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-41
Номер у списку групи: 7

Перевірив:

Сергієнко А. М.

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.

Варіант 7:

Варіант № 7

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

$$\sum_{i=1}^{n} F_i = 0.5 \ln x, \quad x > 0.$$

Текст програми

```c
#include <stdio.h>

typedef struct {
   double last_calculated_value;
   double sum;
} Result;

Result calculate_sum_tail_recursion(const int n, const double x) {
   Result calcData = {0, 0};

   double res;
```

```
    if (n == 1) {
        res = (x - 1) / (x + 1);
    } else {
        calcData = calculate_sum_tail_recursion(n - 1, x);

        res = calcData.last_calculated_value * ((2 * n - 1) * (x - 1) *
(x - 1)) / (
                  (2 * n + 1) * (2 * n + 1) * (x + 1) * (x + 1));
    }



    return (Result){
        .sum = calcData.sum + res,
        .last_calculated_value = res
    };
}

void sum_tail_recursion(const int n, const double x) {
    const double res = calculate_sum_tail_recursion(n, x).sum;
    printf("sum_tail_recursion: %lf\n", res);
}

double calculate_sum_head_recursion(const int n, const int i, const
double x, const double prev) {
    if (i > n) return 0;

    double res = 0;

    if (i == 1) {
```

```
        res = (x - 1) / (x + 1);
    } else {
        res = prev * ((2 * i - 1) * (x - 1) * (x - 1)) / ((2 * i + 1) *
(2 * i + 1) * (x + 1) * (x + 1));
    }

    return res + calculate_sum_head_recursion(n, i + 1, x, res);
}



void sum_head_recursion(const int n, const double x) {
    const double res = calculate_sum_head_recursion(n, 1, x, x);
    printf("sum_head_recursion: %lf\n", res);
}



double calculate_sum_recursive_combined(const int n, const int i,
const double x, const double prev) {
    if (i > n) {
        return prev;
    }

    double val;
    if (i == 1) {
        val = (x - 1) / (x + 1);
    } else {
        val = prev * ((2 * i - 1) * (x - 1) * (x - 1)) / ((2 * i + 1) *
(2 * i + 1) * (x + 1) * (x + 1));
    }
```

```
    double res = calculate_sum_recursive_combined(n, i + 1, x, val);
    res += prev;

    return res;
}

void sum_recursive_combined(const int n, const double x) {
    const double res = calculate_sum_recursive_combined(n, 1, x, 0);
    printf("sum_recursive_combined: %lf\n", res);
}

// циклічний варіант рішення задачі
double sum_iterative(const int n, const double x) {
    double Fi = (x - 1) / (x + 1);

    double sum = Fi;

    for (int i = 2; i ⩽ n; i++) {
        Fi = Fi * ((2 * i - 1) * (x - 1) * (x - 1)) / ((2 * i + 1) * (2
* i + 1) * (x + 1) * (x + 1));
        sum += Fi;
    }

    return sum;
}

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

```c
    printf("Enter the value for n: ");
    scanf("%d", &n);

    printf("Enter the value for x (x > 0): ");
    scanf("%lf", &x);

    sum_tail_recursion(n, x);
    sum_head_recursion(n, x);
    sum_recursive_combined(n, x);

    printf("sum_iterative: %lf\n", sum_iterative(n, x));

    return 0;
}
```

Оточення:

```
➜  lab-new-1 gcc --version
Apple clang version 16.0.0 (clang-1600.0.26.6)
Target: arm64-apple-darwin24.3.0
Thread model: posix
```

```
➜  lab-new-1 gcc -o bin01 main.c
➜  lab-new-1 ls bin*
bin01
```

Тести програми:

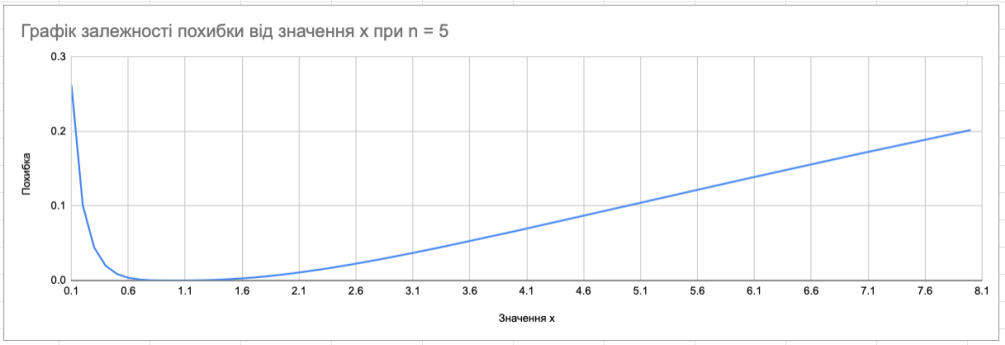
```
(base) ➜  lab-new-1 gcc -o bin01 main.c
(base) ➜  lab-new-1 ./bin01
Enter the value for n: 6
Enter the value for x (x > 0): 0.6
sum_tail_recursion: -0.251887
sum_head_recursion: -0.251887
sum_recursive_combined: -0.251887
sum_iterative: -0.251887
(base) ➜  lab-new-1 ./bin01
Enter the value for n: 5
Enter the value for x (x > 0): 0.5
sum_tail_recursion: -0.337829
sum_head_recursion: -0.337829
sum_recursive_combined: -0.337829
sum_iterative: -0.337829
(base) ➜  lab-new-1 ./bin01
Enter the value for n: 1
Enter the value for x (x > 0): 0.4
sum_tail_recursion: -0.428571
sum_head_recursion: -0.428571
sum_recursive_combined: -0.428571
sum_iterative: -0.428571
(base) ➜  lab-new-1 ./bin01
Enter the value for n: 5
Enter the value for x (x > 0): 0.9
sum_tail_recursion: -0.052649
sum_head_recursion: -0.052649
sum_recursive_combined: -0.052649
sum_iterative: -0.052649
(base) ➜  lab-new-1 ./bin01
Enter the value for n: 5
Enter the value for x (x > 0): 1.1
sum_tail_recursion: 0.047632
sum_head_recursion: 0.047632
sum_recursive_combined: 0.047632
sum_iterative: 0.047632
```

```
(base) ➜  lab-new-1 ./bin01
Enter the value for n: 5
Enter the value for x (x > 0): 1.2
sum_tail_recursion: 0.090999
sum_head_recursion: 0.090999
sum_recursive_combined: 0.090999
sum_iterative: 0.090999
```

Графік залежності похибки від значення x

Table1

#	x	0.5 * ln(x)	Програма	Похибка
0.1	-1.151292546	-0.88867	0.2626225465
0.2	-0.8047189562	-0.703899	0.1008199562
0.3	-0.6019864022	-0.557765	0.04422140216
0.4	-0.4581453659	-0.438197	0.01994836594
0.5	-0.3465735903	-0.337829	0.00874459028
0.6	-0.2554128119	-0.251887	0.003525811883
0.7	-0.178337472	-0.177132	0.001205471969
0.8	-0.1115717757	-0.111276	0.000295775657
0.9	-0.0526802578	-0.052649	0.000031257828
1	0	0	0
1.1	0.0476550899	0.047632	0.000023089902
1.2	0.0911607784	0.090999	0.000161778397
1.3	0.1311821322	0.130702	0.000480132233
1.4	0.1682361183	0.167224	0.001012118311
1.5	0.2027325541	0.200964	0.001768554054
1.6	0.2350018146	0.232252	0.002749814623
1.7	0.2653141255	0.261365	0.003949125531
1.8	0.2938933325	0.288537	0.005356332451
1.9	0.3209269431	0.313967	0.006959943086
2	0.3465735903	0.337829	0.00874459028
2.1	0.3709686724	0.36027	0.01069867236
2.2	0.3942286802	0.38142	0.01280868018
2.3	0.4164545615	0.401393	0.01506156147
2.4	0.4377343687	0.42029	0.01744436868
2.5	0.4581453659	0.438197	0.01994836594
2.6	0.4777557225	0.455195	0.02256072251
2.7	0.4966258865	0.471354	0.02527188651
2.8	0.5148097086	0.486736	0.02807370859
2.9	0.5323553685	0.501398	0.0309573685
3	0.5493061443	0.515391	0.03391514433



Висновок:

Під час виконання роботи навчився використовувати рекурсивні алгоритми. Код, написаний з використанням рекурсії, часто виглядає більш зрозумілим, але часто не є найефективнішим, тому під час використання рекурсії потрібно враховувати вимоги до швидкості алгоритму. Навчився експортувати дані програми в google таблиці та створювати графік