F#. Примеры

Кузаев А.Ф.

Пример 1. Функциональные типы и описание функций

```
// Объявление импорта указывает модуль или пространство имен, на элементы которого
    можно ссылаться без использования полного имени.
open System
[<EntryPoint>]
// Функция нахождения суммы трех целых чисел
let main argv =
    let sum (a,b,c) = a + b + c

    printfn "%A" (sum (1,3,4),sum(5,4,7))
    printfn "Нажмите любую клавишу для продолжения"
    Console.ReadKey()|> ignore
    0
```

Пример 2. Условный оператор

```
// В какой координатной четверти расположена точка?
open System
[<EntryPoint>]
let main argv =
    let chetv (x,y) =
      if (x>0)&&(y>0) then 1
      else if (x<0)&&(y>0) then 2
            else if (x<0)&&(y<0) then 3
                  else 4
    let x =
      Console.Write "Введите x: "
      Convert.ToInt32(Console.ReadLine())
    let y =
      Console.Write "Введите y: "
      Convert.ToInt32(Console.ReadLine())
    printf "%A" (chetv (x,y))
    printfn " четверть"
    printfn "Нажмите любую клавишу для продолжения«
    Console.ReadKey()|> ignore
    0
```

Пример 3. Рекурсия

```
// Вывод всех нечетных чисел от 1 до n
open System
[<EntryPoint>]
let main argv =

    let rec print n =
        if n=1 then printf "%d%s" 1 " "
        else if (n%2=1) then  print (n-1)
                              printf "%d%s" n " "
         else print (n-1)

    printfn "%A" (print 15)
    printfn "Нажмите любую клавишу для продолжения“
    Console.ReadKey()|> ignore
    0
```

Пример 4. Цикл FOR

```
// Вывод всех нечетных чисел от 1 до n
open System
[<EntryPoint>]
let main argv =
    let rec print n =
        // for x=1 to n do
        // if (x%2=1) then printf "%d%s" x " "
        // else printf "%s" ""
        for x in 1..n do
        if (x%2=1) then printf "%d%s" x " "
        else printf "%s" ""
    printfn "%A" (print 15)
    printfn "Нажмите любую клавишу для продолжения"
    Console.ReadKey()|> ignore
    0
```

Списки

Список можно определить путем прямого перечисления элементов, разделенных точкой с запятой и заключенных в квадратные скобки, как показано в следующей строке кода.

```
let list123 = [ 1; 2; 3 ]
```

Вместо точки с запятой для разделения элементов можно также использовать разрыв строки. Такой синтаксис позволяет получить более удобный для чтения код, если список содержит длинные выражения инициализации или к каждому элементу необходимо написать комментарий.

```
let list123 = [
1
2
3 ]
```

Списки

Пустой список определяется парой квадратных скобок, между которыми ничего не указано.

```
let listEmpty = []
```

Также список можно создать с помощью выражения последовательности. Например, в следующем коде создается список квадратов целочисленных значений от 1 до 10.

```
let listOfSquares = [ for i in 1 .. 10 -> i*i ]
```

Генераторы списков

В более сложных случаях можно использовать генераторы списков. Генератор списка — это фрагмент кода, заключённый в квадратные скобки, используемый для создания всех элементов списка.

Элементы списка добавляются с помощью ключевого слова **yield** [ji:ld] .

С помощью такого выражения, например, можно получить список чётных чисел:

```
> let evenlst = [
    for i in 1..10 do
      if i % 2 = 0 then
          yield i
      ]

val evenlst : int list = [2; 4; 6; 8; 10]
```

Пример 5. Списки

```
// Формирование списка из делителей числа N
open System
[<EntryPoint>]
let main argv =
    let listdelit N =
        [
          for i in 1 .. N  do
                if N % i = 0 then yield i
        ]
    printfn "%A" (listdelit 100)
    printfn "Нажмите любую клавишу для продолжения"
    Console.ReadKey()|> ignore
    0
```

Пример 6. Списки

```
// Формирование списка из случайных чисел
open System
[<EntryPoint>]
let main argv =
    let n = Console.Write "Введите n: "
            Convert.ToInt32(Console.ReadLine())
    let res x =
      [
       let r = new Random()
       for i in 1 .. x do
       yield r.Next(-100, 100)
      ]
    let Res = res n
    printfn "%A" Res
    printfn "Нажмите любую клавишу для продолжения«
    Console.ReadKey()|> ignore
    0
```

Пример 7. Обработка списков

```
// Определение длины списка  (количество элементов в списке)
open System
[<EntryPoint>]
let main argv =
    let list = [1;2;3;4;5;6;7]
    let rec kol li =
     match li with
     [ ] -> 0
     | head::tail -> 1+kol tail
    printfn "%A\n%A" list (kol list)
    printfn "Нажмите любую клавишу для продолжения "
    Console.ReadKey()|> ignore
    0
```

Пример 8. Обработка списков

```
// Определение количества нулей в списке
open System
[<EntryPoint>]
let main argv =
    let x = [0;0;0;0;0;0;0]
    let rec nul x =
        match x with
            [ ] -> 0
            |head::tail -> if head = 0 then nul(tail)+1
                           else nul(tail)
    printfn "%A\n%A" x (nul x)
    printfn "Нажмите любую клавишу для продолжения"
    Console.ReadKey()|> ignore
    0
```

Пример 9. Обработка списков

// Подсчитать количество чисел в списке, заканчивающихся на K (с использованием стандартной функции filter). Использование рекурсии не допускается.

```
open System
[<EntryPoint>]
let main argv =
    let k=1
    let list = [50; 105; 10; 31; 7; 11; 5; 24]
    let listres = List.filter (fun x -> x%10=k) list
    printfn "%A" (listres.Length)
    printfn "Нажмите любую клавишу для
     продолжения"
    Console.ReadKey()|> ignore
    0
```

Свертка

Операция свертки применяется тогда, когда необходимо получить по списку некоторый интегральный показатель – минимальный или максимальный элемент, сумму или произведение элементов и т. д.

Свертка является заменой циклической обработки списка, в которой используется некоторый аккумулятор, на каждом шаге обновляющийся в результате обработки очередного элемента.

Поскольку в функциональном программировании нет переменных, то традиционное решение с аккумулятором невозможно. Вместо этого используется в явном виде передаваемое через цепочку функций значение – состояние. Функция свертки будет принимать на вход это значение и очередной элемент списка, а возвращать – новое состояние.

Пример 10. Свертка

```
// сколько раз встречается минимальный элемент в списке
open System
[<EntryPoint>]
let main argv =
    let list=[1; 20; 3; 0; 0; 3; 7; 5; 32; 0; 76; 4; 0; 32; 0]
    let nul (min,k) i =
      if i=min then (min,k+1)
      else if i<min then (i,1)
           else (min,k)
    let x = List.fold nul (List.head list,0) list
    printfn "%A" list
    printfn "%A" x
    printfn "%A" (snd x)
    printfn "Нажмите любую клавишу для продолжения"
    Console.ReadKey()|> ignore
    0
```

Пример 11.

// Количество простых чисел в списке натуральных чисел

```
open System
[<EntryPoint>]
let main argv =
    let listnat = [1; 2; 4; 6; 7; 10; 11; 13]
    let list n =
        [for i in 2..n-1 do
            if n%i=0 then
                yield i
         ]
    let listpr = List.filter (fun x -> (list x).Length=0) listnat
    printfn "%A" (listpr.Length)
    printfn "Нажмите любую клавишу для продолжения"
    Console.ReadKey()|> ignore
    0
```