

**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»

Виконала:

студентка групи ІМ - 43
Хоанг Чан Кам Лі
номер у списку групи: 29

Перевірив:

Сергієнко А. М.

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

Варіант 29:

$$F_1 = x / (0.525 + 0.5x)^2 - 1; \quad F_{i+1} = F_i \cdot F_1(3 - 2i) / (2i), \quad i > 0;$$
$$\sum_{i=1}^n F_i = \sqrt{x}, \quad 0.5 < x < 1.$$

Текст програми

1. Обчислення членів ряду i суми на рекурсивному спуску

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

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

```
double calculateF1(double x) {
```

```
    return x / pow(0.525 + 0.5 * x, 2) - 1;
```

```
}
```

```
double recDescend(int i, int n, double x, double *sum, double F1, double prev_F) {
```

```
    double F_i;
```

```
    if (i == 1) {
```

```
        F_i = F1;
```

```
    } else {
```

```
        F_i = prev_F * F1 * (3.0 - 2.0 * (i - 1)) / (2.0 * (i - 1));
```

```
    }
```

```

    *sum += F_i;

    if (i == n) {
        return F_i;
    }
    return recDescend(i + 1, n, x, sum, F1, F_i);
}

double sumSeries1(int n, double x) {
    double sum = 0;
    double F1 = calculateF1(x);
    recDescend(1, n, x, &sum, F1, 0);
    return sum;
}

int main() {
    int n;
    printf("Enter n: ");
    scanf("%d", &n);

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

    if (n < 1 || x >= 1 || x <= 0.5) {
        printf("Incorrect input (n >= 1, 0.5 < x < 1)\n");
        return 1;
    }

    double result = sumSeries1(n, x);
    printf("Result: %.10f\n", result);
}

```

```
    return 0;
}
```

2. Обчислення членів ряду і суми на рекурсивному поверненні

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

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

```
typedef struct {
    double element;
    double sum;
} Result;
```

```
double calculateF1(double x) {
    return x / pow(0.525 + 0.5 * x, 2) - 1;
}
```

```
Result recReturn(int i, int n, double x, double F1) {
    Result result;

    if (i == 1) {
        result.element = F1;
        result.sum = F1;
        return result;
    }
```

```
    Result prev = recReturn(i - 1, n, x, F1);

    return result;
}
```

```
double sumSeries2(int n, double x) {
    double F1 = calculateF1(x);
```

```

    Result result = recReturn(n, n, x, F1);
    return result.sum;
}

int main() {
    int n;
    printf("Enter n: ");
    scanf("%d", &n);

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

    if (n < 1 || x >= 1 || x <= 0.5) {
        printf("Incorrect input (n >= 1, 0.5 < x < 1)\n");
        return 1;
    }
    double result = sumSeries2(n, x);
    printf("Result: %.10f\n", result);

    return 0;
}

```

3. Обчислення членів ряду на рекурсивному спуску, а обчислення суми - на рекурсивному поверненні.

```

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

```

```

double calculateF1(double x) {
    return x / pow(0.525 + 0.5 * x, 2) - 1;
}

```

```

double recMixed(int i, int n, double x, double F1, double prev_element) {

```

```
double F_i;
```

```
if (i == 1) {
```

```
    F_i = F1;
```

```
} else {
```

```
    F_i = prev_element * F1 * (3.0 - 2.0 * (i - 1)) / (2.0 * (i - 1));
```

```
}
```

```
if (i == n) {
```

```
    return F_i;
```

```
}
```

```
return F_i + recMixed(i + 1, n, x, F1, F_i);
```

```
}
```

```
double sumSeries3(int n, double x) {
```

```
    double F1 = calculateF1(x);
```

```
    return recMixed(1, n, x, F1, 0);
```

```
}
```

```
int main() {
```

```
    int n;
```

```
    printf("Enter n: ");
```

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

```
    double x;
```

```
    printf("Enter x: ");
```

```
    scanf("%lf", &x);
```

```
if (n < 1 || x >= 1 || x <= 0.5) {
```

```
    printf("Incorrect input (n >= 1, 0.5 < x < 1)\n");
```

```

    return 1;
}
double result = sumSeries3(n, x);
printf("Result: %.10f\n", result);

```

```

    return 0;
}

```

4. Циклічна програма вирішення задачі

```

#include <stdio.h>

```

```

#include <math.h>

```

```

double calculateF1(double x) {
    return x / pow(0.525 + 0.5 * x, 2) - 1;
}

```

```

double sumSeriesLoop(int n, double x) {
    double sum = 0;
    double F1 = calculateF1(x);
    double F_i = F1;

```

```

    sum += F_i;

```

```

    for (int i = 2; i <= n; i++) {
        F_i = F_i * F1 * (3 - 2 * (i - 1)) / (2 * (i - 1));
        sum += F_i;
    }

```

```

    return sum;
}

```

```

int main() {

```

```

int n;

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

double x;

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

if (n < 1 || x >= 1 || x <= 0.5) {
    printf("Incorrect input (n >= 1, 0.5 < x < 1)\n");
    return 1;
}

double result = sumSeriesLoop(n, x);
printf("Result: %.10f\n", result);
return 0;
}

```

Результати тестування програми та перевірки

```

C:\Users\tranc\ASD_2\Lab_1>1_descent
Enter n: 5
Enter x: 0.75
Result: -0.0712778158

C:\Users\tranc\ASD_2\Lab_1>2_ascent
Enter n: 5
Enter x: 0.75
Result: -0.0712778158


C:\Users\tranc\ASD_2\Lab_1>3_mixed
Enter n: 5
Enter x: 0.75
Result: -0.0712778158

C:\Users\tranc\ASD_2\Lab_1>4_loop
Enter n: 5
Enter x: 0.75
Result: -0.0712778158

```


$$f(1) = \frac{0.75}{(0.525 + 0.5 \cdot 0.75)^2} - 1$$

$$f(n) = f(n-1) \cdot f(1) \cdot \frac{(3 - 2(n-1))}{2(n-1)}$$

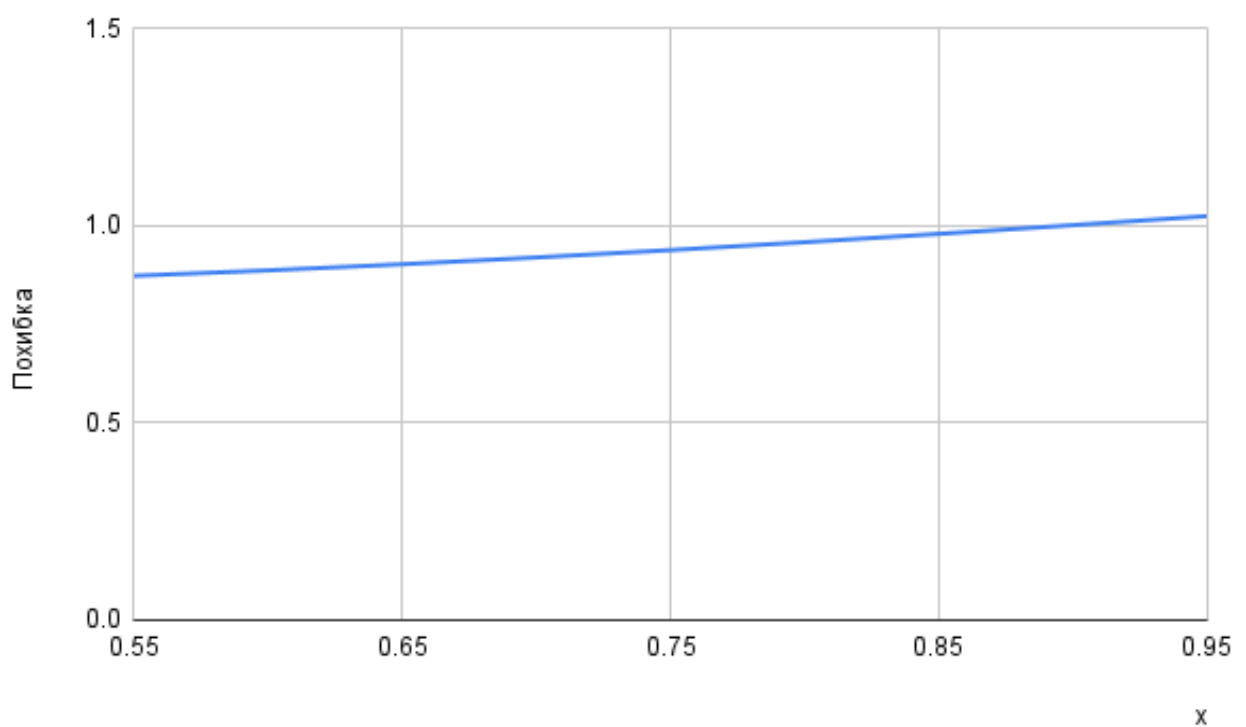
x	 $f(x)$	
1	-0.074074074	
2	0.0027434842	
3	5.080526×10^{-5}	
4	1.881676×10^{-6}	
5	8.711465×10^{-8}	

$$f(1) + f(2) + f(3) + f(4) + f(5)$$

$$= -0.0712778157946$$

Графік похибки обчислення функції

при $n = 5$



Висновки

У цій лабораторній роботі було реалізовано три різні підходи до обчислення рекурсивної функції, що задається у вигляді ряду.

У першому способі обчислення членів ряду та суми відбувалося під час рекурсивного спуску. Такий підхід є простим у реалізації, однак менш гнучким, оскільки вся логіка працює "вперед", без можливості уточнення результату при поверненні.

У другому способі всі обчислення виконувалися на етапі рекурсивного повернення. Це дозволяє краще контролювати акумуляцію суми, однак ускладнює логіку побудови, бо потрібно зберігати проміжні значення.

Третій спосіб поєднав обидва підходи: значення членів ряду обчислювалися на спуску, а підсумовування — на поверненні. Такий варіант виявився найбільш оптимальним за балансом ефективності й зрозумілості коду.

У процесі виконання було також проаналізовано похибку результатів у порівнянні з еталонним значенням функції, що дозволило оцінити точність реалізації. Рекурсивні методи зручні для реалізації складних математичних формул, однак можуть бути менш ефективними за ресурсами, у порівнянні з ітеративними аналогами (тобто, з розрахунками за допомогою циклів), особливо при значних значеннях n .