**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: Перевірила:

студент групи ІМ-31 Молчанова А. А.  
Литвиненко Сергій Андрійович  
номер у списку групи: 14

Київ 2023

**Завдання**

1. Задане натуральне число ***n***. Вирахувати значення заданої формули за варіантом.
2. Для вирішення 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 вирішувати поставлену задачу при будь-якому заданому ***n***, для якого результат обчислення може бути коректно представлений типом ***double***.
5. Результуючі дані вивести у форматі з сімома знаками після крапки.

Варіант 14:

**Спосіб I**

**Текст програми**

#include<stdio.h>

#include<math.h>

double f(unsigned n, unsigned\* cntOperations) {

double result = 1;

unsigned cnt = 4; // =1 | =1 | \*cntOperations | =

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

double denominator = 0.0;

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

denominator = denominator + sin(j);

cnt += 6; // <= | ++ | = | + | sin | jmp

}

result = result \* ( ( cos(i) + 1 ) / denominator );

cnt += 10; // <= | ++ | =0.0 | =1 | = | \* | cos | + | / | jmp

}

\*cntOperations = cnt;

return result;

}

int main(int argc, char\* argv[]) {

unsigned n, countOfOperations;

printf("Enter natural number: ");

scanf("%u", &n);

double res = f(n, &countOfOperations);

printf("f(%d) = %.7lf\nCount of operations = %u\n", n, res, countOfOperations);

return 0;

}

**Кількість операцій**

Виведемо формулу для підрахунку загальної кількості виконаних операцій в алгоритмі. Кількість ітерацій зовнішнього циклу дорівнює переданому аргументу n. Кількість ітерацій внутрішнього циклу залежить від лічильника зовнішнього циклу (*i*), і на кожній ітерації зовнішнього циклу буде виконуватися *i* разів. Таким чином, загальна кількість ітерацій зовнішнього циклу дорівнює n, а внутрішнього – . Отже, загальна кількість ітерацій дорівнює . Кількість операцій, що не залежать від переданого аргументу дорівнює 4. Знаючи загальну кількість ітерацій, кількість операцій в кожній ітерації та кількість операцій, що не залежать від переданого аргументу, можна обчислити загальну кількість операцій: . Спростивши вираз, отримаємо . Отже, даний алгоритм має складність .

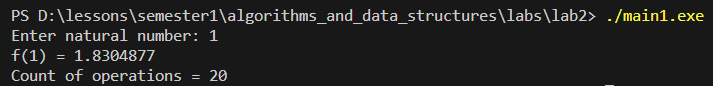
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **n** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| **Кількість операцій** | 4 | 20 | 42 | 70 | 104 | 144 | 190 | 242 | 300 | 364 |

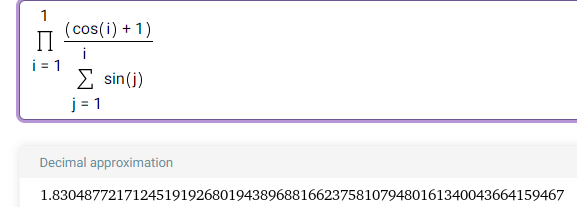
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **n** | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| **Кількість операцій** | 434 | 510 | 592 | 680 | 774 | 874 | 980 | 1092 | 1210 | 1334 |

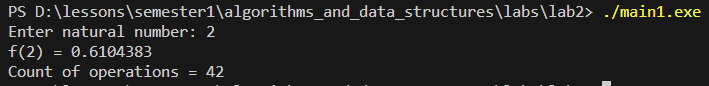
Рисунок 1 - Графік функції

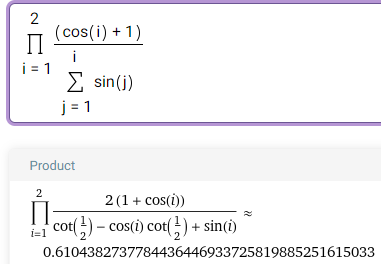
**Тестування програми**

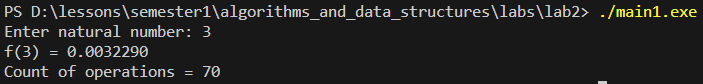
Обчислимо формулу за n = 3.

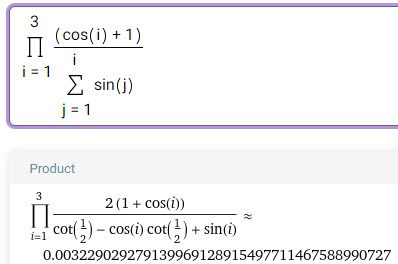












**Спосіб II**

**Текст програми**

#include<stdio.h>

#include<math.h>

double f(unsigned n, unsigned\* cntOperations) {

double result = 1;

unsigned cnt = 5; // =1 | =0.0 | =1 | \*cntOperations | =

double sinsSum = 0.0;

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

sinsSum = sinsSum + sin(i);

result = result \* ( ( cos(i) + 1 ) / sinsSum );

cnt += 11; // <= | ++ | = | + | sin | = | \* | cos | + | / | jmp

}

\*cntOperations = cnt;

return result;

}

int main(int argc, char\* argv[]) {

unsigned n, countOfOperations;

printf("Enter natural number: ");

scanf("%u", &n);

double res = f(n, &countOfOperations);

printf("f(%d) = %.7lf\nCount of operations = %u\n", n, res, countOfOperations);

return 0;

}

**Кількість операцій**

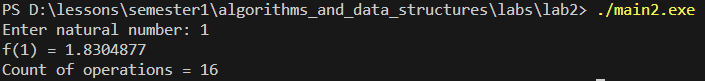
Виведемо формулу для підрахунку загальної кількості виконаних операцій в алгоритмі. Алгоритм має лише один цикл, кількість ітерацій якого дорівнює переданому аргументу n. Тобто, загальна кількість ітерацій дорівнює . Кількість операцій, що не залежать від переданого аргументу дорівнює 5. Знаючи загальну кількість ітерацій, кількість операцій в кожній ітерації та кількість операцій, що не залежать від переданого аргументу, можна обчислити загальну кількість операцій: . Отже, даний алгоритм має складність .

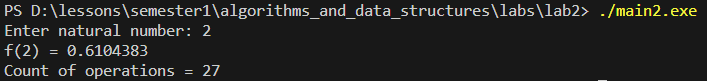
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **n** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| **Кількість операцій** | 5 | 16 | 27 | 38 | 49 | 60 | 71 | 82 | 93 | 104 |

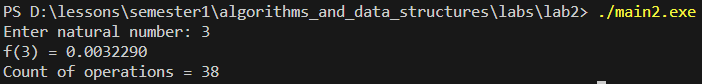
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **n** | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| **Кількість операцій** | 115 | 126 | 137 | 148 | 159 | 170 | 181 | 192 | 203 | 214 |

Рисунок 2 - Графік функції

**Тестування програми**







Результати обох програм співпадають з виразами, обчисленими на калькуляторі.

**Висновок**

В ході виконання лабораторної роботи було розроблено два алгоритми для обрахування виразу за формулою. Для зменшення складності алгоритму був використаний метод динамічного програмування. Перший алгоритм мав складність , другий - . Побудувавши таблички та намалювавши графіки алгоритмів, стало зрозуміло, що другий алгоритм є більш ефективним.

Нижче зображені графіки обох алогритмів, синьою лінією позначений перший алгоритм, червоною – другий. По горизонтальній осі відкладено значеня аргумента n, по вертикальній – кількість операцій.

Рисунок 3 – Порівнння графіків алгоритмів