**Лабораторна робота №3**

**Функціонали**

**Цілі роботи**: отримання навичок використання функціоналів і операторів композиції функції.

**Завдання:**

Відповідно завдання з лабораторної роботи №2 скласти функцію для обробки списку з використанням функціоналів і операторів композиції функції.



Короткі теоретичні відомості

У функціональних мовах програмування, внаслідок побудови на основі лямбда-числення, функції розглядаються як дані і можуть бути використані в якості фактичних параметрів при виклику функції. Такі параметри отримали назву функціональних, а функції з функціональним параметром називаються функціоналами (які зазвичай також називають функціями вищого порядку).

Функціонали широко використовуються при обробки списків. Всі складні операції відносно списків зводяться до трьох базових: *відображення, фільтрація і згортка*.

Операція відображення map застосовує деяку функцію F для кожного елементу списку. Якщо функція F має тип ‘a -> ‘b, тоді функція map застосовується до списку типу ‘a list і повертає тип ‘b list.

Операція фільтрації filter дозволяє залишити у списку тільки ті елементи, які відповідають тестової функції. Таким чином, тестова функція застосовується відносно кожного елементу списку і, якщо елемент задовольняє умовам тесту, він залишається у списку

Операція згортки fold/reduce застосовується тоді, коли необхідно отримати зі списку деякий інтегральний результат (мінімальний або максимальний елементи, сума, добуток і т.п.). Згортка заміняє циклічну обробку списку з акумулятором. З урахування того, що у функціональному програмуванні відсутні змінні, традиційне рішення з акумулятором неможливе. Тоді використовується деякий стан, який передається через ланцюг викликів функції для застосування відносно кожного елементу. Функція згортки приймає на вході це значення і черговий елемент списку і повертає новий стан.

**Рішення**

Аналіз задачі

Замість функції *productOfNegs* реалізуємо фукцнію *product*, яка буде знаходити добуток елементів списку, що задовольняють певній умові (хоча у реальній задачі, можливо, треба подумати чи дійсно виконувати фільтрацію повинна функция *product)*.

Завдяки цьому функція стане значно гнучкіше – ми зможемо знаходити добуток як негативних елементів списку так і тих, що відповідають будь-якій умові.

Функція *product* буде отримувати два параметри – предикат (функція, що буде отримувати елемент списку і перевіряти, чи треба його включати до результату) та список. При цьому предикат розмістимо першим, що було можливим створювати карировані функції.

У функції *product* спочатку виконаємо фільтрацію а потім згортку через функцію fold. При цьому використаемо оператор композиції функцій (завдяки «функціональності» мови F# ми можемо представити реалізацію *product* просто через композицію функцій, тому сама функція навіть не містить явно список у параметрах).

Текс програми

let product predicate =

   List.filter predicate >> (List.fold (\*) 1)

let productOfNegs = product (fun x -> x < 0)

let productOfOdds = product (fun x -> x % 2 <> 0)

productOfNegs [-1;-2;-3]

productOfNegs [2;3;-2;-5]

productOfNegs [1;2;3;-5;4]

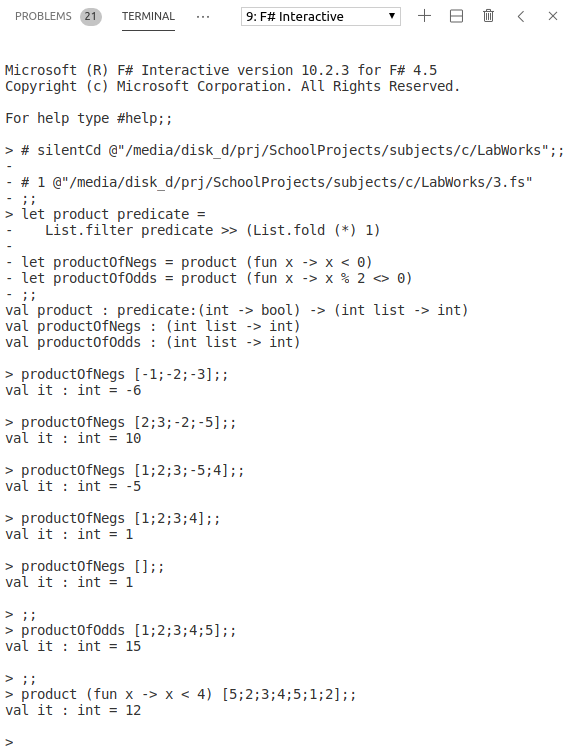
productOfNegs [1;2;3;4]

productOfNegs []

productOfOdds [1;2;3;4;5]

product (fun x -> x < 4) [5;2;3;4;5;1;2]

Результат виконання програми



**Висновок:** на цій лабораторній роботі я отримав навички використання функціоналів і операторів композиції функції.