**Московский государственный технический**

**университет им. Н.Э. Баумана.**

Факультет «Информатика и управление»

Кафедра «Системы обработки информации и управления»

Курс «Основы информатики»

Отчет по лабораторной работе №2

«трек курса «функциональное программирование»»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-34 |  | преподаватель каф. ИУ5 |
| Кравцов А.Н. |  | Ю.Е. Гапанюк. |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2020 г.

1. Создайте два варианта функции, которая возвращает кортеж значений. Первый вариант принимает на вход параметры в виде кортежа, второй вариант параметры в каррированном виде.

2. Выберите простой алгоритм, который может быть реализован в виде рекурсивной функции и реализуйте его в F#. Пример – вычисление суммы целых чисел в заданном диапазоне.

3. Преобразуйте разработанную рекурсивную функцию в форму хвостовой рекурсии.

4. Разработайте конечный автомат из трех состояний и реализуйте его в виде взаимно-рекурсивных функций.

5. Разработайте функцию, которая принимает 3 целых числа и лямбда-выражение для их суммирования в виде кортежа и в каррированном виде.

|  |
| --- |
| open System |
|  | //1. Создайте два варианта функции, которая возвращает кортеж значений. |
|  | //Первый вариант принимает на вход параметры в виде кортежа, |
|  | //второй вариант параметры в каррированном виде. |
|  | //2. Выберите простой алгоритм, который может быть реализован в виде рекурсивной функции |
|  | //и реализуйте его в F#. |
|  | //Пример – вычисление суммы целых чисел в заданном диапазоне. |
|  | //3. Преобразуйте разработанную рекурсивную функцию в форму хвостовой рекурсии. |
|  | //4. Разработайте конечный автомат из трех состояний и |
|  | //реализуйте его в виде взаимно-рекурсивных функций. |
|  | //5. Разработайте функцию, которая принимает 3 целых числа и |
|  | //лямбда-выражение для их суммирования в виде кортежа и в каррированном виде. |
|  |  |
|  |  |
|  | let rec State1(x:int) = |
|  | printfn "%i - (+1) %i" x (x+1) |
|  | let x\_next = x+1 |
|  | if x\_next>3 then State2(x\_next) |
|  | else State1(x\_next) |
|  | and State2(x:int) = |
|  | printfn "%i - (^2) %i" x (x\*x) |
|  | let x\_next = x+1 |
|  | if x\_next>6 then State3(x\_next) |
|  | else State2(x\_next) |
|  | and State3(x:int) = |
|  | printfn "%i - (^3) %i" x (x\*x\*x) |
|  | let x\_next = x+1 |
|  | if x\_next<=10 then State3(x\_next) |
|  |  |
|  | let uncarry(a:int, b:int, c:int) = (a+b+c, a\*b\*c) |
|  | let carry (a:int) (b:int) (c:int) = (a+b+c, a\*b\*c) |
|  | let rec recsum a b = if a<b then b + recsum (a) (b-1) else a |
|  | let rec recsum\_tr(a:int, b:int, acc:int):int = |
|  | if a=b then acc+a |
|  | else recsum\_tr(a, b-1, acc+b) |
|  |  |
|  | ///Обертка для сокрытия хвостовой рекурсии |
|  | let rec recsum2 a b = recsum\_tr(a,b,0) |
|  | let lambda = fun (a:int, b:int, c:int) -> a+b\*2+c |
|  | let Func (a:int, b:int, c:int, func1: int\*int\*int->int) = lambda(a,b,c) |
|  |  |
|  | let lambda2 = fun (a:int) (b:int) (c:int)-> a+b\*2+c |
|  | let Func2 (a:int, b:int, c:int, func1: int->int->int->int)= lambda2 a b c |
|  |  |
|  | let menu() = |
|  | let mutable work = true |
|  | let mutable x = 0 |
|  | while work do |
|  | printfn "=================MENU================= |
|  | 1 - два варианта функции - парамеры в виде кортежа/ параметры в каррированном виде |
|  | 2 - рекурсивное вычисление суммы целых чисел в заданном диапазоне |
|  | 3 - вычисление суммы целых чисел в заданном диапазоне в виде хвостовой рекурсии |
|  | 4 - конечный автомат из 3 состояний в виде взаимно-рекурсивных функций |
|  | 5 - Func |
|  | 6 - exit\n======================================" |
|  | x <- int32(System.Console.ReadLine()); |
|  | if x=1 then |
|  | let q1=uncarry(1,2,3) |
|  | let q2=carry 1 2 3 |
|  | printfn "uncarry(1,2,3)= %A" q1 |
|  | printfn "carry 1 2 3= %A" q2 |
|  | else if x=2 then |
|  | let q3 = recsum 1 4 |
|  | printfn "recsum 1 4= %i" q3 |
|  | else if x=3 then |
|  | let q4 = recsum2 1 4 |
|  | printfn "recsum2 1 4= %i" q4 |
|  | else if x=4 then State1(1) |
|  | else if x=5 then |
|  | let q5=Func(1, 2, 3, fun(a,b,c)->a+b+c) |
|  | printfn "uncarry Func= %i" q5 |
|  | let q6 = Func2(1, 2, 3, fun a b c -> a+b+c) |
|  | printfn "carry Func2= %i" q6 |
|  | else if x=6 then |
|  | work<-false |
|  | printfn "Exit\n" |
|  | else printfn "Повторите ввод\n" |
|  |  |
|  |  |
|  | menu() |

