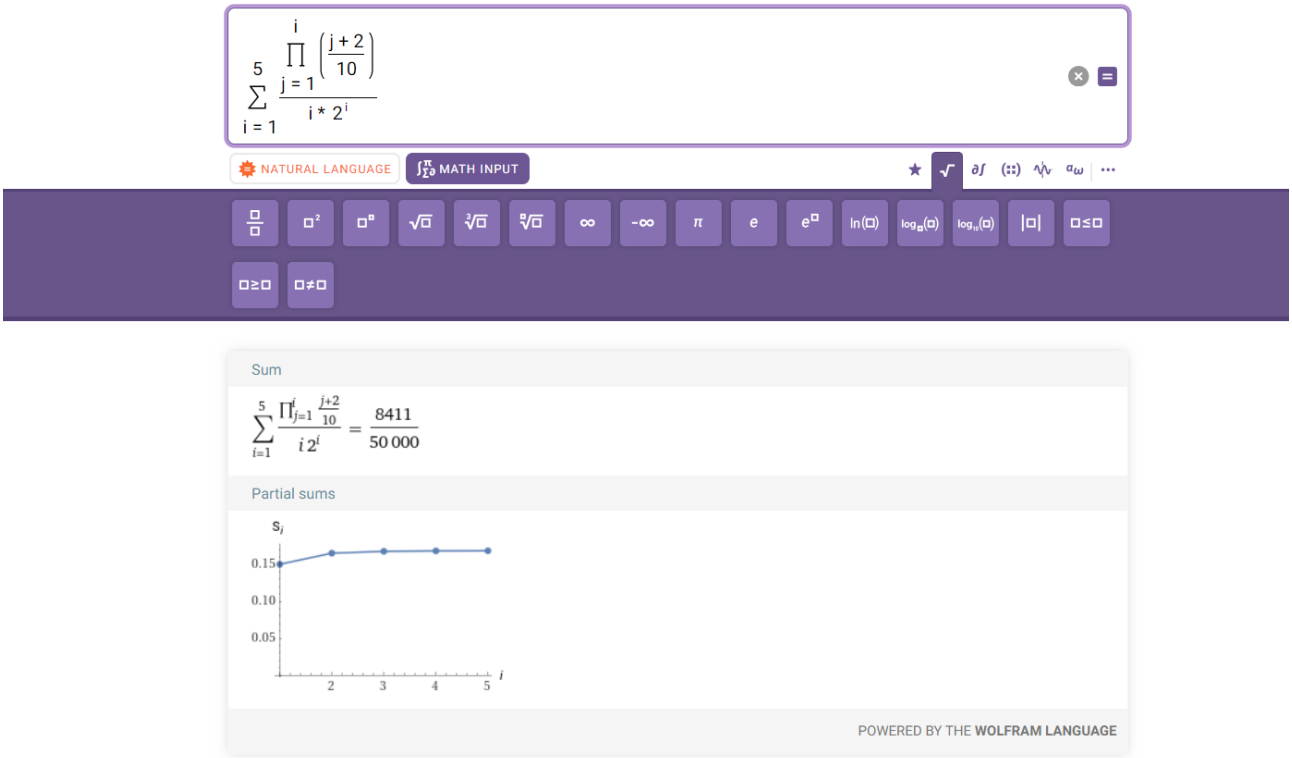
Partial sums



POWERED BY THE WOLFRAM LANGUAGE

2689 ÷ 16000 =

0,1680625



8411 ÷ 50000 =

0,16822

Висновок

Отже, після вирішення поставленої задачі із використанням двох підходів — вкладених циклів та методу динамічного програмування — стало очевидним, що метод динамічного програмування значно спрощує алгоритм програми. Він зменшує кількість операцій, необхідних для виконання, що робить цей метод більш ефективним у порівнянні з підходом, який базується на вкладених циклах. Використання методу динамічного програмування дозволяє значно знизити обчислювальну складність задачі, особливо при роботі з великими значеннями n. Таким чином, цей метод не лише спрощує алгоритм, але й підвищує продуктивність програми, що підтверджується зменшенням кількості викликів стандартних функцій і скороченням часу виконання.