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

з дисципліни
«Алгоритми і структури даних»

Виконав:
студент групи ІМ-44
Мундурс Нікіта Юрійович
номер у списку групи: 16

Перевірив:
Сергієнко М. А.

Київ 2025

Завдання

Дане натуральне число *n*. Знайти суму перших *n* членів ряду чисел, заданого рекурентною формулою. Розв’язати задачу ***трьома способами***:

1) у програмі використати рекурсивну функцію, яка виконує обчислення і членів ряду, і суми на рекурсивному спуску;
2) у програмі використати рекурсивну функцію, яка виконує обчислення і членів ряду, і суми на рекурсивному поверненні;
3) у програмі використати рекурсивну функцію, яка виконує обчислення членів ряду на рекурсивному спуску, а обчислення суми на рекурсивному поверненні.

При проєктуванні програм ***слід врахувати наступне***:

1) програми повинні працювати коректно для довільного цілого додатного *n* включно з *n* = 1;
2) видимість змінних має обмежуватися тими ділянками, де вони потрібні;
3) функції повинні мати властивість модульності;
4) у кожному з трьох способів рекурсивна функція має бути одна (за потреби, можна також використати додаткову функцію-обгортку (wrapper function));
5) у другому способі можна використати запис (struct) з двома полями (але в інших способах у цьому немає потреби і це вважатиметься надлишковим);
6) програми мають бути написані мовою програмування С.

Варіант № 16

$$F_1 = x; \quad F_{i+1} = F_i \cdot (2i-1)^2 \cdot x^2/(4i^2+2i), \quad i > 0;$$

$$\textstyle\sum_{i=1}^{n} F_i = \arcsin x, \quad |x| < 1.$$

Тексти програм

1. Обчислення і членів ряду, і суми на рекурсивному спуску (descent.c)

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

double recursion(double x, unsigned int n, unsigned int i, double Fi, double sum) {
    if (i > 0) {
        Fi = Fi * (2 * i - 1) * (2 * i - 1) * x * x / (4 * i * i + 2 * i);
```

```
    }
    else {
        Fi = x;
    }

    sum += Fi;
    i++;

    if (i < n) {
        return recursion(x, n, i, Fi, sum);
    }
    return sum;
}

int main()
{
    unsigned int n;
    double x;

    printf("Enter x: ");
    scanf("%lf", &x);
    printf("Enter n: ");
    scanf("%u", &n);

    if (x > -1 && x < 1) {
        printf("Result: %.8lf\n", recursion(x, n, 0, 0, 0));
    }
    else {
        printf("x must be in the range (-1; 1)\n");
    }
```

```
    return 0;
}
```

2. Обчислення і членів ряду, і суми на рекурсивному поверненні (comeback.c)

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

struct previousRes {
    double Fi;
    double sum;
};

struct previousRes recursion(double x, unsigned int n) {
    double i = n - 1;
    struct previousRes currentRes;

    if (n > 1) {
        currentRes = recursion(x, n - 1);
        currentRes.Fi = currentRes.Fi * (2 * i - 1) * (2 * i - 1) * x * x / (4 * i * i + 2 * i);
    }
    else {
        currentRes.Fi = x;
        currentRes.sum = 0;
    }

    currentRes.sum += currentRes.Fi;
    return currentRes;
}
```

```c
int main()
{
    unsigned int n;
    double x;

    printf("Enter x: ");
    scanf("%lf", &x);
    printf("Enter n: ");
    scanf("%u", &n);

    if (x > -1 && x < 1) {
        struct previousRes finalRes = recursion(x, n);
        printf("Result: %.8lf\n", finalRes.sum);
    }
    else {
        printf("x must be in the range (-1; 1)\n");
    }

    return 0;
}
```

3. Обчислення членів ряду на рекурсивному спуску, а обчислення суми на рекурсивному поверненні (mixed.c)

```c
#include <stdio.h>
#include <stdlib.h>

double recursion(double x, unsigned int n, unsigned int i, double Fi, double sum) {
    if (i > 0) {
        Fi = Fi * (2 * i - 1) * (2 * i - 1) * x * x / (4 * i * i + 2 * i);
```

```
    }
    else {
        Fi = x;
    }

    double currentFi = Fi;
    i++;

    if (i < n) {
        sum = recursion(x, n, i, Fi, sum);
    }

    sum += currentFi;
    return sum;
}

int main()
{
    unsigned int n;
    double x;

    printf("Enter x: ");
    scanf("%lf", &x);
    printf("Enter n: ");
    scanf("%u", &n);

    if (x > -1 && x < 1) {
        printf("Result: %.8lf\n", recursion(x, n, 0, 0, 0));
    }
    else {
```

```
        printf("x must be in the range (-1; 1)\n");
    }

    return 0;
}
```

4. Тестування: циклічний варіант рішення задачі (test.c)

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

double loop(double x, unsigned int n) {
    int i;
    double xSquare = x * x;
    double sum = x;
    double F = x;

    for (i = 1; i < n; i++) {
        F = F * (2 * i - 1) * (2 * i - 1) * xSquare / (4 * i * i + 2 * i );
        sum += F;
    }

    return sum;
}



int main()
{
    unsigned int n;
    double x;
```

```
    printf("Enter x: ");
    scanf("%lf", &x);
    printf("Enter n: ");
    scanf("%u", &n);

    if (x > -1 && x < 1) {
        printf("Result: %.8lf\n", loop(x, n));
    }
    else {
        printf("x must be in the range (-1; 1)\n");
    }

    return 0;
}
```

Результати тестування

Обране значення: *x = 0,459*

1 спосіб

```
nikita@MiWiFi-R4CM-srv:~/Programming/LabsASD/semester2/lab1$ ./descent
Enter x: 0.459
Enter n: 5
Result: 0.47686420
```

2 спосіб

```
nikita@MiWiFi-R4CM-srv:~/Programming/LabsASD/semester2/lab1$ ./comeback
Enter x: 0.459
Enter n: 5
Result: 0.47686420
```

3 спосіб

```
nikita@MiWiFi-R4CM-srv:~/Programming/LabsASD/semester2/lab1$ ./mixed
Enter x: 0.459
Enter n: 5
Result: 0.47686420
```

4 спосіб

```
nikita@MiWiFi-R4CM-srv:~/Programming/LabsASD/semester2/lab1$ ./test
Enter x: 0.459
Enter n: 5
Result: 0.47686420
```

Калькулятор

$n_1 = 0{,}459$

$n_2 = 0{,}0161171$

$$\frac{(0{,}459)(2\times1-1)^2(0{,}459)^2}{(4\times1^2+2\times1)}$$

$$=\frac{153\times459^2}{2\times1000^3}$$

Альтернативна форма

$\approx 0{,}0161171$

$n_3 = 0{,}001528$

$$\frac{\frac{153\times459^2}{2\times1000^3}(2\times2-1)^2(0{,}459)^2}{(4\times2^2+2\times2)}$$

$$=\frac{1377\times459^4}{40\times1000^5}$$

Альтернативна форма

$\approx 0{,}001528$

$n_4 = 0{,}00019162$

$$\frac{\frac{1377 \times 459^4}{40 \times 1000^5}(2 \times 3 - 1)^2 (0{,}459)^2}{(4 \times 3^2 + 2 \times 3)}$$

$$= \frac{459^7}{14 \times 40^6 \times 5^8 \times 1000^2}$$

Альтернативна форма

$\approx 0{,}00019162$

$n_5 = 0{,}0000274745$

$$\frac{\frac{459^7}{14 \times 40^6 \times 5^8 \times 1000^2}(2 \times 4 - 1)^2 (0{,}459)}{(4 \times 4^2 + 2 \times 4)}$$

$$= \frac{357 \times 459^8}{16 \times 40^6 \times 5^8 \times 1000^4}$$

Альтернативна форма

$\approx 2{,}74745 \times 10^{-5}$

sum = **0,476864**

$$0{,}459 + \frac{153 \times 459^2}{2 \times 1000^3} + \frac{1377 \times 459^4}{40 \times 1000^5} + \frac{459^7}{14 \times 40^6 \times 5^8 \times 1000^2} + \frac{357 \times 459^8}{16 \times 40^6 \times 5^8 \times 1000^4}$$

Перетворіть вираз

$$\frac{459}{1000} + \frac{153 \times 459^2}{2 \times 1000^3} + \frac{1377 \times 459^4}{40 \times 1000^5} + \frac{459^7}{14 \times 40^6 \times 5^8 \times 1000^2} + \frac{357 \times 459^8}{16 \times 40^6 \times 5^8 \times 1000^4}$$

$\approx 0{,}476864$

Похибка обчислень

n = 5

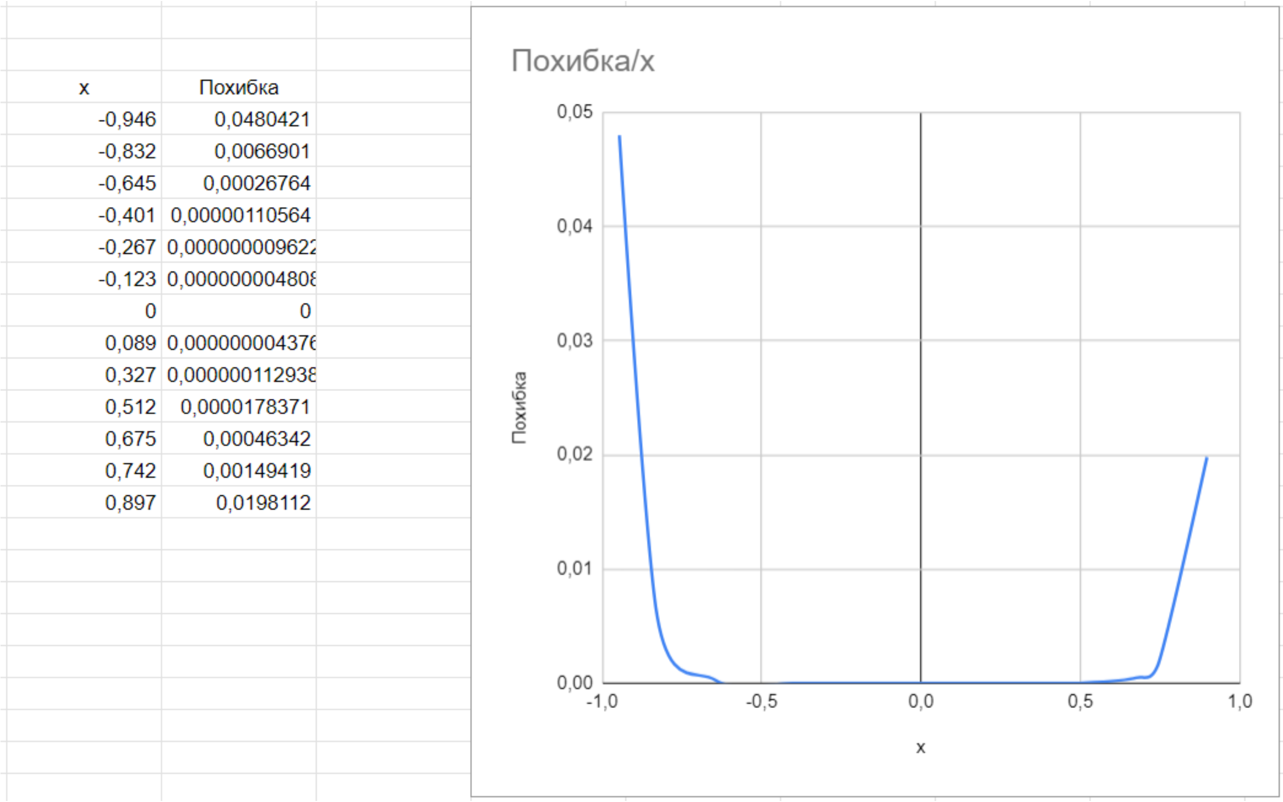
Таблиця значень

Значення x		-0,946	-0,832	-0,645	-0,401	-0,267	-0,123
Результат	Програма	-1,19262360	-0,97601296	-0,70075564	-0,41260709	-0,27027866	-0,12331228
	Калькулятор	-1,24066568	-0,98270306	-0,70102328	-0,41260820	-0,27027867	-0,12331228
	Похибка	0,0480421	0,0066901	0,00026764	$1{,}106 * 10^{-6}$	$9{,}62 * 10^{-9}$	$4{,}81 * 10^{-9}$

	Значення x	0	0,089	0,327	0,512	0,675	0,742	0,897
Результат	Програма	0	0,08911792	0,33312719	0,53749367	0,74050128	0,83455456	1,09312400
	Калькулятор	0	0,08911792	0,33312730	0,53751151	0,74096470	0,83604875	1,11293521
	Похибка	0	$4{,}38 * 10^{-9}$	$1{,}13 * 10^{-7}$	$1{,}78 * 10^{-5}$	0,00046342	0,00149419	0,0198112

Графік

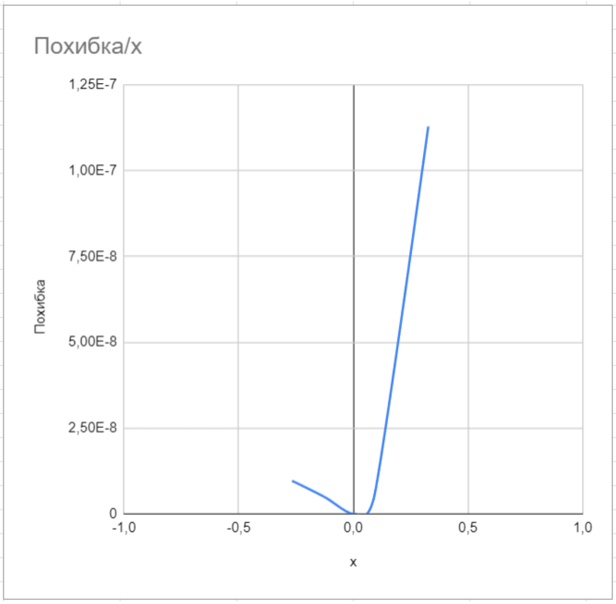
1. Загальний графік



x	Похибка
-0,946	0,0480421
-0,832	0,0066901
-0,645	0,00026764
-0,401	0,00000110564
-0,267	0,000000009622
-0,123	0,000000004808
0	0
0,089	0,000000004376
0,327	0,000000112938
0,512	0,0000178371
0,675	0,00046342
0,742	0,00149419
0,897	0,0198112

2. Графік для менших значень похибки (щоб можна було роздивитися графік, де похибка значно менша)

x	Похибка
-0,267	0,000000009622
-0,123	0,000000004808
0	0
0,089	0,000000004376
0,327	0,000000112938



Висновок: на цій лабораторній роботі я навчився використовувати рекурсивні алгоритми при написанні програм. Під час виконання завдання я зрозумів, що рекурсію наочніше та доречніше використовувати для задач, заданих рекурентними відношеннями. Однак рекурсивні алгоритми мають також і свої недоліки, зокрема: низька ефективність через велику кількість викликів підпрограм, необхідність використання великого обсягу пам’яті. На графіку вимірювань можна побачити, що найбільшою похибка буде при значеннях x, близьких до -1 та 1, а найменшою – при значеннях, близьких до 0. Також можна зробити висновок, що використання рядів для обчислення наближених значень функцій є корисним, коли певна похибка є не суттєвою.