**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

**ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ 1**

по дисциплине «Базы знаний и поддержка принятия решений»

на тему: «Язык программирования ЛИСП»

Выполнил: студент гр. ИТП-31

Коркуц С. И.

Принял: профессор

Мурашко И. А.

Гомель 2020

**Цель работы:**

Знакомство с языком *Lisp*. Изучение представления данных в языке Лисп в

виде атомов, списков, консов и символьных выражений. Изучение основных числовых функций языка.

**Задание:**

Разработать программу вычисления факториала на процедурном (Pascal, C,

можно C#), объектно-ориентированном (C++ или Java) и функциональном языке

программирования (Lisp). Найти максимальное значение аргумента, от которого

можно вычислить факториал без потери точности (в целых числах).

Изучить базовые функции языка для работы со списками – CAR, CDR, CONS, предикаты ATOM и EQ и условное выражение COND. Разработать программы для решения следующих задач:

1. Удалить первый и последний элемент списка.

2. В простом списке чисел заменить все отрицательные числа нулями.

3. Из простого списка чисел удалить все нулевые элементы

4. Продублировать все вхождения атома X в данный список.

5. Подсчитать число вхождений атома Х в простой список.

6. Выделить первую цифру натурального N.

7. Получить все делители натурального N.

8. Подсчитать число и сумму цифр целого N.

9. Найти все общие делители натуральных M и N.

10. Найти наибольший общий делитель чисел из заданного списка.

**Ход выполнения:**

Изначально была разработана программа для вычисления факториала числа на языках *Lisp* и *Java*. Результат выполнения программы на языке *Lisp* на рисунке 1. Результат выполнения на языке *Java* на рисунке 2.



Рисунок 1 – Результат выполнения функции факториал на языке *Lisp*

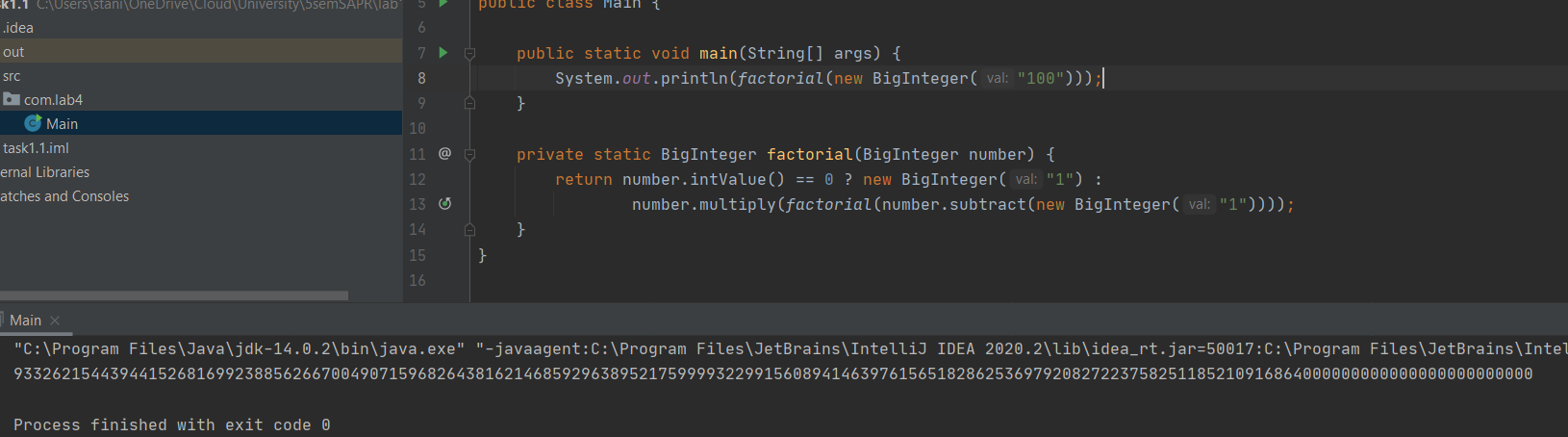


Рисунок 2 – Результат выполнения функции на языке *Java*

В результате сравнения двух программ было установлено что максимальное число, которой посчитал *Lisp* – это факториал числа 1117. *Java* вычисляет числа больше 10000.

Далее на рисунках 3 – 4 изображены результаты выполнения заданий.

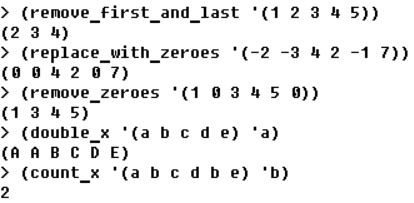


Рисунок 3 – Результат выполнения заданий 1 – 5

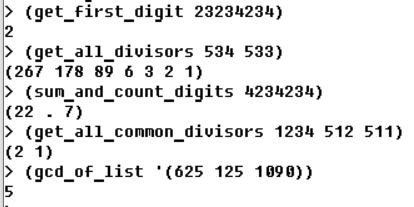


Рисунок 4 – Результат выполнения заданий 6 – 10

**Вывод:**

Были изучены базовые функции языка *Lisp* для работы со списками и числами. Были написаны функции, выполняющие действия согласно заданию.

ПРИЛОЖЕНИЕ А

Листинг функций

(defun fact1 (n)

(cond ((zerop n) 1)

(t (\* n (fact1 (- n 1))))))

(defun remove\_last (lst)

(cond

((null lst) nil)

((null (cdr lst)) nil)

(t (cons (car lst) (remove\_last (cdr lst))))

)

)

(defun remove\_first\_and\_last (lst)

(cdr (remove\_last lst))

)

(defun replace\_with\_zeroes (lst)

(cond

((null lst) nil)

((minusp (car lst)) (cons 0 (replace\_with\_zeroes (cdr lst))))

(t (cons (car lst) (replace\_with\_zeroes (cdr lst))))

)

)

(defun remove\_zeroes (lst)

(cond

((null lst) nil)

((zerop (car lst)) (remove\_zeroes (cdr lst)))

(t (cons (car lst) (remove\_zeroes (cdr lst))))

)

)

(defun double\_x (lst x)

(cond

(

(null lst)

nil)

(

(eql (car lst) x)

(cons (car lst) (cons (car lst) (double\_x (cdr lst) x)))

)

(

t

(cons (car lst) (double\_x (cdr lst) x))

)

)

)

(defun count\_x (lst x)

(cond

((null (cdr lst)) 0)

((eq (car lst) x) (+ 1 (count\_x (cdr lst) x)))

(t (+ 0 (count\_x (cdr lst) x)))

)

)

(defun get\_first\_digit (N)

(cond

((null N) nil)

((< N 10) N)

(t (get\_first\_digit (floor (/ N 10))))

)

)

(defun get\_all\_divisors (N i)

(cond

((zerop i) nil)

((zerop (rem N i)) (cons i (get\_all\_divisors N (- i 1))))

(t (get\_all\_divisors N (- i 1)))

)

)

(defun count\_digits (N)

(cond

((null N) nil)

((< N 10) 1)

(t (+ 1 (count\_digits (floor (/ N 10)))))

)

)

(defun sum\_digits (N)

(cond

((null N) nil)

((< N 10) N)

(t (+ (rem N 10) (sum\_digits (floor (/ N 10)))))

)

)

(defun sum\_and\_count\_digits (N)

(cons (sum\_digits N) (count\_digits N))

)

(defun get\_all\_common\_divisors (N M i)

(cond

((zerop i) nil)

((and (zerop (rem M i)) (zerop (rem N i))) (cons i (get\_all\_common\_divisors N M (- i 1))))

(t (get\_all\_common\_divisors N M (- i 1)))

)

)

(defun gcd\_of\_list (mylst)

(cond ((null mylst) nil)

((null (cdr mylst)) (car mylst))

((gcd (car mylst) (gcd\_of\_list (cdr mylst))))))