**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

КАФЕДРА «КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ»

**Ю.Ю ГОРЮНОВ, Т.Ю. ЛЕОНОВА**

ПРОГРАММИРОВАНИЕ НА **VISUAL F#**

лабораторный практикум

Вариант 9

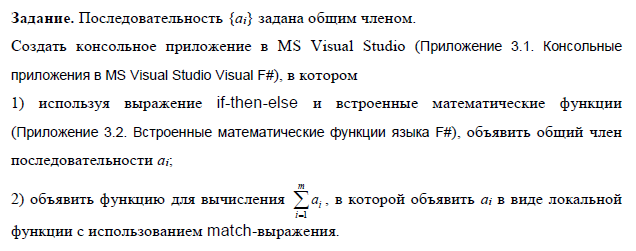
Исполнитель:

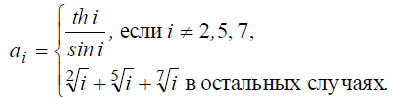
студент группы

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ И. О. Фамилия

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

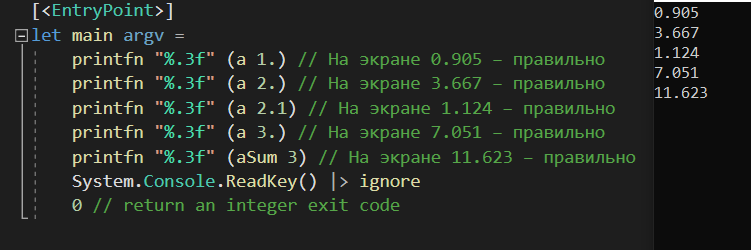
**Пенза 2021**





Для нахождения суммы была реализована рекурсивная функция вместо циклов (чтобы лишний раз не нарушать концепцию функционального программирования).

Демонстрация работы:



Исходный код:

open System

let a i =

match i with

| 2. | 5. | 7. -> sqrt i + i \*\* (1. / 5.) + i \*\* (1. / 7.)

| \_ -> tanh i / sin i

let funcSum l r f =

let rec funcSum' i acc=

if i > r then acc

else funcSum' (i + 1.) (acc + f i)

funcSum' l 0.

let aSum m =

let a i =

match i with

| 2. | 5. | 7. -> sqrt i + i \*\* (1. / 5.) + i \*\* (1. / 7.)

| \_ -> tanh i / sin i

funcSum 1. (float m) a

[<EntryPoint>]

let main argv =

printfn "%.3f" (a 1.) // На экране 0.905 – правильно

printfn "%.3f" (a 2.) // На экране 3.667 – правильно

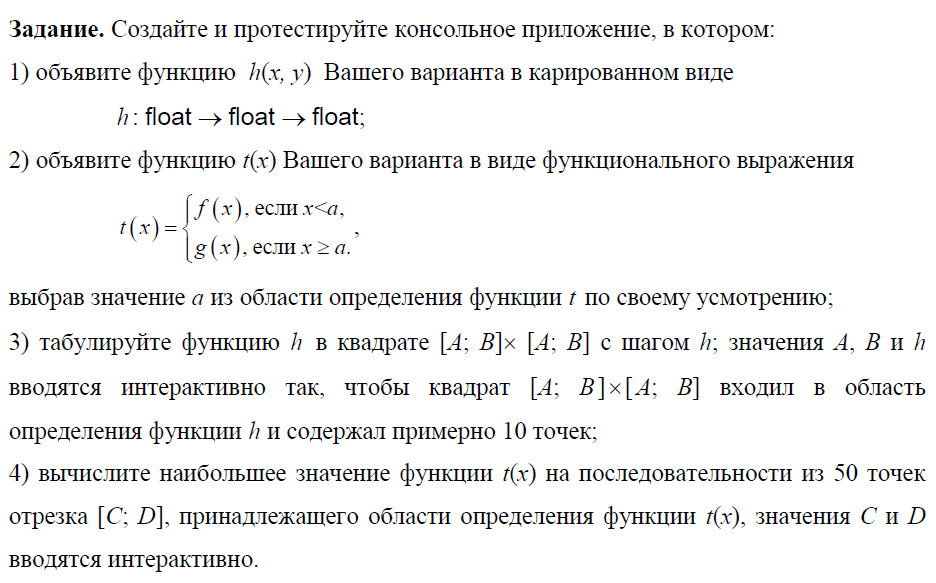
printfn "%.3f" (a 2.1) // На экране 1.124 – правильно

printfn "%.3f" (a 3.) // На экране 7.051 – правильно

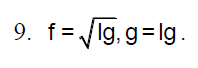
printfn "%.3f" (aSum 3) // На экране 11.623 – правильно

System.Console.ReadKey() |> ignore

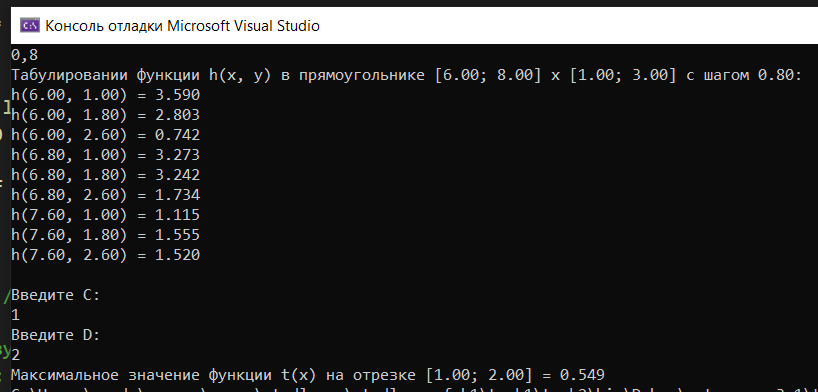
0 // return an integer exit code







Т. к. ни один квадрат [A; B] x [A; B] не входит в область определения функции h(x, y) 9-го варианта, то было решено заменить область табулирования на прямоугольник [A1; B1] x [A2; B2] (также может быть квадратом). Остальные комментарии есть в исходном коде.

Демонстрация работы:

Исходный код:

open System

let h x y =

(x - 2. \* y) \*\* (cos x \* cos x + log10 y)

let t x =

let f x = x |> log10 |> sqrt

let g x = log10 x

let a = 20.

if x < a then f x

else g x

let EPS = 0.000001 // Допустимая погрешность

(\* Т. к. не существует квадрата [A; B] x [A; B], входящего в D(h),

то ф-ей tabFunc используется прямоугольник [A1; B1] x [A2; B2] (м. б. квадратом)\*)

let tabFunc f a1 b1 a2 b2 inc =

let rec tabFunc' i j =

if (i <= b1 + EPS && j <= b2 + EPS) then

printf "h(%.2f, %.2f) = %.3f\n" i j (f i j)

tabFunc' i (j + inc)

else if (i <= b1 && j > b2) then

tabFunc' (i + inc) a2

tabFunc' a1 a2

let findMax f l r =

let inc = (r - l) / 49.

let rec findMax' i max =

if i > r + EPS then max

else

let cur = f i

if cur > max then findMax' (i + inc) cur

else findMax' (i + inc) max

findMax' l (f l)

let enterFloat name =

printf "Введите %s:\n" name

let s = System.Console.ReadLine()

System.Convert.ToDouble s

[<EntryPoint>]

let main argv =

let a1 = enterFloat "A1"

let b1 = enterFloat "B1"

let a2 = enterFloat "A2"

let b2 = enterFloat "B2"

let inc = enterFloat "шаг d"

printf "Табулировании функции h(x, y) в прямоугольнике [%.2f; %.2f] x [%.2f; %.2f] с шагом %.2f:\n" a1 b1 a2 b2 inc

tabFunc h a1 b1 a2 b2 inc

printf "\n"

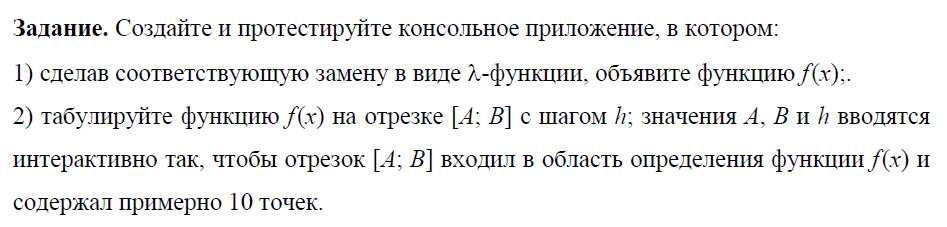
let c = enterFloat "C"

let d = enterFloat "D"

printf "Максимальное значение функции t(x) на отрезке [%.2f; %.2f] = %.3f" c d (findMax t c d)

System.Console.ReadKey() |> ignore

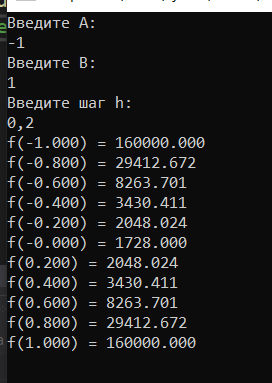
0 // return an integer exit code

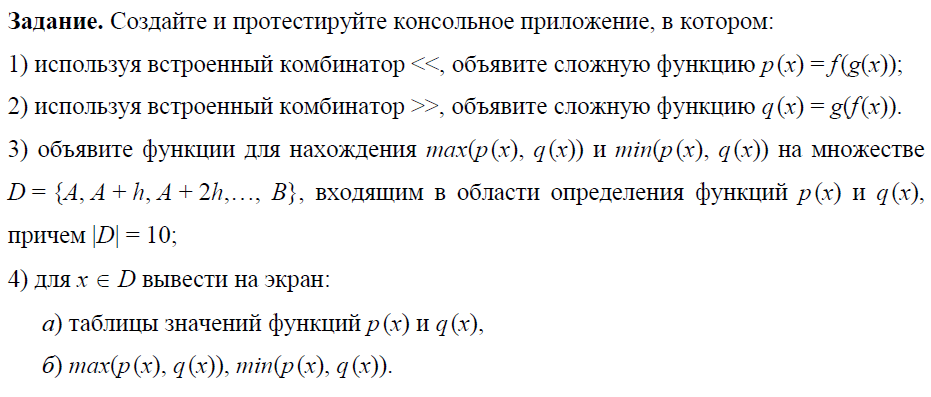




Т. к. область определения исходной функции была пустым множеством (основание действительной степени всегда отрицательно), то минус во внешней скобке (после числа 3) был заменен на плюс.

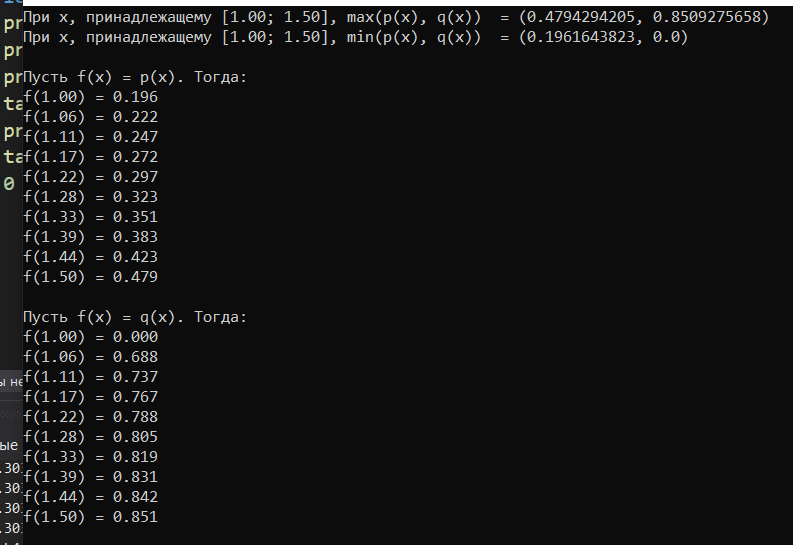
Демонстрация работы:







Демонстрация работы:



Исходный код:

open System

let f = log10 >> sqrt

let g x = (tan x) \*\* (1. / 5.)

let p = f << g

let q = f >> g

let EPS = 0.000001

let findMaxOrMin op f l r inc =

let rec findMaxOrMin' i m =

if i > r + EPS then m

else

let cur = f i

if (op cur m) then findMaxOrMin' (i + inc) cur

else findMaxOrMin' (i + inc) m

findMaxOrMin' l (f l)

let findMaxPair f g l r inc =

let findMax h = findMaxOrMin (>) h l r inc

(findMax f, findMax g)

let findMinPair f g l r inc =

let findMin h = findMaxOrMin (<) h l r inc

(findMin f, findMin g)

let tabFunc f l r inc =

let rec tabFunc' i =

if i <= r + EPS then

printf "f(%.2f) = %.3f\n" i (f i)

tabFunc' (i + inc)

tabFunc' l

[<EntryPoint>]

let main argv =

let a = 1.

let b = 1.5

let inc = (b - a) / 9.

printf "При x, принадлежащему [%.2f; %.2f], max(p(x), q(x)) = %A\n" a b (findMaxPair p q a b inc)

printf "При x, принадлежащему [%.2f; %.2f], min(p(x), q(x)) = %A\n" a b (findMinPair p q a b inc)

printf "\nПусть f(x) = p(x). Тогда:\n"

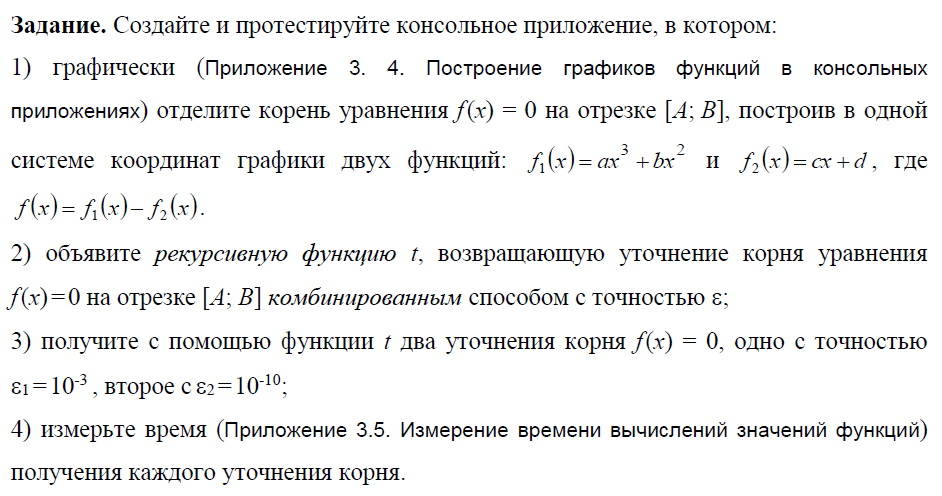
tabFunc p a b inc

printf "\nПусть f(x) = q(x). Тогда:\n"

tabFunc q a b inc

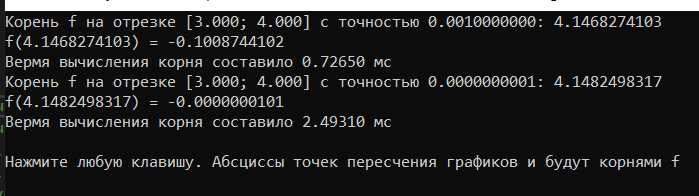
System.Console.ReadKey() |> ignore

0 // return an integer exit code

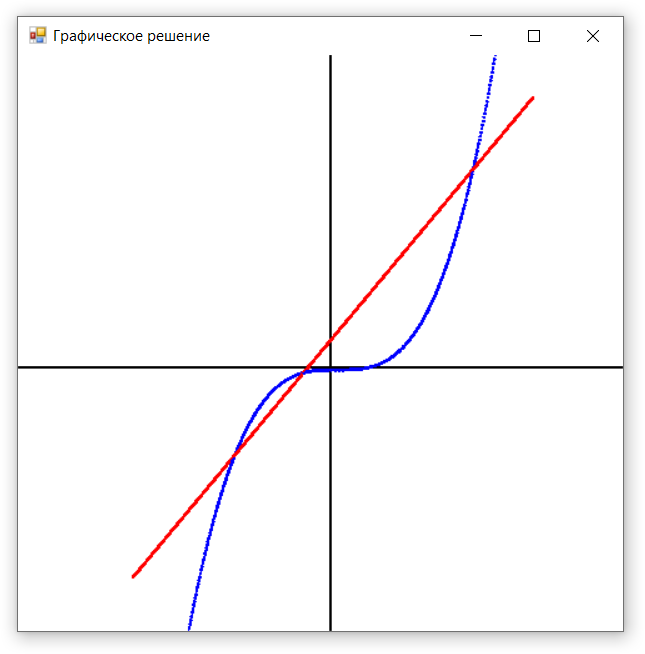




Демонстрация работы:



Демонстрация графического решения была реализована путем рисования функций f1 и f2, абсциссы точек их пересечения и есть корни функции f. Масштаб по оси OX был растянут в 20 раз:



Исходный код:

open System

open System.Windows.Forms

open System.Drawing

let f1 x = 2. \* x \* x \* x - x \* x

let f2 x = 24. \* x + 26.

let f x = f1 x - f2 x

let f' x = 6. \* x \* x - 2. \* x - 24. // 1-я производная f

let f'' x = 12. \* x - 2. // 2-я производная f

let EPS1 = 1. / 1E3

let EPS2 = 1. / 1E10

let rec findRoot f f' f'' a b eps =

if abs (a - b) <= 2. \* eps then

(a + b) / 2.

else

let mutable \_a = a

let mutable \_b = b

if f a \* f'' a < 0. then \_a <- a - f a / f' a / 10. // Делим на 10, т. к. ф-я слишком резкая

if f b \* f'' b < 0. then \_b <- b - f b / f' b / 10.

findRoot f f' f'' \_a \_b eps

[<EntryPoint>]

let main argv =

let a = 3.

let b = 4.

let stopWatch2 = System.Diagnostics.Stopwatch.StartNew()

let root2 = (findRoot f f' f'' a b EPS2)

stopWatch2.Stop()

let stopWatch1 = System.Diagnostics.Stopwatch.StartNew()

let root1 = (findRoot f f' f'' a b EPS1)

stopWatch1.Stop()

printf "Корень f на отрезке [%.3f; %.3f] с точностью %.10f: %.10f\n" a b EPS1 root1

printf "f(%.10f) = %.10f\n" root1 (f root1)

printf "Вермя вычисления корня составило %.5f мс\n" stopWatch1.Elapsed.TotalMilliseconds

printf "Корень f на отрезке [%.3f; %.3f] с точностью %.10f: %.10f\n" a b EPS2 root2

printf "f(%.10f) = %.10f\n" root2 (f root2)

printf "Вермя вычисления корня составило %.5f мс\n" stopWatch2.Elapsed.TotalMilliseconds

// Объявляем экземпляр класса «Форма»:

let form = new Form(BackColor = Color.White, Text = "Графическое решение")

let fW = 500.0

let fH = 500.0

form.Width <- int fW // ширина окна формы,

form.Height <- int fH // высота окна формы,

form.Paint.Add( // рисование,

fun drawGr->

// объявляем экземпляр класса “Pen” – карандаш,

let pen=new Pen(Color.Black,Width = 2.0f)

// объявляем экземпляр класса “Brush” – кисть,

let oX = fW / 2.0 // середина ширины окна формы,

let oY = fH / 2.0 // середина высоты окна формы,

drawGr.Graphics.DrawLine(pen, 0.0f, float32 oY, float32 fW, float32 oY) // рисуем ось OX,

drawGr.Graphics.DrawLine(pen, float32 oX, 0.0f, float32 oX, float32 fH) // рисуем ось OY,

for t in -6.0..0.01..6.0 do // строим график функции по точкам,

let x = t \* 30. // Растягиваем масштаб по оси OX

let y1 = f1 t

let brush=new SolidBrush(Color.Blue)

drawGr.Graphics.FillEllipse(brush, float32 (oX + x), float32 (oY - y1), 3.0f, 3.0f) // «точка» графика,

let y2 = f2 t

let brush=new SolidBrush(Color.Red)

drawGr.Graphics.FillEllipse(brush, float32 (oX + x), float32 (oY - y2), 3.0f, 3.0f) // «точка» графика,

)

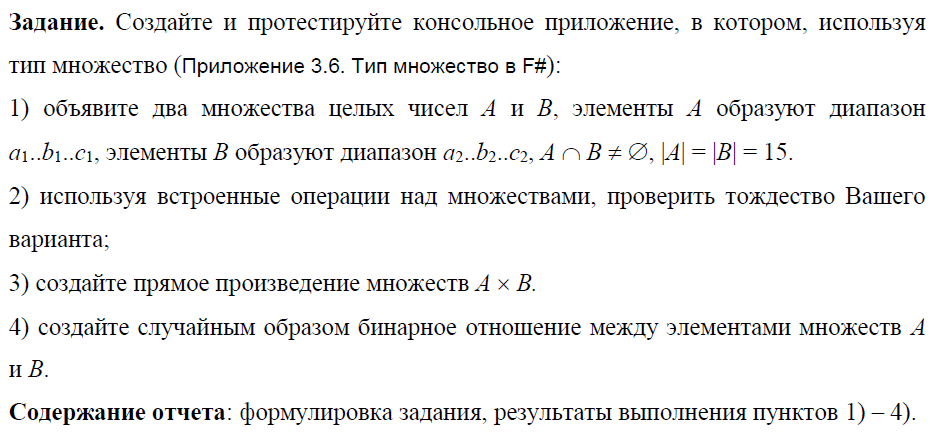
printf "\nНажмите любую клавишу. Абсциссы точек пересчения графиков и будут корнями f\n"

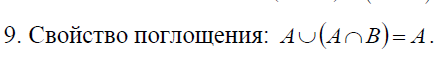
System.Console.ReadKey() |> ignore // ждём в консоли разрешения на построение,

Application.Run(form) // открываем приложение Windows Forms из консольного приложения

System.Console.ReadKey() |> ignore

0 // return an integer exit code





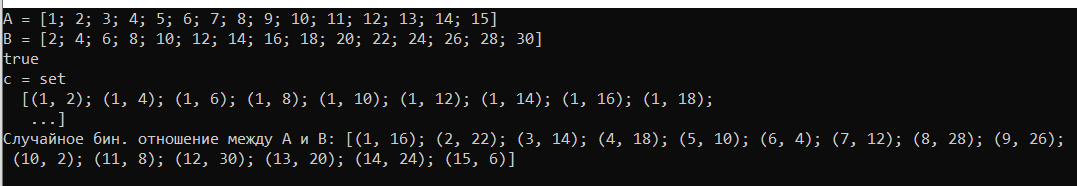
Свойство поглощения было проверено с помощью вывода логического типа, который получался при проверке тождества, изображенного выше, в котором все операции были переведены на F# в соответствии с таблицей из приложения 3.6:

printf "%A\n" (Set.union a (Set.intersect a b) = a)

Случайное бинарное отношение генерируется рекурсивной функцией genBinRelation в виде множества пар, каждая из которой показывает, в какое число из B (2-я позиция пары) переходит число из A (1-я позиция пары).

Декартово произведение генерируется с помощью функции с циклами, аналогично тому, как это демонстрируется в приложении 3.6.

Демонстрация работы:



Исходный код:

open System

let cartesian s1 s2 =

let mutable c = Set.empty

for a in s1 do

for b in s2 do

c <- Set.add (a, b) c

c

let rec genBinRelation (s1 : 't Set) (s2 : 't Set) =

if s1.IsEmpty || s2.IsEmpty then Set.empty

else

let r = Seq.item (System.Random().Next() % s2.Count) s2

let min = s1.MinimumElement

Set.union (Set.singleton (min, r)) (genBinRelation (Set.remove min s1) (Set.remove r s2))

[<EntryPoint>]

let main argv =

let a = set[1..1..15]

let b = set[2..2..30]

printf "A = %A\n" (List.ofSeq a) // Приводим к list, чтобы выводилось полностью

printf "B = %A\n" (List.ofSeq b)

printf "%A\n" (Set.union a (Set.intersect a b) = a)

let c = cartesian a b

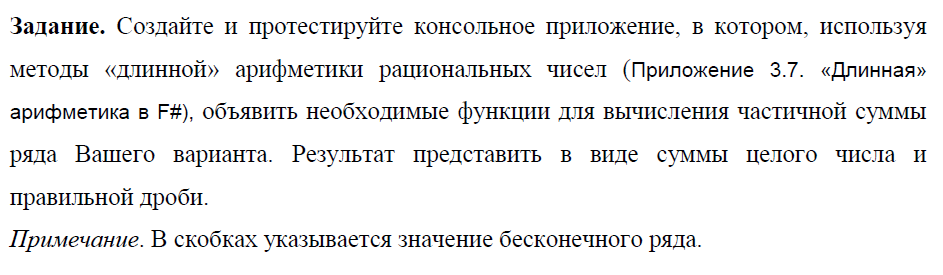
printf "с = %A\n" c

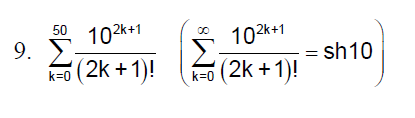
let rel = genBinRelation a b

printf "Случайное бин. отношение между A и B: %A" (List.ofSeq rel)

System.Console.ReadKey() |> ignore

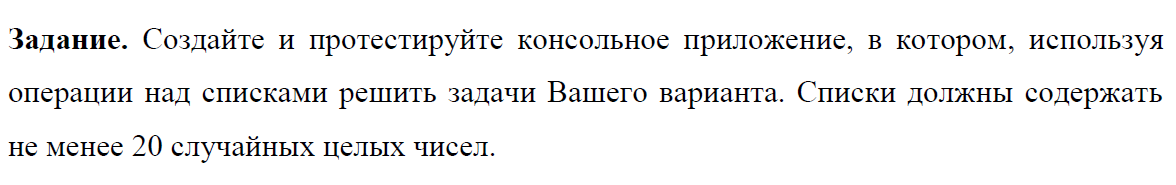
0 // return an integer exit code

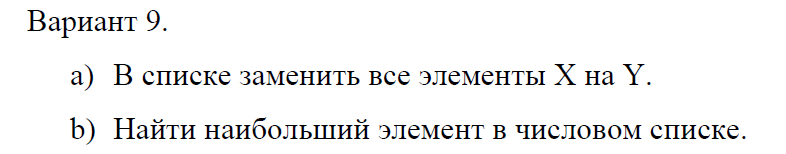




Сумма ряда считается рекурсивной функции, которая принимает текущий индекс и накопленную к текущему моменту времени. При работе выводится на экран частичная сумма ряда и функции, которой равна сумма в пределе (при k стремящемся к бесконечности):

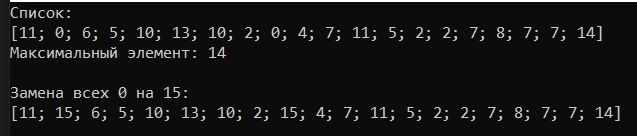






Замена элементов производится с помощью функции List.map, максимальный элемент ищется с помощью функции List.fold (в качестве второго параметра передается минимально возможное число для типа Int32).

Демонстрация работы:



Исходный код:

open System

[<EntryPoint>]

let main argv =

let l = [for x in 1..20 -> Random().Next() % 15]

printf "Список:\n%A\n" l

let x = 0

let y = 15

let maxEl = List.fold (fun x y -> if x > y then x else y) Int32.MinValue l

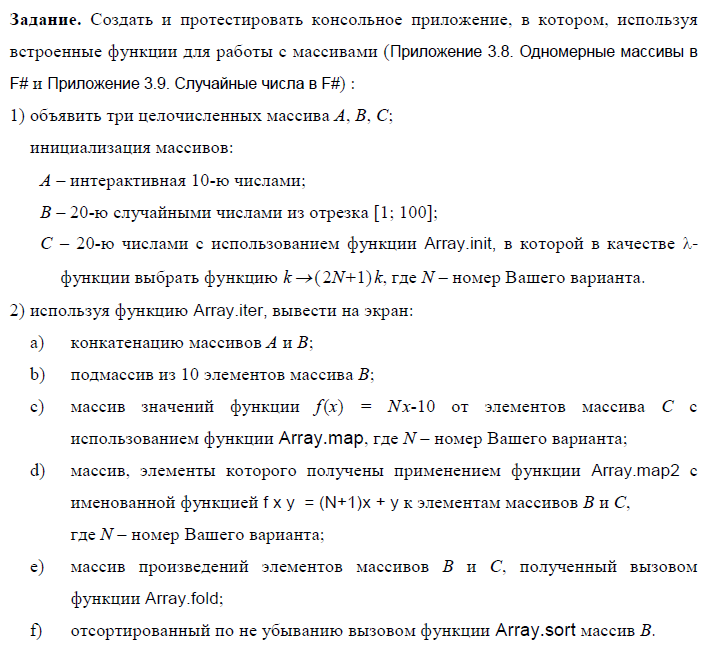
printf "Максимальный элемент: %d\n\n" maxEl

let l1 = List.map (fun elem -> if elem = x then y else elem) l

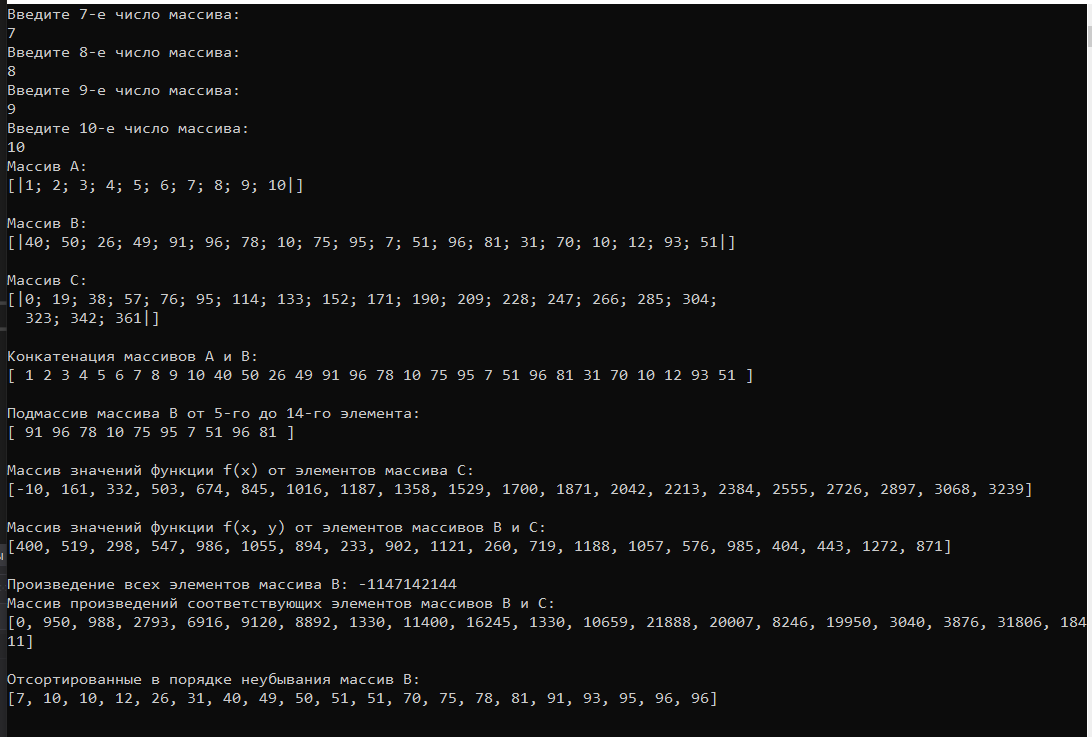
printf "Замена всех %d на %d:\n%A\n" x y l1

System.Console.ReadKey() |> ignore

0 // return an integer exit code



В пункте e) задания 2) вместо Array.fold использовалась функция Array.map2 (для построения массива произведений соответствующих элементов), т. к. не очень понятно, как это задание можно сделать при помощи первой функции. Array.fold же была использована для нахождения произведения всех чисел массива B. Остальные комментарии можно увидеть в исходном коде.

Демонстрация работы:

Исходный код:

open System

let enterInt name =

printf "Введите %s:\n" name

let s = System.Console.ReadLine()

System.Convert.ToInt32 s

let printArray (arr : 't array)=

printf "["

Array.iteri (fun i elem -> printf "%A" elem; if i < arr.Length - 1 then printf ", ") arr

printf "]\n"

[<EntryPoint>]

let main argv =

let N = 9

let a = [|for i in 1..10 -> enterInt (String.Format("{0}-е число массива", i))|]

printf "Массив A:\n%A\n\n" a

let b = [|for i in 1..20 -> Random().Next() % 100 + 1|]

printf "Массив B:\n%A\n\n" b

let c = Array.init 20 (fun k -> (2 \* N + 1) \* k)

printf "Массив C:\n%A\n\n" c

printf "Конкатенация массивов A и B:\n[ "

Array.iter (fun elem -> printf "%d " elem) a

Array.iter (fun elem -> printf "%d " elem) b

printf "]\n\n"

let l = 4

let r = 13

printf "Подмассив массива B от %d-го до %d-го элемента:\n[ " (l + 1) (r + 1)

Array.iteri (fun i elem -> if i >= l && i <= r then printf "%d " elem) b

printf "]\n\n"

printf "Массив значений функции f(x) от элементов массива C:\n"

printArray (Array.map (fun x -> N \* x - 10) c)

printf "\n"

printf "Массив значений функции f(x, y) от элементов массивов B и C:\n"

printArray (Array.map2 (fun x y -> (N + 1) \* x + y) b c)

printf "\n"

printf "Произведение всех элементов массива B: %d\n" (Array.fold (\*) 1 b)

printf "Массив произведений соответствующих элементов массивов B и C:\n"

printArray (Array.map2 (fun x y -> x \* y) b c)

printf "\nОтсортированные в порядке неубывания массив B:\n"

printArray (Array.sort b)

System.Console.ReadKey() |> ignore

0 // return an integer exit code