**Лабораторна робота №1**

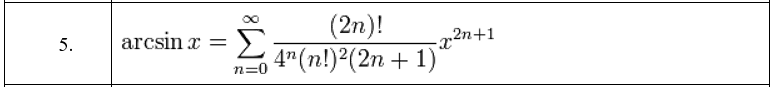
**Визначення та композиція функцій**

**Цілі роботи**

* отримання навичок програмування на мові F#, використовуючи принципи функціонального програмування;
* засвоєння методів рішення складних задач на основі декомпозиції.

**Завдання:**

1. Провести аналіз виразу та виконати декомпозицію на елементарні вирази
2. Визначити елементарні функції
3. Визначити функцію для виразу на основі виклику елементарних функцій



Короткі теоретичні відомості

Мова програмування F# позиціонується як мова, яка підтримує декілька парадигм: функціональне програмування, імперативне програмування, об’єктно-орієнтоване програмування. Однак в основі мови покладені функціональні принципи програмування: відсутність змінних, відсутність побічного ефекту, застосування функцій у якості даних, відкладені обчислення та ін.

Мова програмування F# має ряд переваг:

1. Підтримка всіх найбільш важливих властивостей функціонального програмування:
   * незмінність даних;
   * можливість композиції функції;
   * використання функцій у якості даних;
   * відкладені обчислення;
   * співставлення зі зразком (pattern matching).
2. Інтероперабельність з іншими мовами програмування на платформі .Net.
3. Універсальність мови, підтримка об’єктно-орієнтованого і імперативного стилів програмування.
4. Постійне вдосконалення мови за рахунок підтримки Microsoft.

Мова F# відноситься до мов зі статичної типізацією.

Для створення будь-якого ідентифікатора і надання йому значення у мові F# передбачається операція зв’язування **let.**

Функціональний стиль програмування в цілому тотожний математичному рішенню задачі.

Одним з найбільш важливих в програмуванні є можливість здійснити циклічні операції. В мові F# є оператори циклу, однак вони відносяться до імперативних властивостей мови. У функціональному програмуванні циклічні операції виконуються за допомогою рекурсії, що відповідає і математичному, і функціональному сенсу рекурсії. У якості прикладу розглянемо функцію розрахунку факторіала.

Математично факторіал задається функцією:



при обмеженнях: 

Тоді на мові F# функція визначення факторіалу в функціональному стилі має наступний вигляд:

let rec fact n =

if n = 1 then 1

else n \* fact (n-1)

Ключове слово rec використовується для вказівки компілятору на рекурсивність процесу.

**Рішення**

Аналіз виразу та декомпозиція на елементарні вирази, визначення елементарних та основної функцій.

\arcsin x = \sum^{\infin}_{n=0} \frac{(2n)!}{4^n (n!)^2 (2n+1)} x^{2n+1}

Необхідно реалізувати функцію для знаходження значення ascsin x через степеневий ряд.

Виконаємо декомпозицію цієї функції.

Спочатку виділимо основні математичні функції, що нам знадобляться:

* let rec fact x – знаходження факторіалу числа х
* pow x n – знаходження степені n числа х
* pow2 х – піднесення числа х у другу степінь (звичайно, для цього можна використовувати і функцію pow, але можна написати більш конкретну функцію для цієї задачі, що підвищить її продуктивність та зрозумілість).

Необхідна функція – це сума ряду. Тому очевидно що нам потрібна функція для знаходження конкретного елемента ряду а також функція для знаходження суми.

Функція знаходження елемента ряду:

let step x n =

    fact (n \* 2) / (pow 4. n \* (fact >> pow2) n) \*

    (fun k -> (pow x k) / float k)(n \* 2 + 1)

Тут використовуємо оператор композиції функцій fact >> pow2 так лямбда функцію для більшої зрозумілості та гнучкості функції.

Тепер залишається лише знайти суму ряду. Зрозуміло, що для цього ми маємо знайти певну кількість членів ряду і знайти їх суму. Будемо знаходити члени ряду від n = 0 збільшуючи n до тих пір, поки значення наступного кроку не стане менше певної похибки (оптимальніше буде продовжувати до тих пір поки менше похибки не буде різниця останнього і попереднього результатів, але в даній задачі будемо продовжувати поки член ряду не буде менше похибки).

Також є як мінімум два варіанти: включити знаходження суми в саму функцію знаходження arcsin, або написати більш загальну функцію для знаходження суми і в функції arcsin знайти суму ряду використовуючи її (тобто функцію, що отримує фукнції step – знаходження конкретного значення члену ряду, початкове значення n та похибку), але так як в даній задачі необхідно знайти лише arcsin то буде краще використати перший варіант (так як другий варіант потребує додаткового аналізу)

Таким чином напишемо функцію arcsin x n eps . Але так як, з практичної точки зору незручно кожен раз вказувати похибку (скоріше всього, тому, хто буде її використовувати похибка не буде грати ролі), то розіб’ємо її на дві функції – одна з похибкою, а друга буде ховати її в середні.

Текс програми

let rec pow x n =

    if n = 0 then 1.

    else x \* pow x (n - 1)

let pow2 x = x \* x

let rec fact x =

    if x <= 1 then 1.

    else float x \* fact (x - 1)

let step x n =

    fact (n \* 2) / (pow 4. n \* (fact >> pow2) n) \*

    (fun k -> (pow x k) / float k)(n \* 2 + 1)

let rec \_arcsin x n eps =

    let v = step x n

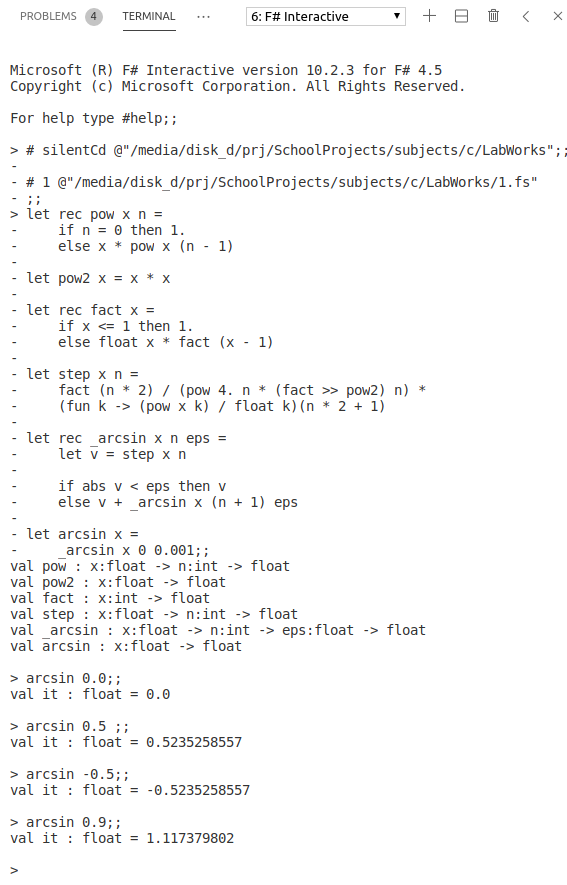
    if abs v < eps then v

    else v + \_arcsin x (n + 1) eps

let arcsin x =

    \_arcsin x 0 0.001

Результат виконання програми



**Висновок:** на цій лабораторній роботі я отримав навички програмування на мові F# використовуючи принципи функціонального програмування; засвоїв методи рішення складних задач на основі декомпозиції.