|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **ПРОГРАМНОЕ ОБЕСПЕЧЕНИЕ ЭВМ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ (ИУ7)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.03 ПРОГРАММНАЯ ИНЖЕНЕРИЯ**

**Отчет**

|  |  |
| --- | --- |
| **По лабораторной работе №** | 7 |

**Дисциплина:** Функциональное и логическое программирование

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ7-62Б |  |  | Н.А. Гарасев |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | Н.Б.Толпинская |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2021

**Задание 1. Проверка массива на полином.**

**(defun** pol **(**lst**)**

**(**compare lst **(**my\_revers lst**)))**

**(defun** compare **(**lst1 lst2**)**

**(cond** **((null** lst1**)** T**)**

**((eq** **(car** lst1**)** **(car** lst2**))** **(**compare **(cdr** lst1**)** **(cdr** lst2**)))))**

**Revers**

**(defun** my\_revers **(**lst**)** **(**my\_rev lst Nil**))**

**(defun** my\_rev **(**lst res**)**

**(cond** **((null** lst**)** res**)**

**(**T **(**my\_rev **(cdr** lst**)** **(cons** **(car** lst**)** res**)))))**

\* (pol `(1 2 3 2 1))

T

\* (pol `(1 2 3 2 2))

NIL

**Задание 2. Предикат сравнения множеств**

**(defun** my\_len **(**lst**)**

**(cond** **((null** lst**)** 0**)**

**(**t **(+** 1 **(**my\_len**(cdr** lst**))))))**

**(defun** check\_len **(**set1 set2**)**

**(equal** **(**my\_len set1**)** **(**my\_len set2**)))**

**(defun** find\_set **(**el set2**)**

**(cond** **((null** set2**)** Nil**)**

**((eq** el **(car** set2**))** T**)**

**(**T **(**find\_set el **(cdr** set2**)))))**

**(defun** set-equal\_rec **(**set1 set2**)**

**(cond** **((null** set1**)** T**)**

**((**find\_set **(car** set1**)** set2**)** **(**set-equal\_rec **(cdr** set1**)** set2**))))**

**(defun** set-equal **(**set1 set2**)**

**(and** **(**check\_len set1 set2**)** **(**set-equal\_rec set1 set2**)))**

\* (set-equal `(1 2 3 4) `(5 4 2 3 1))

NIL

\* (set-equal `(1 2 3 4) `(1 4 2 3))

T

**Задание 3. Поиск по таблице**

**(defun** find\_key **(**table key**)**

**(cond** **((null** table**)** Nil**)**

**((eq** **(caar** table**)** key**)** **(cdar** table**))**

**(**T **(**find\_key **(cdr** table**)** key**))))**

**(defun** find\_value **(**table v**)**

**(cond** **((null** table**)** Nil**)**

**((eq** **(cdar** table**)** v**)** **(caar** table**))**

**(**T **(**find\_value **(cdr** table**)** v**))))**

\* (find\_key table `russia)

MOSCOW

\* (find\_key table `rus)

NIL

\* (find\_value table `moscow)

RUSSIA

\* (find\_value table `mos)

NIL

**Задание 4. Swap-first-last**

**(defun** my\_last **(**lst**)**

**(cond** **((null** **(cdr** lst**))** **(car** lst**))**

**(**T **(**my\_last **(cdr** lst**)))))**

**(defun** first-to-last **(**lst**)**

**(**my\_revers **(cons** **(car** lst**)(cdr** **(**my\_revers **(cdr** lst**))))))**

**(defun** swap-first-last **(**lst**)**

**(cons** **(**my\_last lst**)** **(**swap-main lst**)))**

\* (swap-first-last '(1 2 3 4 5 6))

(6 2 3 4 5 1)

\* (swap-first-last '(1 2))

(2 1)

**Задание 5. Swap-two-element**

**(defun** my\_nth **(**lst n**)**

**(cond** **((null** lst**)** Nil**)**

**((=** n 0**)** **(car** lst**))**

**(**T **(**my\_nth **(cdr** lst**)** **(-** n 1**)))))**

**(defun** swap-two-element **(**lst a b**)**

**(**reduce **(lambda** **(**res tmp**)**

**(cond** **((eq** **(**my\_len res**)** a **)** **(append** res **(list** **(**my\_nth lst b**))))**

**((eq** **(**my\_len res**)** b **)** **(append** res **(list** **(**my\_nth lst a**))))**

**(**T **(append** res **(list** tmp**)))))**

lst **:initial-value** Nil**))**

\* (swap-two-element '(1 2 3 4 5 6 7 8 9) 3 6)

(1 2 3 7 5 6 4 8 9)

\* (swap-two-element '(1 2 3 4 5 6 7 8 9) 3 8)

(1 2 3 9 5 6 7 8 4)

\* (swap-two-element '(1 2 3 4 5 6 7 8 9) 2 8)

(1 2 9 4 5 6 7 8 3)

**Задание 6. Круговая перестановка**

; swap-to-left

**(defun** swap-to-left **(**lst**)**

**(append** **(cdr** lst**)** **(list** **(car** lst**))))**

; swap-to-right

**(defun** swap-to-right **(**lst**)**

**(append** **(list** **(**my\_last lst**))** **(**my\_revers **(cdr** **(**my\_revers lst**)))))**

\* (swap-to-left `(1 2 3 4 5 6))

(2 3 4 5 6 1)

\* (swap-to-right `(1 2 3 4 5 6))

(6 1 2 3 4 5)

**Вопросы.**

**Вопрос 1. Способы определения функций.**

Существуют безымянные функции (lambda функции), они имеют синтаксис: (LAMBDA (список параметров) (тело функции)). Для создания собственных функций используется синтаксис: (DEFUN имя функции (список параметров) (тело функции)).

**Вопрос 2. Варианты и методы модификации элементов списка.**

Функции делятся на два типа: структуроразрушающие и не разрушающие структуру. Первый тип функций используют структуры, которые им переданы. Например, функция nconc – объединяет структуры, заставляя последний элемент первого аргумента ссылаться на второй аргумент и т.д.