

Министерство науки и высшего образования Российской Федерации  
Севастопольский государственный университет  
Кафедра ИС

Отчет  
по лабораторной работе №2  
«Сравнение итерационного и рекурсивного методов решения задач»  
по дисциплине  
«МЕТОДЫ И СРЕДСТВА ИСКУССТВЕННОГО ИНТЕЛЛЕКТА»

Выполнил студент группы ИС/б-17-2-о  
Горбенко К. Н.  
Проверил  
Сметанина Т.И.

Севастополь  
2020

# 1 ЦЕЛЬ РАБОТЫ

Исследование способов организации циклических вычислений в языке Лисп с помощью итерационного и рекурсивного методов, сравнение указанных методов по вычислительной эффективности и выразительности, получение практических навыков работы со списочными структурами.

## 2 ЗАДАНИЕ НА РАБОТУ

Задача: описать функцию, которая добавляет число в упорядоченный по возрастанию список без нарушения порядка рекурсивно и с помощью механизмов организации итерационных циклов. Сравнить оба способа по вычислительной эффективности. Рекурсивное решение должно удовлетворять требованиям функционального программирования.

## 3 ХОД РАБОТЫ

Составим функцию для решения задачи рекурсивно.

```

1 (defun addToSortedListRecursive (item list)
2   (if (null list)
3       (list item)
4       (if (> item (first list))
5           (append (list (first list)) (addToSortedListRecursive item (cdr list)))
6           (append (list item) list)
7       )
8   )
9 )

```

В данном листинге использованы следующие функции:

- `null` – для проверки массива на NIL.
- `append` – объединяет несколько списков в один.

Для сравнения приведу функцию, выполняющую те же функции на языке F# (также функциональном).

```

1 let rec addToSortedListRecursive item list =
2   match list with
3   | [] -> [ item ]
4   | head :: tail -> if item > head
5                       then head :: (addToSortedListRecursive item tail)
6                       else item :: list

```

На мой взгляд, функциональный подход в языке LISP менее выразителен, чем в современных функциональных языках программирования.

Функция для итерационного вычисления:

```

1 (defun getRange (list start end)
2   (loop for i from start to end
3     collect (nth i list)
4   )
5 )
6
7 (defun range (min max step)
8   (loop for n from min below max by step
9     collect n
10  )
11 )
12
13 (defun addToSortedListIterative (item list)
14   (cond
15     ((null list) (list item))
16     ((> item (car (last list))) (append list (list item)))
17     ((< item (first list)) (append (list item) list))
18     (T (loop for i in (reverse (range 0 (length list) 1))
19       when (> item (nth i list))
20         return (append (getRange list 0 i) (list item) (getRange list (+ i 1)
21           (- (length list) 1)))
22   ))
23 )

```

В функции `addToSortedListIterative` перед проведением итерации происходит проверка пограничных случаев (список пуст, вставляемый элемент – первый или последний в результирующем списке). Затем при итерации последовательно находим позицию, в которую нужно вставить элемент и формируем новый список.

Использованные функции:

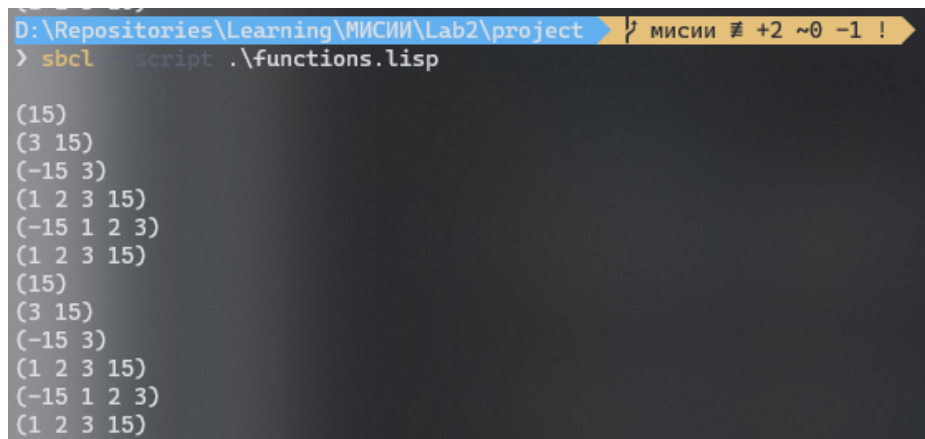
- `loop` – макро для организации циклов.
- `reverse` – оборачивает список.
- `return` – возвращает результат из цикла (функции `loop`).
- `length` – возвращает размер списка.

Составим тестовые случаи:

```

1 (print (addToSortedListRecursive 15 nil))

```



```

D:\Repositories\Learning\МИСИИ\Lab2\project > sbcl script .\functions.lisp

(15)
(3 15)
(-15 3)
(1 2 3 15)
(-15 1 2 3)
(1 2 3 15)
(15)
(3 15)
(-15 3)
(1 2 3 15)
(-15 1 2 3)
(1 2 3 15)

```

Рисунок 1 – Результат выполнения программы

```

2 (print (addToSortedListRecursive 15 (list 3)))
3 (print (addToSortedListRecursive -15 (list 3)))
4 (print (addToSortedListRecursive 15 (list 1 2 3)))
5 (print (addToSortedListRecursive -15 (list 1 2 3)))
6 (print (addToSortedListRecursive 3 (list 1 2 15)))
7
8 (print (addToSortedListIterative 15 nil))
9 (print (addToSortedListIterative 15 (list 3)))
10 (print (addToSortedListIterative -15 (list 3)))
11 (print (addToSortedListIterative 15 (list 1 2 3)))
12 (print (addToSortedListIterative -15 (list 1 2 3)))
13 (print (addToSortedListIterative 3 (list 1 2 15)))

```

Результат выполнения программы изображен на рисунке 1:

По рисунку 1 видно, что две функции работают одинаково.

## ВЫВОДЫ

В ходе лабораторной работы были исследованы рекурсивный и итерационный подходы языка программирования LISP. Для решения задачи были разработаны две функции с помощью разных подходов. Рекурсивная функция значительно компактнее и проще, чем итерационная. Кроме того, на мой взгляд, она более выразительна.

Тем не менее, рекурсивная функция менее выразительна, чем такая же на современном языке.