## МИНОБРНАУКИ РОССИИ

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра МОЭВМ**

## ОТЧЕТ **по лабораторной работе** №2 **по дисциплине «Алгоритмы и структуры данных»** ТЕМА: Иерархические списки

|  |  |  |
| --- | --- | --- |
| Студент гр. 7303 |  | Юсковец А.В. |
| Преподаватель |  | Балтрашевич Т.А. |

Санкт-Петербург  
2018

Оглавление

[Цель работы 3](#__RefHeading___Toc1709_692322312)

[Задание 3](#__RefHeading___Toc8601_2844259154)

[Ход работы 3](#__RefHeading___Toc1711_692322312)

[Вывод 3](#__RefHeading___Toc1731_692322312)

[Приложение А. Исходный код 4](#__RefHeading___Toc1095_472082746)

# Цель работы

Ознакомиться с реализаций иерархических списков с использованием базовых функций рекурсивной обработки списков.

# Задание

Сформировать линейный список атомов исходного иерархического списка путем устранения всех внутренних скобок в его сокращенной скобочной записи.

# Ход работы

Структура иерархического списка и реализация функции flatten приведена в приложении А.

В каждом lisp, хранится булевский флаг, который определяет лежит ли в данном узле атом или очередной список. Каждый список имеет два поля: указатель на голову и указатель на хвост списка.

Функция flatten принимает список, который нужно привести к линейному.  
- если список пустой, то возвращается нулевой указатель.   
- если список — атом, то возвращается список сконструированный из этого атома.  
-иначе   
- если в голове списка лежит атом, то возвращается список сконструированный из этого атома и линейного списка, который получится из хвоста.  
 - иначе возвращается список, сконструированный из линейного списка, в который преобразуется голова и линейного списка, в который преобразуется хвост.  
\* Функция isAtom — проверяет лежит ли в данном узле атом или очередной список.  
\* Функция cons — конструирует новый список на основе двух других списков.

# Вывод

Была написана программа работающая с иерархическим списком. При написании программы была использована рекурсивная обработка списка. При компиляции и выполнении программы не возникает ошибок и предупреждений.

# Приложение А. Исходный код

typedef std::string base;

struct **s\_expr**;

struct **two\_ptr** {

s\_expr \* hd = nullptr;

s\_expr \* tl = nullptr;

};

struct **s\_expr** {

bool tag;

struct {

base atom;

two\_ptr pair;

} node;

}; typedef s\_expr\* lisp;

lisp **flatten**(const lisp s) {

if (!s) return nullptr;

else if (isAtom(s))

return cons(make\_atom(getAtom(s)), nullptr);

else {

if (isAtom(head(s)))

return cons(make\_atom(getAtom(head(s))), flatten(tail(s)));

else

return concat(flatten(head(s)), flatten(tail(s)));

}

}