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 «Программное обеспечение ЭВМ и информационные технологии»

ОТЧЕТ

по Лабораторной работе №3

по курсу «Функциональное и логическое программирование»

на тему: «Работа интерпретатора Lisp»

Студент ИУ7-63Б _____ Миронов Г. А.
(Группа) (Подпись, дата) (И. О. Фамилия)

Преподаватель _____ Толпинская Н. Б.
(Подпись, дата) (И. О. Фамилия)

2022 г.

1 Практическая часть

1.1 Написать функцию, которая принимает целое число и возвращает первое четное число, не меньшее аргумента

Листинг 1.1 – Функция, возвращающая первое четное число, не меньшее аргумента

```
1 (defun first-even-ge (arg) (if (evenp arg) arg (+ arg 1)))
```

Листинг 1.2 – Lambda-функция, возвращающая первое четное число, не меньшее аргумента

```
1 (lambda (arg) (if (evenp arg) arg (+ arg 1)))
```

1.2 Написать функцию, которая принимает число и возвращает число того же знака, но с модулем на 1 больше модуля аргумента

Листинг 1.3 – Функция, увеличивающая аргумент на 1 по модулю

```
1 (defun module-plus (arg) (+ arg (if (< arg 0) -1 1)))
```

Листинг 1.4 – Lmabda-функция, увеличивающая аргумент на 1 по модулю

```
1 (lambda (arg) (+ arg (if (< arg 0) -1 1)))
```

1.3 Написать функцию, которая принимает два числа и возвращает список из этих чисел, расположенный по возрастанию

Листинг 1.5 – Функция, упорядочивающая аргументы по возрастанию

```
1 (defun growing-lst (a b)
2   (if (< a b) (list a b) (list b a)))
```

Листинг 1.6 – Lambda-функция, упорядочивающая аргументы по возрастанию

```
1 (lambda (a b) (if (< a b) (list a b) (list b a)))
```

1.4 Написать функцию, которая принимает три числа и возвращает T только тогда, когда первое число расположено между вторым и третьим

Листинг 1.7 – Функция проверки принадлежности интервалу

```
1 (defun pred (a b c) (and (> a b) (< a c)))
```

Листинг 1.8 – Lamba-функция проверки принадлежности интервалу

```
1 (lambda (a b c) (and (> a b) (< a c)))
```

1.5 Каков результат вычисления следующих выражений

В Таблице 1.1 приведены результаты вычисления выражений, а так же варианты устранения возникших ошибок.

Таблица 1.1 – Результаты вычисления выражений

Выражение	**Результат**
(and 'fee 'fie 'foe)	foe
(or 'fee 'fie 'foe)	fee
(or nil 'fie 'foe)	fie
(and nil 'fie 'foe)	nil
(and (equal 'abc 'abc) 'yes)	yes
(or (equal 'abc 'abc) 'yes)	T

1.6 Написать предикат, который принимает два числа-аргумента и возвращает T, если первое число не меньше второго

Листинг 1.9 – Предикат сравнения чисел

```
(defun predicate-2 (a b) (>= a b))
```

1.7 Какой из следующих двух вариантов предиката ошибочен и почему

Листинг 1.10 – Первая реализация предиката

```
(defun pred1 (x) (and (numberp x) (plusp x)))
```

Листинг 1.11 – Вторая реализация предиката

```
(defun pred2 (x) (and (plusp x) (numberp x)))
```

Ошибочен второй вариант, потому что функция `plusp` принимает на вход один аргумент типа `number` и проверять, является ли аргумент числом, после выполнения функции `plusp` не имеет смысла, причем аргументы, не являющиеся числами, будут вызывать ошибку, в то время как 1 вариант будет работать с любым аргументом и возвращать `T` для положительных чисел.

1.8 Решить задачу 4, используя для ее решения конструкции IF, COND, AND/OR

Используя `if`:

Листинг 1.12 – Реализация с использованием `if`

```
(defun pred (a b c) (if (> a b) (< a c) Nil))
```

Используя `cond`:

Листинг 1.13 – Реализация с использованием `cond`

```
(defun pred (a b c)
    (cond ((> a b) (cond ((< a c) T)
                         (T Nil)))
          (T Nil))
)
```

Используя `and/or`:

Листинг 1.14 – Реализация с использованием and/or

```
1 (defun pred (a b c) (and (> a b) (< a c)))
```

1.9 Переписать функцию how-alike, приведенную в лекции и использующую COND, используя только конструкции IF, AND/OR

Используя cond:

Листинг 1.15 – Реализация с использованием cond

```
1 (defun how-alike (x y)
2   (cond ((or(= x y) (equal x y)) 'the_same)
3         ((and (oddp x) (oddp y)) 'both_odd)
4         ((and (evenp x) (evenp y)) 'both_even)
5         (T 'diff)))
```

Используя if:

Листинг 1.16 – Реализация с использованием if

```
1 (defun how-alike-if (x y)
2   (if (or (= x y) (equal x y)) 'the_same
3     (if (and (oddp x) (oddp y)) 'both_odd
4       (if (and (evenp x) (evenp y)) 'both_even
5         'diff))))
```

Используя and/or:

Листинг 1.17 – Реализация с использованием if

```
1 (defun how-alike-and-or (x y)
2   (or (and (or (= x y) (equal x y)) 'the_same)
3         (and (and (oddp x) (oddp y)) 'both_odd)
4         (and (and (evenp x) (evenp y)) 'both_even)
5         'diff))
```

2 Контрольный вопросы

2.1 Базис языка

Базис состоит из:

1. структуры, атомы;
2. встроенные (примитивные) функции (`atom`, `eq`, `cons`, `car`, `cdr`);
3. специальные функции и функционалы, управляющие обработкой структур, представляющих вычислимые выражения (`quote`, `cond`, `lambda`, `label`, `eval`).

2.2 Классификация функций

Функции в `Lisp` классифицируют следующим образом:

- чистые математические функции;
- рекурсивные функции;
- специальные функции — формы (сегодня 2 аргумента, завтра - 5);
- псевдофункции (создают эффект на внешнем устройстве);
- функции с вариативными значениями, из которых выбирается 1;
- функции высших порядков — функционал: используется для синтаксического управления программ (абстракция языка).

По назначению функции разделяются следующим образом:

- конструкторы — создают значение (`cons`, например);
- селекторы — получают доступ по адресу (`car`, `cdr`);
- предикаты — возвращают `Nil`, `T`.
- функции сравнения – такие как: `eq`, `eql`, `equal`, `equalp`.

2.3 Способы создания функций

Функции в Lisp можно задавать следующими способами:

Lambda-выражение

Синтаксис:

(lambda <λ-список> форма)

Пример:

Листинг 2.1 – Функция определенная Lambda-выражением

```
1 (lambda (a b) (sqrt (+ (* a a) (* b b))))
```

Именованая функция

Синтаксис:

(defun <имя функции> <λ-выражение>)

Пример:

Листинг 2.2 – Функция определенная Lambda-выражением

```
1 (defun hyp (a b) (sqrt (+ (* a a) (* b b))))
```

2.4 Работа функций and, or, if, cond

2.4.1 Функция and

Синтаксис:

Листинг 2.3 – функция and

```
1 (and expression-1 expression-2 ... expression-n)
```

Функция возвращает первое expression, результат вычисления которого = Nil. Если все не Nil, то возвращается результат вычисления последнего выражения.

Примеры:

Листинг 2.4 – пример использования `and`

```
(and 1 Nil 2)
```

Результат: `Nil`

Листинг 2.5 – пример использования `and`

```
(and 1 2 3)
```

Результат: 3

2.4.2 Функция `or`

Синтаксис:

Листинг 2.6 – функция `or`

```
(or expression-1 expression-2 ... expression-n)
```

Функция возвращает первое expression, результат вычисления которого не `Nil`. Если все `Nil`, то возвращается `Nil`.

Примеры:

Листинг 2.7 – пример использования `or`

```
(or Nil Nil 2)
```

Результат: 2

Листинг 2.8 – пример использования `or`

```
(or 1 2 3)
```

Результат: 1

2.4.3 Функция `if`

Синтаксис:

Листинг 2.9 – функция `if`

```
(if condition t-expression f-expression)
```

Если вычисленный предикат не `Nil`, то выполняется `t-expression`, иначе - `f-expression`.

Примеры:

Листинг 2.10 – пример использования `if`

```
(if Nil 2 3)
```

Результат: 3

Листинг 2.11 – пример использования `if`

```
(if 0 2 3)
```

Результат: 2

2.4.4 Функция `cond`

Синтаксис:

Листинг 2.12 – Функция `cond`

```
(cond
  (condition-1 expression-1)
  (condition-2 expression-2)
  ...
  (condition-n expression-n))
```

По порядку вычисляются и проверяются на равенство с `Nil` предикаты. Для первого предиката, который не равен `Nil`, вычисляется находящееся с ним в списке выражение и возвращается его значение. Если все предкаты вернут `Nil`, то и `cond` вернет `Nil`.

Примеры:

Листинг 2.13 – Пример использования `cond`

```
(cond (Nil 1) (2 3))
```

Результат: 3

Листинг 2.14 – Пример использования `cond`

```
(cond (Nil 1) (Nil 2))
```

Результат: `Nil`