Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

Лабораторна робота №2

Тема: «Функціональне програмування»

Виконав

студент групи ІТ-03:

Яремчук Д. В.

Перевірив:

ас. Очеретяний О. К.

Київ 2022

Мета роботи: ознайомитися з поняттям функціонального програмування, його принципами та мовою функціонального програмування SML of New Jersey

Хід роботи:

1. Завдання:

   1. Напишіть функцію is_older, яка приймає дві дати та повертає значення true або false. Оцінюється як true, якщо перший аргумент - це дата, яка раніша за другий аргумент. (Якщо дві дати однакові, результат хибний.)

   Код:

```
fun is_older(date1: int*int*int, date2:int*int*int) =
    if (#1 date1) < (#1 date2)
    then true
    else if (#1 date1) > (#1 date2)
    then false
    else
        if (#2 date1) < (#2 date2)
        then true
        else if (#2 date1) > (#2 date2)
        then false
        else
            if (#3 date1) < (#3 date2)
            then true
            else false

val test1 = is_older((2021, 5, 21),(2021, 5, 21))
val test2 = is_older((2021, 5, 21),(2021, 5, 22))
val test3 = is_older((2021, 5, 23),(2021, 5, 22))
```

   Результат:

```
- use "func1.sml";
[opening func1.sml]
val is_older = fn : (int * int * int) * (int * int * int) -> bool
val test1 = false : bool
val test2 = true : bool
val test3 = false : bool
val it = () : unit
```

   Пояснення: за допомогою послідовної перевірки року, місяця та дня ми визначаємо, яка із дат є старішою

2. Завдання:

2. Напишіть функцію number_in_month, яка приймає список дат і місяць (тобто int) і повертає скільки дат у списку в даному місяці.

Код:

```
fun number_in_month(dates_list : (int*int*int) list, month : int) =
    if null dates_list
    then 0
    else
        if (#2 (hd dates_list)) = month
        then 1 + number_in_month(tl dates_list, month)
        else 0 + number_in_month(tl dates_list, month)

val test1 = number_in_month([(2021, 10, 21),(2022, 10, 22), (2022, 9, 22),
                             (2022, 11, 22), (2022, 10, 20)], 10)
val test2 = number_in_month([], 10)
```

Результат:

```
- use "func2.sml";
[opening func2.sml]
val number_in_month = fn : (int * int * int) list * int -> int
val test1 = 3 : int
val test2 = 0 : int
val it = () : unit
```

Пояснення: спочатку визначаємо чи закінчився список дат (умова виходу з рекурсії). Далі ми визначаємо чи дорівнює місяць поточного елементу місяцю, який нам треба. Якщо так, то ми додаємо 1 і продовжуємо ітеруватись, якщо ні, то нічого не додаємо і продовжуємо ітеруватись по списку дат.

3. Завдання:

3. Напишіть функцію number_in_months, яка приймає список дат і список місяців (тобто список int) і повертає кількість дат у списку дат, які знаходяться в будь-якому з місяців у списку місяців. **Припустимо, що в списку місяців немає повторюваних номерів**. Підказка: скористайтеся відповіддю до попередньої задачі.

Код:

```
fun number_in_months(dates_list : (int*int*int) list, month : int list) =
    if null month
    then 0
    else number_in_month(dates_list, hd month) + number_in_months(dates_list, tl month)

val test1 = number_in_months([(2021, 10, 21),(2022, 10, 22), (2022, 9, 22),
                              (2022, 11, 22), (2022, 10, 20)], [9, 10])
val test2 = number in months([],[9, 10])
```

Результат:

```
- use "func3.sml";
[opening func3.sml]
val number_in_month = fn : (int * int * int) list * int -> int
val number_in_months = fn : (int * int * int) list * int list -> int
val test1 = 4 : int
val test2 = 0 : int
val it = () : unit
```

Пояснення: спочатку визначаємо чи закінчився список місяців (умова виходу з рекурсії). Далі ми визначаємо місяць поточної дати і додаємо його до всіх інших (за допомогою рекурсії).

4. Завдання:

> 4. Напишіть функцію dates_in_month, яка приймає список дат і число місяця (тобто int) і повертає список, що містить дати з аргументу "список дат", які знаходяться в переданому місяці. Повернутий список повинен містять дати в тому порядку, в якому вони були надані спочатку.

Код:

```
fun dates_in_month(dates_list : (int*int*int) list, month : int) =
    if null dates_list
    then []
    else
        if (#2 (hd dates_list)) = month
        then (hd dates_list) :: dates_in_month(tl dates_list, month)
        else dates_in_month(tl dates_list, month)

val test1 = dates_in_month([(2021, 10, 21),(2022, 10, 22), (2022, 9, 22),
                            (2022, 11, 22), (2022, 10, 20)], 10)
val test2 = dates_in_month([], 10)
```

Результат:

```
- use "func4.sml";
[opening func4.sml]
val dates_in_month = fn :
  (int * int * int) list * int -> (int * int * int) list
val test1 = [(2021,10,21),(2022,10,22),(2022,10,20)] : (int * int * int) list
val test2 = [] : (int * int * int) list
val it = () : unit
```

Пояснення: спочатку визначаємо чи закінчився список дат (умова виходу з рекурсії). Далі ми визначаємо чи дорівнює місяць поточного елементу місяцю, який нам треба. Якщо так, то ми додаємо цю дату до списка і продовжуємо ітеруватись, якщо ні, то нічого не додаємо і продовжуємо ітеруватись по списку дат.

5. Завдання:

Напишіть функцію dates_in_months, яка приймає список дат і список місяців (тобто список int) і повертає список, що містить дати зі списку аргументів дат, які знаходяться в будь-якому з місяців у списку місяців. Для простоти, припустимо, що

в списку місяців немає повторюваних номерів. Підказка: Використовуйте свою відповідь на попередню задачу та оператор додавання списку SML (@).

Код:

```
fun dates_in_months(dates_list : (int*int*int) list, month : int list) =
    if null month
    then []
    else dates_in_month(dates_list, hd month) :: dates_in_months(dates_list, tl month)

val test1 = dates_in_months([(2021, 10, 21),(2022, 10, 22), (2022, 9, 22),
                (2022, 11, 22), (2022, 10, 20)], [9, 10])
val test2 = dates_in_months([], [9, 10])
```

Результат:

```
- use "func5.sml";
[opening func5.sml]
val dates_in_month = fn :
  (int * int * int) list * int -> (int * int * int) list
val dates_in_months = fn :
  (int * int * int) list * int list -> (int * int * int) list list
val test1 = [[(2022,9,22)],[(2021,10,21),(2022,10,22),(2022,10,20)]] :
  (int * int * int) list list
val test2 = [[],[]] : (int * int * int) list list
val it = () : unit
```

Пояснення: спочатку визначаємо чи закінчився список місяців (умова виходу з рекурсії). Далі ми визначаємо дати поточного місяця і додаємо його до всіх інших (за допомогою рекурсії).

6. Завдання:

   6. Напишіть функцію get_nth, яка приймає список рядків і int n та повертає n-й елемент списку, де голова списку є першим значенням. Не турбуйтеся якщо в списку занадто мало елементів; у цьому випадку ваша функція може навіть застосувати hd або tl до порожнього списку, і це нормально.

Код:

```
fun get_nth(strings : string list, n : int) =
    if null strings
    then "too small string list"
    else if n = 1
    then hd(strings)
    else get_nth(tl strings, n - 1)

val test1 = get_nth(["a", "b", "c", "d", "e", "f", "g", "h"], 5)
val test2 = get_nth(["a", "b", "c", "d", "e", "f", "g", "h"], 10);
```

Результат:

```
- use "func6.sml";
[opening func6.sml]
val get_nth = fn : string list * int -> string
val test1 = "e" : string
val test2 = "too small string list" : string
val it = () : unit
```

Пояснення: спочатку перевіряємо чи закінчився список по якому ми ітеруємося (за допомогою рекурсії). Далі ми перевіряємо чи дійшли ми до необхідного елемента і якщо так, то повертаємо його. Інакше, ми ітеруємося далі.

7. Завдання:

7. Напишіть функцію date_to_string, яка приймає дату і повертає рядок у вигляді “February 28, 2022” Використовуйте оператор ^ для конкатенації рядків і бібліотечну функцію Int.toString для перетворення int в рядок. Для створення частини з місяцем не використовуйте купу розгалужень. Замість цього використайте список із 12 рядків і свою відповідь на попередню задачу. Для консистенції пишіть кому після дня та використовуйте назви місяців англійською мовою з великої літери.

Код:

```
val months_strings = ["January", "February", "March", "April", "May", "June",
                      "July", "August", "September", "October", "November", "December"]
fun date_to_string(date: int*int*int) =
    let
        val month = get_nth(months_strings, (#2 date))
        val result = month ^ " " ^ Int.toString(#3 date) ^ ", " ^ Int.toString(#1 date)
    in result
    end

val test1 = date_to_string((2022, 10, 21))
val test2 = date_to_string((2021, 5, 28))
```

Результат:

```
- use "func7.sml";
[opening func7.sml]
val get_nth = fn : string list * int -> string
val months_strings =
  ["January","February","March","April","May","June","July","August",
   "September","October","November","December"] : string list
val date_to_string = fn : int * int * int -> string
val test1 = "October 21, 2022" : string
val test2 = "May 28, 2021" : string
val it = () : unit
```

Пояснення: спочатку отримуємо місяць передаючи список всіх місяців, а також необхідний номер місяця в функцію, описану в попередньому пункті. Далі конкатенуємо все до купи за допомогою оператора ^.

8. Завдання:

8. Напишіть функцію number_before_reaching_sum, яка приймає додатний int під назвою sum, та список int, усі числа якої також додатні. Функція повертає int. Ви повинні повернути значення int n таке, щоб перші n елементів списку в сумі будуть менші sum, але сума значень від n + 1 елемента списку до кінця був більше або рівний sum.

Код:

```
fun number_before_reaching_sum(sum : int, num_list : int list) =
    if sum - (hd num_list) < 0
    then 0
    else 1 + number_before_reaching_sum(sum - (hd num_list), tl num_list)

val test1 = number_before_reaching_sum(11, [10,10,10,10])
val test2 = number_before_reaching_sum(22, [1,2,3,4,5,10]);
val test3 = number_before_reaching_sum(22, [1,2,3,4,5,6])
```

Результат:

```
- use "func8.sml";
[opening func8.sml]
val number_before_reaching_sum = fn : int * int list -> int
val test1 = 1 : int
val test2 = 5 : int

uncaught exception Empty
  raised at: smlnj/init/pervasive.sml:193.19-193.24
             ../compiler/TopLevel/interact/interact.sml:56.32-56.36
-
```

Пояснення: спочатку ми віднімаємо від суми (буде зменшуватись з кожною ітерацією) наступний елемент списку і перевіряємо чи вона менше нуля (це означає, що ми знайшли необхідний елемент). Якщо так,

то повертаємо необхідне значення. Якщо ні, то продовжуємо ітеруватись по списку. В разі якщо ми не знайшли необхідний елемент (сума всіх елементів масиву менша від заданої), то з'являється помилка

9. Завдання:

9. Напишіть функцію what_month, яка приймає день року (тобто int між 1 і 365) і повертає в якому місяці цей день (1 для січня, 2 для лютого тощо). Використовуйте список, що містить 12 цілих чисел і вашу відповідь на попередню задачу.

Код:

```
val month_days = [(1, 31), (2, 28), (3, 31), (4, 30), (5, 31), (6, 30),
                  (7, 31), (8, 31), (9, 30),(10, 31), (11, 30), (12, 31)];

fun what_month(day_of_year: int, months: (int*int) list) =
    if day_of_year - (#2 (hd months)) <= 0
    then (#1 (hd months))
    else what_month(day_of_year - (#2 (hd months)), tl months)

val test1 = what_month(143, month_days)
val test2 = what_month(90, month_days);
```

Результат:

```
- use "func9.sml";
[opening func9.sml]
val month_days =
  [(1,31),(2,28),(3,31),(4,30),(5,31),(6,30),(7,31),(8,31),(9,30),(10,31),
   (11,30),(12,31)] : (int * int) list
val what_month = fn : int * (int * int) list -> int
val test1 = 5 : int
val test2 = 3 : int
val it = () : unit
```

Пояснення: Ми ітеруємось по місяцям і визначаємо чи є наступний місяць таким, що якщо відняти від дня року (на кожній ітерації він зменшується на кількість днів у поточному місяці) кількість днів у наступному місяці, то це буде менше 0.

10. Завдання:

10. Напишіть функцію month_range, яка приймає два дні року day1 і day2 і повертає список int [m1,m2,...,mn] де m1 – місяць day1, m2 – місяць day1+1, ..., а mn – місяць day2. Зверніть увагу, що результат матиме довжину day2 - day1 + 1 або довжину 0, якщо day1>day2.

Код:

```
val month_days = [(1, 31), (2, 28), (3, 31), (4, 30), (5, 31), (6, 30),
                  (7, 31), (8, 31), (9, 30),(10, 31), (11, 30), (12, 31)];

fun month_range(day1 : int, day2: int, months: (int*int) list) =
    if day1 > day2
    then []
    else what_month(day1, months) :: month_range(day1 + 1, day2, months)

val test1 = month_range(52, 63, month_days)
val test2 = month_range(79, 50, month_days)
```

Результат:

```
- use "func10.sml";
[opening func10.sml]
val what_month = fn : int * (int * int) list -> int
val month_days =
  [(1,31),(2,28),(3,31),(4,30),(5,31),(6,30),(7,31),(8,31),(9,30),(10,31),
   (11,30),(12,31)] : (int * int) list
val month_range = fn : int * int * (int * int) list -> int list
val test1 = [2,2,2,2,2,2,2,2,3,3,3,3] : int list
val test2 = [] : int list
val it = () : unit
```

Пояснення: спочатку перевіряємо чи ми дійшли до останнього дня (умова виходу з рекурсії). Якщо ні, то ми далі ітеруємось додаючи на кожній ітерації номер місяця до списку.

11. Завдання:

    ~~якщо day1 = day2:~~

    11. Напишіть найстарішу функцію, яка бере список дат і оцінює параметр (int*int*int). Він має оцінюватися як NONE, якщо список не містить дат, і SOME d, якщо дата d є найстарішою датою у списку.

Код:

```
fun the_oldest_date(dates : (int*int*int) list) =
    if null dates
    then NONE
    else
        let val min = the_oldest_date(tl dates)
        in if isSome min andalso is_older(valOf min, hd dates)
            then min
            else SOME (hd dates)
        end

val test1 = the_oldest_date([(2022, 10, 21), (2022, 5, 12), (2022, 7, 10), (2022, 11, 5)])
val test2 = the_oldest_date([])
```

Результат:

```
- use "func11.sml";
[opening func11.sml]
val is_older = fn : (int * int * int) * (int * int * int) -> bool
val the_oldest_date = fn : (int * int * int) list -> (int * int * int) option
val test1 = SOME (2022,5,12) : (int * int * int) option
val test2 = NONE : (int * int * int) option
val it = () : unit
```

Пояснення: перевіряємо чи закінчились дати по яким ми ітеруємося (умова виходу з рекурсії). Далі за допомогою рекурсії ми оновлюємо найменшу дату на кожній ітерації, порівнюючи з поточною датою. Все це відбувається в оберненому порядку, тобто ми починаємо з останньої дати і йдемо наперед порівнюючи поточну найменшу дату з поточним елементом списку.

Висновки: в ході лабораторної роботи ми ознайомились із поняттям функціонального програмування. Ми закріпили вивчення мови SML of New Jersey виконавши завдання із опрацювання дат.