



*Томский межвузовский центр  
дистанционного образования*

**В. М. Зюзьков**

# **ЛОГИЧЕСКОЕ И ФУНКЦИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ**

Учебное методическое пособие

Томск - 2000

Министерство образования Российской Федерации

**ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ  
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)**

Кафедра автоматизированных систем управления (АСУ)

**В. М. Зюзьков**

## **ЛОГИЧЕСКОЕ И ФУНКЦИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ**

Учебное методическое пособие

**2000**

**Зюзьков ВМ.**

Логическое и функциональное программирование: Учебное пособие. - Томск: Томский межвузовский центр дистанционного образования, 2000. - 72 с.

Учебное методическое пособие содержит учебные программы, описания инсталляций интерпретаторов с Пролога и Лиспа, требования к выполнению контрольных заданий и курсовой работы, варианты заданий, темы курсовых работ, экзаменационные вопросы. В качестве источников для получения теоретических знаний студенту необходимо и достаточно использовать учебные пособия по логическому и функциональному программированию, автор В. М. Зюзьков.

## СОДЕРЖАНИЕ

1. Учебные программы .....	4
1.1. Программа по дисциплине "Функциональное программирование" ...	4
1.2. Программа по дисциплине "Логическое программирование" .....	5
2. Функциональное программирование .....	6
2.1. Контроль обучения .....	6
2.2. Установка XLisp'a .....	6
2.3. Первое контрольное задание .....	7
2.4. Второе контрольное задание .....	9
3. Логическое программирование .....	14
3.1. Контроль обучения .....	14
3.2. Установка SWI-Prolog'a .....	14
3.3. Первое контрольное задание .....	15
3.4. Второе контрольное задание .....	19
3.5. Третье контрольное задание .....	23
4. Курсовые работы .....	28
5. Как выполнять курсовую работу и оформлять пояснительную записку	36
6. Экзаменационные вопросы .....	40
7. Рекомендуемая литература .....	69
Приложение 1. Форма титульного листа к курсовой работе .....	70
Приложение 2. Форма задания для курсовой работы .....	71
Приложение 3. Пример оформления содержания .....	72
Приложения 4. Примеры библиографических описаний источников, помещаемых в список литературы .....	72

## 1. УЧЕБНЫЕ ПРОГРАММЫ

### 1.1. Программа по дисциплине "Функциональное программирование"

Введение в функции. Чистые функции. Функциональность.

Введение в функциональное программирование. О языке Лисп. Примеры на Лиспе. Символьная обработка. Лисп опередил свое время. Одинаковая форма данных и программы. Автоматическое и динамическое управление памятью. Лисп - безтиповой язык программирования. Заблуждения и предрассудки.

Основы языка Лисп. Символы и списки. Символы используются для представления других объектов. Символы в языке Коммон Лисп. Числа являются константами. Логические значения Т и NIL. Константы и переменные. Атомы = Символы + Числа. Построение списков из атомов и подсписков. Пустой список = NIL. Список как средство представления знаний. Значение способа записи. Различные интерпретации списка. Понятие функции. Единообразная префиксная нотация. Диалог с интерпретатором Лиспа. Иерархия вызовов. QUOTE блокирует вычисление выражения. Основные функции обработки списков. First, rest, cons. Предикаты atom, =, eq, eql, equal, equalp, null. Функции second, third, (nth n список), last, list. Имя и значение символа. Значением константы является сама константа. Символ может обозначать произвольное выражение. Set вычисляет имя и связывает его. Setq связывает имя, не вычисляя его. Побочный эффект псевдофункции.

Определение функций. Defun, if, let. Использование квалифицированного выражения. Накапливающие параметры. Локальные функции. Программа: сортировка списка с помощью дерева.

Математические основы. Лямбда-исчисление. Введение в синтаксис. Вычисление  $\lambda$ -выражений. Порядок редукций и нормальные формы.  $\beta$ -редукция и проблема конфликта имен. Рекурсивные выражения. Чистое  $\lambda$ -исчисление. Ламбда-выражения в Лиспе.  $\lambda$ -вызов.

Внутреннее представление списков. Лисповская память состоит из ссылочных ячеек. Значение представляется указателем. First и rest выбирают поля указателя. Cons создает ячейку и возвращает на нее указатель. У списков могут быть общие части. Логическое и физическое равенство не одно и то же. Примеры.

Рекурсия. Простая рекурсия. Простая рекурсия соответствует циклу. Member проверяет, принадлежит ли элемент списку. Использование трассировки. Append объединяет два списка. Remove удаляет элемент из списка. Reverse обращает список. Другие формы рекурсии. Параллельное ветвление рекурсии. Взаимная рекурсия. Рекурсия более высокого порядка. Функции более высокого порядка. Функционал имеет функциональный аргумент.

Функциональное значение функции. Способы композиции функций. Функции более высокого порядка.

Применяющие функционалы. Apply применяет функцию к списку аргументов. Funcall вызывает функцию с аргументами. Отображающие функционалы. Отображающие функции повторяют применение функций. Композиция функционалов. Замыкания. Function блокирует вычисление функции. Замыкание - это функция и контекст ее определения. Связи свободных переменных замыкаются. Замыкание позволяет осуществлять частичное вычисление. Абстрактный подход. Обобщение функций, имеющих одинаковый вид. Параметризованное определение функций. Автоапликация и авторепликация.

## **1.2. Программа по дисциплине "Логическое программирование"**

Логическое программирование. Логический вывод. Метод резолюций. Унификация. Применение метода резолюций для ответа на вопросы. Введение в Пролог. Особенности языка Пролог. Пример программы: родственные отношения. Фразы Хорна как способ представления знаний. Алгоритм работы интерпретатора Пролога.

Порядок предложений и целей. Процедурная и декларативная семантика Пролога. Синтаксис языка Пролог. Арифметические выражения, арифметические функции, арифметические предикаты. Составные термы (структуры), пример программы "Упрощение цепей"

Списки. Предикаты со списками member, append, select. Примеры программ "Сортировка с помощью дерева", "Дифференцирование".

Ограничение перебора. Примеры, использующие отсечение. Отрицание как неудача. Трудности с отсечением и отрицанием. Программирование повторяющихся операций.

Метапрограммирование. Предположение об открытости мира. Внелогические предикаты: доступ к программам и обработка программ. Ввод и вывод. Работа с базой данных "Достопримечательности" (программа, которая учится у пользователя). Программирование второго порядка. Запоминающие функции. Операторная запись.

Примеры программ.

## **2. ФУНКЦИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ**

### **2.1. Контроль обучения**

В процессе дистантного обучения дисциплине "Функциональное программирование" студент должен выполнить два контрольных задания и обучение заканчивается выполнением компьютерной экзаменационной работы. Контрольные задания представлены ниже и состоят каждое из трех задач, требующих составить программы на Лиспе. Составленные и отлаженные программы (обязательно с комментариями) студент по мере освоения функционального программирования периодически пересылает по электронной почте диспетчеру центра дистантного обучения, который в свою очередь пересылает их лектору. Лектор проверяет программы и при правильном выполнении программы студент получает подтверждение о том, что они зачтены. Если программа составлена неправильно, студент получает от лектора текстовый файл, в котором содержится описание ошибок программы.

### **2.2. Инсталляция XLisp'a**

Интерпретатор с языка XLisp представлен в виде упакованного файла XLisp.arj. XLisp работает в среде Windows (3.11 или 95); для его установки на компьютере достаточно распаковать файл XLisp.arj вместе со всеми хранящимися в этом файле каталогами. Запустите файл XLisp.exe из каталога XLisp\bin и вы войдете в интерпретатор. Лисп выдает приглашение ">" и ждет вашего ввода.

Программы вы можете набирать в окне интерпретатора, но это неудобно. Лучше воспользоваться каким-либо внешним текстовым редактором (XLisp не имеет собственного редактора). Например, вы можете пользоваться стандартной программой Notepad ("блокнот"). Набранные программы сохраняйте в виде текстовых файлов с расширением lsp (но можно и расширение txt) в рабочем каталоге. По умолчанию рабочим каталогом будет XLisp\bin, но вы можете это изменить: для этого создайте ярлык для XLisp.exe и измените рабочий каталог в свойствах этого ярлыка. Загрузить программу, находясь в среде интерпретатора, вы можете, если выберете пункт меню File | Load Lisp source... . Отметим следующую особенность работы в Лиспе с блокнотом. После набора текста программы обязательно перейдите курсором на новую строку, иначе XLisp при загрузке файла с программой выдаст ошибку "неожиданный конец файла".

### 2.3. Первое контрольное задание

Задание состоит из трех задач, в которых требуется составить программы на Лиспе. В первой задаче не требуется рекурсия, остальные две задачи требуют применения простой рекурсии. При составлении программ (если не оговорено противное) можно использовать все встроенные функции Лиспа. Тексты всех программ, если вы мыслите в духе функционального программирования, буквально состоят из нескольких строчек.

Отладку программ можно осуществлять с помощью функции трассировки (`trace <имя функции>`), трассировка функции отключается - (`untrace <имя функции>`).

#### Пример задания с решением

##### Задача 1

Пусть `l1` и `l2` -списки. Напишите функцию, которая возвращала бы `t`, если первые два элемента этих списков соответственно равны друг другу, и `nil` – в противном случае (например, если длина одного из списков меньше 2).

##### Задача 2

Напишите функцию, зависящую от двух аргументов `x` и `u`, удаляющую все вхождения `x` в список `u` на всех уровнях.

##### Задача 3

Сортировка слиянием. Даны два упорядоченных по возрастанию списка чисел `x` и `y`. Написать функцию (`merge x y`), которая в качестве значения выдает общий упорядоченный список элементов `x` и `y`. Например, `merge('(1 3 5 7 8) '(2 3 5 7)) => (1 2 3 3 5 5 7 7 8)`.

#### Решение

##### Задача 1

```
(defun f (l1 l2)
  (and (> (length l1) 1)
    (> (length l2) 1)
    (equal (first l1) (first l2))
    (equal (second l1) (second l2))))
```

##### Задача 2

```
(defun f (x u)
  (if (null u) nil
    (let ((c (first u)) (r (rest u)))
      (cond ((and (atom c) (equal x c)) (f x r))
            ((atom c) (cons c (f x r)))
            (t (cons (f x c) (f x r)))))
  ))
)
```

## Задача 3

```
(defun merge (x y)
  (cond ((null x) y)
        ((null y) x)
        ((> (first x) (first y))
         (cons (first y) (merge x (rest y))))
        (t (cons (first x) (merge (rest x) y))))
)
```

**Варианты заданий**Вариант 1

1. Напишите функцию трех аргументов (list3 x y z) такую, что (list3 x y z) =(x y z) для любых символьных выражений; не используйте функцию list.
2. Последовательность чисел Фибоначчи 1, 1, 2, 3, 5, 8, 13... строится по следующему закону: первые два числа - единицы; любое следующее число есть сумма двух предыдущих  $f(n)=f(n-1)+f(n-2)$ . Напишите функцию (f n f1 f2) с накапливающимися параметрами f1 и f2, которая вычисляет n-ое число Фибоначчи.
3. Определите умножение целых чисел (\*2 x y) через сложение и вычитание.

Вариант 2

1. Напишите функцию, которая выдает истину, если ее аргумент удовлетворяет хотя бы одному из следующих условий:
  - а) является списком из двух элементов;
  - б) является списком из двух атомов;
  - в) является списком из трех элементов.
2. Определите возведение в целую степень ( $\wedge$  x n) через умножение и деление.
3. Напишите функцию (fulllength x), считающую полное количество атомов (не равных nil) в списке x.

Вариант 3

1. Напишите с помощью композиции условных выражений функции от четырех аргументов (and4 x1 x2 x3 x4) и (or4 x1 x2 x3 x4), совпадающие с встроенными функциями and и or от четырех аргументов.
2. Напишите функцию, вычисляющую последний элемент списка.
3. Напишите функцию от двух аргументов x и n, которая создает список из n раз повторенных элементов x.

Вариант 4

1. Напишите функцию, осуществляющую циклическую перестановку элементов в списке, т.е. (f g h j) -> (g h j f).

2. Напишите функцию, которая из данного одноуровневого списка строит список списков его элементов, например,  $(a\ b) \rightarrow ((a)\ (b))$ .
3. Определите функцию, зависящую от двух аргументов  $u$  и  $v$ , являющихся списками, которая вычисляет список всех элементов  $u$ , не содержащихся в  $v$ .

#### Вариант 5

1. Определите функцию  $(f\ a\ b\ c)$ , которая равна истине тогда и только тогда, когда из отрезков с длинами  $a, b$  и  $c$  можно построить треугольник.
2. Определите функцию, зависящую от двух аргументов  $u$  и  $v$ , являющихся списками, которая вычисляет список всех элементов, содержащихся либо в  $u$ , либо в  $v$ , но не одновременно в  $u$  и  $v$ .
3. Напишите функцию, осуществляющую замену элементов списка  $u$  на соответствующие элементы списка  $x$  в списке  $w$ , например,  
 $y=(a\ b),\ x=(1\ 2),\ w=((a\ b)\ a\ (c\ (a\ (a\ d)))) \rightarrow ((1\ 2)\ 1\ (c\ (1\ (1\ d))))$ .

#### Вариант 6

1. Определите функцию  $(f\ a\ b\ c)$ , которая вычисляет список корней квадратного уравнения  $a \cdot x^2 + b \cdot x + c = 0$  (если корней нет, то список пустой).
2. Напишите функцию, аналогичную встроенной функции замены `subst` в списке  $s$  выражения  $x$  на  $y$ , но производящую взаимную замену  $x$  на  $y$ , т.е.  $x \rightarrow y$ ,  $y \rightarrow x$ .
3. Определите функцию  $(f\ s)$ , результатом которой является список, получающийся после удаления на всех уровнях всех положительных элементов списка чисел  $s$ .

#### Вариант 7

1. Определите функцию, которая меняет местами первый и последний элементы списка, оставляя остальные на своих местах.
2. Определите функцию  $(\text{summa\_digits}\ n)$ , результатом которой является сумма цифр натурального числа  $n$ .
3. Определите функцию  $(f\ s)$ , которая из данного списка  $s$  удаляет последний элемент.

### **2.4. Второе контрольное задание**

Задание состоит из трех задач, в которых требуется составить программы на Лиспе. В первых двух задачах требуется для программирования использовать локальные или вспомогательные функции. В третьей задаче требуется использовать функционалы. При составлении программ (если не оговорено противное) можно использовать все встроенные функции Лиспа. Тексты всех программ, если вы мыслите в духе функционального программирования, буквально состоят из нескольких строчек.

## Пример задания с решением

### Задача 1

Напишите функцию, вычисляющую полное число подписков, входящих в данный список на любом уровне. Для списка (a b ((a) d) e) оно равно двум.

### Задача 2

Напишите функцию, удаляющую повторные вхождения элементов в список, например, (a b c d d a) -> (a b c d)

### Задача 3

Напишите функцию, строящую список всех подмножеств данного множества.

## Решение

### Задача 1

; Предикат (p x) выясняет является ли список одноуровневым

```
(defun p (x)
  (cond ((null x) t)
        ((listp (first x)) nil)
        (t (p (rest x)))))
)
```

```
(defun f (x)
  (cond ((atom x) 0)
        ((p x) 0)
        (t (if (listp (first x))
                (+ 1 (f (first x)) (f (rest x)))
                (f (rest x)))))
)
```

### Задача 2

```
(defun f (x)
  (labels
    ((f1 (y z)
      (cond ((null y) z)
            ((member (first y) z) (f1 (rest y) z))
            (t (f1 (rest y) (append z (list (first y)))))))
    (f1 x nil))
)
```

### Задача 3

Следующее решение принадлежит Клыкову Виктору, студенту кафедры АСУ группы 437-1.

; встроенная функция (union list1 list2) создает список всех элементов, входящих в список list1 или в list2 (объединение множеств):

```
; (union '(1 2 3) '(2 5 7 1)) => ( 7 5 1 2 3)
(defun f(s)
  (if (null s) (list s)
      (union (mapcar '(lambda (x) (cons (car s) x)) (f (cdr s)))
              (f (cdr s)))
    )
  )
)
```

## Варианты заданий

### Вариант 1

1. Определите функцию, зависящую от одного аргумента, которая по данному списку вычисляет список его элементов, встречающихся в нем более одного раза. Проверьте, как она будет работать на примере '(a a a a b a).
2. Определите функцию, зависящую от двух аргументов u и n, которая по данному списку строит список его элементов, встречающихся в нем не менее n раз. Проверьте работу этой функции на примере (a a b a c b c a b b d a b) для n=1,2,5,0.
3. Используя функционалы, напишите функцию, которая из данного списка строит список списков его элементов, например, (a b) -> ((a) (b)).

### Вариант 2

1. Определите функцию, обращающую список и все его подсписки на любом уровне, например, (a b (c d) e) -> (e (d c) b a).
2. Напишите функцию, заменяющую Y на число, равное глубине вложения Y в W, например, Y=a, W=((a b) a (c (a (a d)))) -> ((2 b) 1 (c (3 (4 d)))).
3. Напишите функцию, единственным аргументом которой являлся бы список списков, объединяющую все эти списки в один.

### Вариант 3

1. Напишите функцию, определяющую глубину первого вхождения элемента u в список w.
2. Напишите функцию, которая делает из списка множество, т.е. удаляет все повторяющиеся элементы.
3. Напишите функцию (exists p x), которая проверяет "Существует ли элемент списка x, удовлетворяющий предикату p?" (p - функция или функциональное имя).

Вариант 4

1. Напишите функцию, которая определяет является ли данное натуральное число простым.

Воспользуйтесь более общей задачей:

(ispr n m) - "Число n не делится ни на одно число большее или равное m и меньшее n".

Имеем (ispr n m) -истинно, во-первых, если  $n = m$ , и, во-вторых, если истинно (ispr n m+1) и n не делится на m.

2. Напишите функцию, которая сортирует список чисел, используя алгоритм простой вставки.

3. Напишите функцию (all p x), которая проверяет

"Для всех ли элементов списка x выполняется предикат p? "

(p - функция или функциональное имя ).

Вариант 5

1. Напишите функцию, которая сортирует список чисел, используя алгоритм простого выбора.

2. Определите функцию (f s), результатом которой является список, получающийся из списка списков s после удаления всех подсписков, содержащих числа.

3. Напишите функцию (filter p x), которая "фильтрует" (создает список) элементы списка x, удовлетворяющие предикату p

(p - функция или функциональное имя ).

Вариант 6

1. Определите функцию (f a n), которая от двух числовых аргументов вычисляет величину  $a + a*(a+1) + a*(a+1)*(a+2) + \dots + a*(a+1)*\dots*(a+n)$ .

2. Определите функцию (f s), которая вычисляет список (m1 m2 m3), состоящий из трех наибольших элементов списка s:  $m1 \geq m2 \geq m3$ .

Исходный список содержит не менее трех элементов.

3. Определите функцию (f s n), которая из списка чисел s создает новый список, прибавляя к каждому атому число n. Исходный список не предполагается одноуровневым.

Вариант 7

1. Определите функцию (f n), n кратное 3, вычисляющую сумму:

$1*2*3 + 4*5*6 + \dots + (n-2)*(n-1)*n$ .

2. Определите функцию (f s), которая в одноуровневом списке чисел s переставляет все отрицательные элементы в начало списка, например, (f'(4 -8 6 -9 -7)) -> (-8 -9 -7 4 6).

3. Определите функцию ( $f\ s$ ), которая из списка чисел  $s$  создает новый список, меняя знак у каждого атома. Исходный список не предполагается одноуровневым.

#### Вариант 8

1. Определите функцию ( $f\ s$ ), вычисляющую знакопеременную сумму  $a_1 - a_2 + a_3 - a_4 + \dots + a_k \cdot (-1)^{(k+1)}$  для списка  $s$ , имеющего вид  $(a_1\ a_2\ a_3\ \dots\ a_k)$ .

2. Определите функцию ( $f\ n$ ), которая для натурального числа  $n$  вычисляет  $1! + 2! + 3! + \dots + n!$ .

3. Напишите функцию ( $\text{count}\ p\ x$ ), которая подсчитывает, сколько атомов в списке  $x$  удовлетворяет предикату  $p$  ( $p$  - функция или функциональное имя). Список  $x$  не предполагается одноуровневым.

### 3. ЛОГИЧЕСКОЕ ПРОГРАММИРОВАНИЕ

*Почти все наши ошибки, в сущности, я языкового характера. Мы сами создаем себе трудности, неточно описывая факты. Так, например, разные вещи мы называем одинаково и, наоборот, даем разные определения одному и тому же.*

*Олдос Хаксли*

#### 3.1. Контроль обучения

В процессе дистантного обучения дисциплине "Логическое программирование" студент должен выполнить три контрольных задания, курсовую работу и обучение заканчивается выполнением компьютерной экзаменационной работы. Контрольные задания представлены ниже и состоят каждое из двух задач, требующих составить программы на Прологе. Составленные и отлаженные программы (обязательно с комментариями) студент по мере освоения логического программирования периодически пересылает по электронной почте диспетчеру центра дистантного обучения, который в свою очередь пересылает их лектору. Лектор проверяет программы и при правильном выполнении программы студент получает подтверждение о том, что они зачтены. Если программа составлена неправильно, студент получает от лектора текстовый файл, в котором содержится описание ошибок программы.

#### 3.2. Инсталляция SWI-Prolog'a

Интерпретатор с языка Пролог представлен в виде упакованного файла pl.arj. SWI-Prolog работает в среде Windows (3.11 или 95); для его установки на компьютере достаточно распаковать файл pl.arj вместе со всеми хранящимися в этом файле каталогами. Запустите файл pl.exe из каталога pl\bin и вы войдете в интерпретатор. Пролог выдает приглашение "?-" и ждет вашего ввода.

Чтобы набирать программы, надо воспользоваться каким-либо внешним текстовым редактором (SWI-Prolog не имеет собственного редактора). Например, вы можете пользоваться стандартной программой Notepad ("блокнот"). Набранные программы сохраняйте в виде текстовых файлов с расширением pl (но можно и расширение txt) в рабочем каталоге. По умолчанию рабочим каталогом будет pl\bin, но вы можете это изменить: для этого создайте ярлык для pl.exe и измените рабочий каталог в свойствах этого ярлыка. Чтобы загрузить программу из файла c:\pl\work\name.pl, находясь в среде интерпретатора, вы должны в ответ на приглашение "?-" набрать имя файла в квадратных скобках

?- [name].

если установлен рабочий каталог - pl\work, или набрать имя файла с полным путем к нему,

?-['c:\pl\work\name.pl'].

если установлен другой рабочий каталог. Отметим следующую особенность работы в SWI-Prolog с блокнотом. После набора текста программы обязательно перейдите курсором на новую строку, иначе SWI-Prolog при загрузке файла с программой выдаст ошибку "неожиданный конец файла".

### **3.3. Первое контрольное задание**

Задание состоит из двух задач, в которых требуется составить программы на Прологе для написания простых предикатов. При составлении программ (если не оговорено противное) можно использовать все встроенные предикаты Пролога. Тексты всех программ, если вы мыслите в духе логического программирования, получаются небольшие.

SWI-Prolog не имеет стандартного help'a для Windows, для этого используется предикат help. Вызов help(<имя предиката>) выдает на экран информацию об этом предикате. Вызов help(7) выдает на экран список всех встроенных предикатов с комментариями. Текстовый файл руководства по SWI-Prolog - pl\library\manual. Отладку предикатов можно осуществлять с помощью предиката трассировки trace(<имя предиката>), трассировка предиката отключается - trace(<имя предиката>, -all).

#### **Пример задания с решением**

##### **Задача 1**

Постройте предикат position\_max(+L, -M, -N), который в списке L находит максимальное значение M и порядковый номер N этого значения.

##### **Задача 2**

Определите умножение целых чисел через сложение и вычитание.

#### **Решение**

##### **Задача 1**

Будем использовать метод накапливающего параметра. Для этого введем вспомогательный предикат position\_max(+List, +I, +M0, +N0, -M, -N), где I равен номеру рассматриваемого элемента в исходном списке, M0 - текущее значение максимума, N0 - позиция текущего максимума.

```
position_max([X|T],M,N):-
    position_max(T,1,X,1,M,N).
position_max([],_,M,N,M,N).
```

```
position_max([A|T],I,X,_,M,N):-
    A>X,
    K is I+1,
    position_max(T,K,A,K,M,N).
```

```
position_max([A|T],I,X,Y,M,N):-
    A=<X,
    K is I+1,
    position_max(T,K,X,Y,M,N).
```

```
?- position_max([3,2,5,1,4,3],M,N).
M = 5
N = 3
Yes
```

## Задача 2

Определим предикат  $p(+X,+Y,?R)$ , где  $X$  и  $Y$  сомножители,  $R$  - произведение. Делаем рекурсию по второму аргументу предиката, используя рекурсивное определение  $X*Y=X*(Y-1)+X$ .

```
p(X,0,0).
```

```
p(X,Y,R):- Y>0,
            Y1 is Y-1,
            p(X,Y1,R1),
            R is R1+X.
```

```
p(X,Y,R):- Y<0,
            Y1 is -Y,
            p(X,Y1,R).
```

## Варианты заданий

### Вариант 1

1. Определите возведение в целую степень через умножение и деление.
2. Напишите предикат  $p(+L, -N)$  - истинный тогда и только тогда, когда  $N$  - предпоследний элемент списка  $L$ , имеющего не менее двух элементов.

### Вариант 2

1. Напишите предикат  $p(+X, +N, -L)$  - истинный тогда и только тогда, когда  $L$  - список из  $N$  раз повторенных элементов  $X$ .

2. Напишите предикат  $p(+L, -S)$  - истинный тогда и только тогда, когда  $S$  - список списков элементов списка  $L$ , например,  $p([a, b, c], [[a], [b], [c]])$  - истина.

### Вариант 3

1. Напишите предикат  $p(+L, -S)$  - истинный тогда и только тогда, когда  $L$  - список списков, а  $S$  - список, объединяющий все эти списки в один.
2. Напишите предикат  $p(+L, -S)$  - истинный тогда и только тогда, когда список  $S$  есть циклическая перестановка элементов списка  $L$ , например,  $p([f, g, h, j], [g, h, j, f])$  - истина.

### Вариант 4

1. Напишите предикат  $p(+X, +N, +V, -L)$  - истинный тогда и только тогда, когда список  $L$  получается после добавления  $X$  на  $N$ -е место в список  $V$ .
2. Напишите предикат  $p(+N, +V, -L)$  - истинный тогда и только тогда, когда список  $L$  получается после удаления  $N$ -го элемента из списка  $V$ .

### Вариант 5

1. Напишите предикат  $p(+V, -L)$  - истинный тогда и только тогда, когда список  $L$  получается после удаления всех повторных вхождений элементов в список  $V$ , например,  $p([a, b, c, d, d, a], [a, b, c, d])$  - истина.
2. Напишите предикат  $p(+V, -L)$  - истинный тогда и только тогда, когда список  $L$  получается после удаления из списка  $V$  всех элементов, стоящих на четных местах, например,  $p([1, 2, 3, 4, 5, 6], [1, 3, 5])$  - истина.

### Вариант 6

1. Напишите предикат  $p(+V, +X, -L)$  - истинный тогда и только тогда, когда список  $L$  получается из списка  $V$  после удаления всех вхождений  $X$  на всех уровнях, например,  $p([1, [2, 3, [1]], [3, 1]], 1, [[2, 3, []], [3]])$  - истина.
2. Напишите обобщение предиката `member`, когда ищется элемент на всех уровнях в списке.

### Вариант 7

1. Определите предикат  $p(+U, +V, -L)$  - истинный тогда и только тогда, когда список  $L$  есть список всех элементов списка  $U$ , не содержащихся в списке  $V$ .
2. Определите предикат  $p(+U, +V, -L)$  - истинный тогда и только тогда, когда  $L$  - список всех элементов, содержащихся либо в списке  $U$ , либо в списке  $V$ , но не одновременно в  $U$  и  $V$ .

### Вариант 8

1. Определите предикат  $p(+V, -L)$  - истинный тогда и только тогда, когда  $L$  - список всех элементов списка  $V$ , встречающихся в нем более одного раза.

2. Напишите предикат  $\text{subst}(+V, +X, +Y, -L)$  - истинный тогда и только тогда, когда список  $L$  получается после замены всех вхождений элемента  $X$  в списке  $V$  на элемент  $Y$ .

#### Вариант 9

1. Напишите предикат  $p(+V, -L)$  - истинный тогда и только тогда, когда список  $L$  получается из списка  $V$  после удаления всех повторяющихся элементов, т. е. из списка получается множество.
2. Напишите предикат  $\text{exists}(+P, +L)$ , который проверяет "Существует ли элемент списка  $L$ , удовлетворяющий предикату  $P$ ?"

#### Вариант 10

1. Напишите предикат  $\text{all}(+P, +L)$ , который проверяет "Для всех ли элементов списка  $L$  выполняется предикат  $P$ ?"
2. Напишите предикат  $\text{filter}(+V, +P, -L)$  - истинный тогда и только тогда, когда список  $L$  есть список всех элементов из списка  $V$ , удовлетворяющих предикату  $P$  ("фильтрация" списка).

#### Вариант 11

1. Используя предикаты "родитель"(Родитель, Отпрыск), "женщина"(Человек), "мужчина"(Человек) и "супруги"(Жена, Муж), определите отношения теща, шурин и зять.
2. Башня из кубиков может быть описана совокупностью фактов вида "на"(Кубик1, Кубик2), которые истинны, если Кубик1 поставлен на Кубик2. Определите предикат "выше"(Кубик1, Кубик2), который истинен, если Кубик1 расположен на башне выше, чем Кубик2. (Указание: "выше" является транзитивным замыканием отношения "на".)

#### Вариант 12

1. Напишите предикат  $\text{gcd}(+A, +B, -D)$  - истинный тогда и только тогда, когда  $D$  -наибольший общий делитель двух целых положительных чисел  $A$  и  $B$ .
2. Напишите программу для отношения  $\text{double}(+List, -ListList)$ , в котором каждый элемент списка  $List$  удваивается в списке  $ListList$ , например,  $\text{double}([1,2,3],[1,1,2,2,3,2])$  выполнено.

#### Вариант 13

1. Напишите новую версию предиката  $\text{length}(+L, -N)$ , в котором при подсчете количества элементов списка не учитывается пустой список. К примеру, для списка  $[a,b,c,d,e]$  новая версия процедуры должна сообщить, что длина списка равна пяти, а для списка  $[a,[],c,d,[]]$  эта процедура должна давать длину, равную трем.
2. Пусть имеется список структур "client":  $[\text{client}(a,29,3), \text{client}(b,29,6), \text{client}(c,40,2)]$ .

Первым аргументом каждой структуры служит имя клиента, вторым - суточный тариф, а третьим - количество дней, на которое взята автомашина. Напишите правило, позволяющее вычислить итоговую сумму оплаты, объединяющую выплаты всех клиентов, данные о которых содержатся в списке.

#### Вариант 14

1. Опишите процедуру для предиката `расщепить/4`, которая берет список целых чисел `L1` и целое число `N` и выдает списки `L2` и `L3` такие, что числа из исходного списка, меньшие, чем `N`, помещаются в список `L2`, а остальные - в список `L3`.
2. Напишите предикат для вычисления чисел Фибоначчи, используя метод накапливающего параметра.

#### Вариант 15

1. Напишите предикат `digits(+N, -L)` - истинный тогда и только тогда, когда `L` - список цифр натурального числа `N`.
2. Напишите предикат `summa_digits(+N, -S)` - истинный тогда и только тогда, когда `S` - сумма цифр натурального числа `N`.

### **3.4. Второе контрольное задание**

Задание состоит из двух задач, в которых требуется составить программы на Прологе для написания простых программ. При составлении программ (если не оговорено противное) можно использовать все встроенные предикаты Пролога. Тексты всех программ, если вы мыслите в духе логического программирования, получаются небольшие.

#### **Пример задания с решением**

##### Задача 1

Сортировка списка простым обменом (по возрастанию).

##### Задача 2

Пусть бинарное дерево задается рекурсивной структурой `tree(<левое поддерево>, <корень>, <правое поддерево>)` и пустое дерево задано термом `nil`. Составьте программу `subtree(+S, +T)`, определяющую, является ли `S` поддеревом `T`.

#### Решение

##### Задача 1

Заданный список `L` сортируется по следующему алгоритму:

- (1) если `L` содержит два соседних элемента, нарушающих требуемое упорядочение, то эти элементы меняются местами, после чего к полученному списку применяется этот же алгоритм;

(2) если в  $L$  не встречается ни одной неупорядоченной пары соседних элементов, то список  $L$  отсортирован и тем самым является искомым списком. Предикат  $\text{ord}(+L)$  проверяет является ли список  $L$  отсортированным.

```
ord([]).
ord([_]).
ord([X,Y|T]):-
    X <= Y,
    ord([Y|T]).

change(L,L):-
    ord(L),!.
change(L,S):-
    append(L1,[X,Y|L2],L),
    X > Y,!,
    append(L1,[Y,X|L2],Z),
    change(Z,S).
```

Отсечения в данном случае ликвидируют многочисленные правильные ответы.

## Задача 2

Для решения используется очевидная рекурсия.

```
subtree(X, X).
subtree(X, tree(L,_,_)):-
    subtree(X,L).
subtree(X, tree(_,_,R)):-
    subtree(X,R).
```

## Варианты заданий

### Вариант 1

1. Напишите предикат, аналогичный предикату  $\text{subst}$  (см. первое контрольное задание, вариант 8, задача 2), но производящий взаимную замену  $X$  на  $Y$ , т.е.  $X \rightarrow Y, Y \rightarrow X$ .

2. Напишите предикат, который определяет, является ли данное натуральное число простым.

Воспользуйтесь более общей задачей:

$\text{ispr}(N, M)$  - "Число  $N$  не делится ни на одно число большее или равное  $M$  и меньшее  $N$ ".

Имеем  $\text{ispr}(N, M)$  - истинно, во-первых, если  $N = M$ , и, во-вторых, если истинно  $\text{ispr}(N, M+1)$  и  $N$  не делится на  $M$ .

Вариант 2

1. Сортировка списка простой вставкой (по возрастанию).
2. Сортировка списка простым выбором (по возрастанию).

Вариант 3

1. Напишите предикат  $p(+L, -N)$  - истинный тогда и только тогда, когда  $N$  - количество различных элементов списка  $L$ .
2. Напишите предикат  $p(+X, +Y, -Z)$  - истинный тогда и только тогда, когда  $Z$  есть "пересечение" списков  $X$  и  $Y$ , т.е. список, содержащий их общие элементы, причем кратность каждого элемента в списке  $Z$  равняется минимуму из его кратностей в списках  $X$  и  $Y$ .

Вариант 4

1. Запрограммируйте предикат  $p(+A, +B)$ , распознающий, можно ли получить список элементов  $A$  из списка элементов  $B$  посредством вычеркивания некоторых элементов.

Алгоритм: Если  $A$  - пустой список, то ответом будет "да". В противном случае нужно посмотреть, не пуст ли список  $B$ . Если это так, то ответом будет "нет". Иначе нужно сравнить первый элемент списка  $A$  с первым элементом списка  $B$ . Если они совпадают, то надо снова применить тот же алгоритм к остатку списка  $A$  и остатку списка  $B$ . В противном случае нужно снова применить тот же алгоритм к исходному списку  $A$  и остатку списка  $B$ .

2. Напишите предикат  $p(+X, +Y, +L)$  - истинный тогда и только тогда, когда  $X$  и  $Y$  являются соседними элементами списка  $L$ .

Вариант 5.

1. Определите отношение  $\text{sum\_tree}(\text{+TreeOfInteger}, -\text{Sum})$ , выполненное, если число  $\text{Sum}$  равно сумме целых чисел, являющихся вершинами дерева  $\text{TreeOfInteger}$ .
2. Определим операторы:  
 $\text{:- op( 100, fy, ~).}$   
 $\text{:- op( 110, xfy, \&).}$   
 $\text{:- op( 120, xfy, v).}$

Булева формула есть терм, определяемый следующим образом: константы  $\text{true}$  и  $\text{false}$  - булевы формулы; если  $X$  и  $Y$  - булевы формулы, то и  $X \vee Y$ ,  $X \& Y$ ,  $\sim X$  - булевы формулы, здесь  $\vee$  и  $\&$  - бинарные инфиксные операторы дизъюнкции и конъюнкции, а  $\sim$  - унарный оператор отрицания. Напишите предикат  $p(+T)$ , определяющий, является ли данный терм  $T$  булевой формулой.

Вариант 6

1. Встроенный предикат `functor(+Term, ?Functor, ?Arity)` определяет для заданного составного термина `Term` его функтор `Functor` и местность `Arity`.

Встроенный предикат `arg(+N, +Term, ?Value)` определяет для целого числа `N` и заданного составного термина `Term` его `N`-ый аргумент `Value`.

Определите предикаты `functor1` и `arg1` - аналоги предикатов `functor` и `arg` через предикат `univ (=..)`

2. Напишите предикат `range(?M, ?N, ?L)`, истинный тогда и только тогда, когда `L` - список целых чисел, расположенных между `M` и `N` включительно (предикат должен допускать различное использование, когда не менее двух из трех аргументов конкретизованы).

(Указание. Используйте предикаты `var(+X)` и `nonvar(+X)`).

Вариант 7

1. Напишите вариант программы `plus(?X, ?Y, ?Z)`, пригодный для сложения, вычитания и разбиения чисел на слагаемые.

(Указание. Используйте для порождения чисел встроенный предикат `between(+Low, +High, ?Value)`, который порождает все целые числа от нижней границы `Low` до верхней границы `High`.)

2. Напишите программу вычисления целочисленного квадратного корня из натурального числа `N`, определяемого как число `I`, такое, что  $I * I \leq N$ , но  $(I+1) * (I+1) > N$ . Используйте определение предиката `between/3` для генерирования последовательности натуральных чисел с помощью механизма возвратов.

Вариант 8

1. Напишите новую версию процедуры "предок", которая вырабатывает список представителей всех промежуточных поколений, располагающихся между предком и потомком. Предположим, например, что Генри является отцом Джека, Джек - отцом Ричарда, Ричард - отцом Чарльза, а Чарльз - отцом Джейн. При запросе о том, является ли Генри предком Джейн, должен выдаваться список, характеризующий родственную связь этих людей, конкретно: [джек, ричард, чарльз].

2. Определите предикат `p(+V, +N, -L)` - истинный тогда и только тогда, когда `L` - список элементов списка `V`, встречающихся в нем не менее `N` раз. Проверьте работу этого предиката на примере `[a, a, b, a, c, b, c, a, b, b, d, a, b]` для `N=1,2,5,0`.

### 3.5. Третье контрольное задание

Задание состоит из двух задач, в которых требуется составить более сложные программы на Прологе (как правило, требуется определить несколько предикатов). При составлении программы (если не оговорено противное) можно использовать все встроенные предикаты Пролога. Текст программы, если вы мыслите в духе логического программирования и используете рекурсию, получается небольшой.

#### Пример задания с решением

##### Задача 1

Определим операторы:

`:- op( 100, fy, ~).`

`:- op( 110, xfy, &).`

`:- op( 120, xfy, v).`

Булева формула есть терм, определяемый следующим образом: константы `true` и `false` - булевы формулы; если  $X$  и  $Y$  - булевы формулы, то и  $X \vee Y$ ,  $X \& Y$ ,  $\sim X$  - булевы формулы, здесь  $\vee$  и  $\&$  - бинарные инфиксные операторы дизъюнкции и конъюнкции, а  $\sim$  - унарный оператор отрицания.

Напишите программу, распознающую логические формулы в конъюнктивной нормальной форме, т.е. формулы, являющиеся конъюнкцией дизъюнкций литералов, где литерал - атомарная формула или ее отрицание.

##### Задача 2

Напишите предикат `r(+N, -L)` - истинный тогда и только тогда, когда список  $L$  содержит все последовательности (списки) из  $N$  нулей, единиц и двоек, в которых никакая цифра не повторяется два раза подряд (нет куска вида  $XX$ ).

#### Решение

##### Задача 1

Из определения операций видно, что конъюнкция и дизъюнкция являются право ассоциативными, т. е. запрос

?-  $X \& Y \& Z = A \& B$

приводит к унификации  $X = A$ ,  $Y \& Z = B$ .

Определим два вспомогательных предиката `literal(X)` и `dis(X)`, которые проверяют является ли  $X$  литералом и элементарной дизъюнкцией соответственно.

`literal(true).`

`literal(false).`

`literal(~ X):-`

`literal(X).`

```
dis(X):-
    literal(X).
```

```
dis(X v Y):-
    literal(X),
    dis(Y).
```

```
con(X):- dis(X).
```

```
con(X & Y):-
    dis(X),
    con(Y).
```

?- con( true & (~ false v true v false) & ~ true).

Yes

### Задача 2

```
p(1,[[0],[1],[2]]).
```

```
p(N,R):- N>1,
    N1 is N-1,
    p(N1,R1),
    t(R1,R).
```

```
t([],[]).
```

```
t([X|T],Z):-
    t(T,R1),
    s(X,R),
    append(R,R1,Z).
```

```
s([0|T],[[1,0|T],[2,0|T]]).
```

```
s([1|T],[[0,1|T],[2,1|T]]).
```

```
s([2|T],[[0,2|T],[1,2|T]]).
```

## Варианты заданий

### Вариант 1

1. Напишите предикат  $p(+N, +K, -L)$  - истинный тогда и только тогда, когда  $L$  - список всех последовательностей (списков) длины  $K$  из чисел  $1, 2, \dots, N$ .
2. Напишите предикат  $p(+N, -L)$  - истинный тогда и только тогда, когда список  $L$  содержит все последовательности (списки) из  $N$  нулей и единиц, в которых никакая цифра не повторяется три раза подряд (нет куса вида  $XXX$ ).

Вариант 2

1. Определите отношение `ordered(+Tree)`, выполненное, если дерево `Tree` является упорядоченным деревом целых чисел, т. е. число, стоящее в любой вершине дерева, больше любого элемента в левом поддереве и меньше любого элемента в правом поддереве. Определение структуры "дерево" см. раздел "Второе контрольное задание".

Указание. Можно использовать вспомогательные предикаты `ordered_left(+X, +Tree)` и `ordered_right(+X, +Tree)`, которые проверяют, что `X` меньше (больше) всех чисел в вершинах левого (правого) поддерева дерева `Tree` и дерево `Tree` - упорядочено.

2. Определим операторы:

`:- op( 100, fy, ~).`

`:- op( 110, xfy, &).`

`:- op( 120, xfy, v).`

Булева формула есть терм, определяемый следующим образом: константы `true` и `false` - булевы формулы; если `X` и `Y` - булевы формулы, то и `X v Y`, `X & Y`, `~X` - булевы формулы, здесь `v` и `&` - бинарные инфиксные операторы дизъюнкции и конъюнкции, а `~` - унарный оператор отрицания.

Напишите программу, распознающую логические формулы в дизъюнктивной нормальной форме, т.е. формулы, являющиеся дизъюнкцией конъюнкций литералов, где литерал - атомарная формула или ее отрицание.

Вариант 3

1. Определим операторы:

`:- op( 100, fy, ~).`

`:- op( 110, xfy, &).`

`:- op( 120, xfy, v).`

Булева формула есть терм, определяемый следующим образом: константы `true` и `false` - булевы формулы; если `X` и `Y` - булевы формулы, то и `X v Y`, `X & Y`, `~X` - булевы формулы, здесь `v` и `&` - бинарные инфиксные операторы дизъюнкции и конъюнкции, а `~` - унарный оператор отрицания.

Напишите программу, задающую отношение `negation_inward(+F1,-F2)`, которое выполнено, если логическая формула `F2` получается из логической формулы `F1` внесением всех операторов отрицания внутрь конъюнкций и дизъюнкций.

Вариант 4

1. Определите предикат `occurrences(+Sub,+Term,-N)`, истинный, если число `N` равно числу вхождений подтерма `Sub` в терм `Term`. Предполагается, что терм `Term` не содержит переменных.

2. Разработайте программу "Советник по транспорту". Выберите либо сеть,

состоящую из городов, либо транспортную сеть маршрутов поездов или автобусов в пределах одного города. Вы должны информировать систему о том, откуда и куда Вы собираетесь добраться, а система должна выдавать рекомендации о том, какими поездами, автобусами, самолетами и т. д. Вам следует воспользоваться, чтобы добраться до пункта назначения.

### Вариант 5

1. Напишите предикат  $p(+S, -L)$ , который переводит предложение  $S$ , представленное строкой, в список атомов  $L$ . Например,  $p('gfrtyre hjnki <> pi\ 876\ h', [gfrtyre, hjnki, '<>', pi, 876, h])$  выполнено. Указание. Воспользуйтесь предикатом `name/2`.

2. Множественное число большинства английских существительных получается путем добавления буквы "s" к форме единственного числа. Но если существительное заканчивается буквой "y", следующей за согласной, множественное число образуется путем замены буквы "y" на сочетание "ies"; если же существительное заканчивается буквой "o", следующей за согласной, множественное число образуется путем добавления сочетания "es". Напишите утверждения для предиката `множественное_число/2`, которые задают все эти правила. Указание. Воспользуйтесь предикатом `name/2`.

### Вариант 6

1. Простейшая система кодирования сообщений заключается в замене каждой буквы сообщения на букву, находящуюся на  $N$ -й по отношению к ней позиции в алфавите. Например, для  $N=2$  буква "a" заменяется на "c", буква "y" на "a" и т.д. Зная, что коды ASCII букв от "a" до "z" изменяются от 97 до 122, напишите процедуру для предиката `шифратор/3`, который берет шифруемое слово и целое число и выдает слово, представляющее шифр данного слова, полученный с помощью указанного метода.

Указание. Воспользуйтесь предикатом `name/2`.

2. Напишите предикат `предшествует/2`, который берет два атома в качестве своих аргументов и успешно согласуется, если первый из них в лексикографическом порядке предшествует второму.

Указание. Воспользуйтесь предикатом `name/2`.

### Вариант 7

1. Одним из примеров использования предиката `name/2` может служить генерация новых атомов для представления вновь вводимых объектов, например, `abc1`, `abc2`, `abc3` и т.д. Эти имена характеризуются тем, что все они состоят из корня, определяющего тип именуемого объекта, и целочисленного суффикса для различения объектов одного типа. Напишите программу `новое_имя(+X, -Y)`. Последовательность имен создается с помощью возвратов. Указание. Воспользуйтесь предикатом `int_to_atom(+N,-X)`, который конвертирует натуральное число  $N$  в атом  $X$ .

2. Построить программу "сжать", назначение которой - преобразование английских слов в их "звуковой" код. Этот процесс предусматривает "сжатие" примерно одинаково звучащих слов в одинаковый их код - своего рода, аббревиатуру этих слов. Слова "сжимаются" в соответствии со следующими правилами:

- первая буква слова сохраняется;
- все последующие за ней гласные, а также буквы "h", "w" и "y" удаляются;
- двойные буквы заменяются одиночными;
- закодированное слово состоит не более чем из четырех букв, остальные буквы удаляются.

Примеры: сжать(barrington, brng) и сжать(llewellyn, ln) - выполнено.

Указание. Воспользуйтесь предикатом name/2.

#### 4. КУРСОВЫЕ РАБОТЫ

Выполнение курсовой работы по логическому программированию ("логическое программирование" понимается в "широком" смысле этого слова) требует решения одной из нижеперечисленных задач и, как результат, создания программы на Прологе или Лиспе и написания пояснительной записки к работе. Созданную программу и пояснительную записку (в электронном виде) студент пересылает по электронной почте диспетчеру центра дистантного обучения, который в свою очередь пересылает их лектору. Лектор проверяет программу и пояснительную записку и при правильном выполнении работы студент получает подтверждение о том, что они зачтены. Если программа или записка составлена неправильно, студент получает от лектора текстовый файл, в котором содержится описание ошибок программы и недостатков пояснительной записки.

Для некоторых курсовых работ вам потребуются некоторые знания из теории графов. В качестве литературы можно использовать любой учебник по дискретной математике, содержащий теорию графов в небольшом объеме. Например,

Нефедов В. Н., Осипова В. А. Курс дискретной математики. - М.: Изд-во МАИ, 1992.-264 с.

Независимо от того, на каком языке требуется выполнить курсовую работу, на Лиспе или Прологе, представляйте граф в виде двух списков: список вершин и список ребер. Каждое ребро, в свою очередь, есть список из двух вершин.

Следующие два алгоритма для курсовых работ с графами возможно будут полезны.

##### **Алгоритм поиска в глубину в графе для реализации на Лиспе**

Функция (depth V,E,x,y,r,end) выдает путь (список вершин):

V - список вершин графа;

E - список ребер;

x - стартовая (начальная) вершина, при рекурсивном вызове depth, x - текущая вершина, откуда ведется поиск пути;

y - список вершин - соседей вершины x;

r - накапливаемый путь (накапливающий параметр), в начале поиска - пустой список, вершины накапливаются в обратном пройденному порядке;

end - предикат (функциональный аргумент), которому должна удовлетворять целевая (конечная) вершина искомого пути.

If x- целевая вершина ,

then получаем результат , добавляя к пути r вершину x, else

if список y вершин-соседей пуст then ответ = nil else

if первая вершина в списке  $y$  принадлежит пройденному пути  $p$   
     then вызываем рекурсивно функцию  $depth$  для хвоста списка  $y$  else  
 if первая вершина в списке  $y$  не принадлежит пройденному пути  $p$   
     then вызываем рекурсивно функцию, накапливая параметр  $p$  и  
         меняя параметры  $x$  и  $y$   
 else вызываем рекурсивно функцию  $depth$  для хвоста списка  $y$ .

### Алгоритм поиска в глубину для Пролога

Описание необходимых предикатов:

$next(+X, ?Y)$  - выдает истину тогда и только тогда, когда  $X$  и  $Y$  - смежные вершины;

$path(+Start, -Path)$  - вычисляет путь  $Path$  (список вершин в порядке, обратном пройденному) из начальной вершины  $Start$  в целевую вершину, удовлетворяющую некоторому предикату  $end$ ;

$depth(+P, +X, -Path)$  - предикат, вычисляющий путь  $Path$  из стартовой вершины в целевую,  $P$  - накапливающийся параметр, содержащий уже пройденный путь, прежде чем попасть в вершину  $X$ .

Предикат  $path$  вычисляется с помощью правила

```

path(Start, Path):-
    depth([], Start, Path).
  
```

Алгоритм для вычисления  $depth(P, X, Path)$ :

- 1) если  $X$  - целевая вершина, то результатом является путь  $P$ , к которому добавлена вершина  $X$  ;
- 2) в противном случае находим соседнюю вершину  $Y$  для  $X$ , которая не входит в пройденный путь  $P$ , добавляем вершину  $X$  к пути  $P$  и рекурсивно вызываем  $depth$ .

## ВАРИАНТЫ ТЕМ ДЛЯ КУРСОВЫХ РАБОТ

### 1. Упрощение электрических цепей

Постановка задачи.

Цепь состоит из компонентов только трех видов: резисторов, емкостей и индуктивностей. Преобразовать цепь в более простую, используя упрощающие правила при параллельном и последовательном соединении компонент одного вида. Треугольники преобразовывать в звезды. Язык программирования: SWI-prolog.

Используйте следующее представление предметной области. Структуры  $r$ (Номинал),  $l$ (Номинал) и  $c$ (Номинал) изображают соответственно резистор, индуктивность и емкость с номиналами. Вся цепь представляется в базе данных фактами вида

```
comp(<метка элемента>,
      <элемент: резистор, индуктивность или емкость>,
      <список узлов элемента>).
```

Упрощение цепи сводится к изменению базы данных.

### 2. Программа для алгебраических вычислений

Объекты, с которыми вы будете работать, - это многочлены от нескольких переменных, представленных в символьном виде с вещественными коэффициентами. Многочлены должны изображаться как арифметические выражения, так умножение изображается знаком '\*', а возведение в степень - знаком '^'. Язык реализации - SWI-Prolog.

Для манипуляций с многочленами нужны некоторые предикаты, чтобы пользователь мог получать ответы на вопросы, на которые не удастся ответить с помощью традиционных языков программирования. Для этого вам понадобится обозначать многочлены идентификаторами и хранить в базе данных пару (имя многочлена, сам многочлен). Предикаты выполняют некоторые операции над своими операндами и помещают результат в качестве значения некоторого имени многочлена.

Список предикатов.

1. Ввести многочлен и записать его под некоторым именем.
2. Образовать алгебраическую сумму (разность, произведение) двух многочленов и записать полученный многочлен под некоторым именем.
3. Возвести данный многочлен в целую степень и результат записать под некоторым именем.
4. Заменить каждое вхождение некоторой переменной в многочлене на данный многочлен и результат записать под некоторым именем.
5. Вычислить производную многочлена по переменной и результат записать под некоторым именем.
6. Напечатать данный многочлен.

Многочлен представлять в виде суммы членов, включающих только операции умножения и возведения в степень. В каждом таком одночлене все константы перемножены и образуют числовой коэффициент (первый сомножитель), переменные упорядочены по алфавиту и все степени одной переменной объединены так, что каждая переменная встречается лишь один раз. Следует приводить подобные члены, т. е. объединять одночлены, имеющие одинаковые наборы переменных и степеней, с соответствующим изменением коэффициентов.

Литература (полезная, но можно без нее обойтись)

1. Уэзерелл Ч. Этюды для программистов: Пер. с англ.-М.: Мир,1982.- С. 114-120.

2. Стерлинг Л., Шапиро Э. Искусство программирования на языке ПРОЛОГ.- М.: Мир.- С. 57-61.

Указания. Некоторые фрагменты программы:

```
% Полиномы хранятся в виде фактов
% pol(+Name, -<список одночленов>).
```

```
% Ввод полинома: enter(+Name)
```

```
enter(X):-
    write('Введите полином с именем '),write_ln(X),
    enter(X,[],_).
```

```
enter(X,S,P):-
    write_ln('Введите очередной одночлен или end'),
    read(T),
    ((T=end,place(X,S,P));
    (T \= end,
    enter(X,[T|S],P))).
```

```
% привести подобные во введенном полиноме, упорядочить члены
% загрузить в базу данных
```

```
place(X,S,P):-
    merge(S,[],P),
    retractall(pol(X,_)),
    assert(pol(X,P)).
```

```
% сложение полиномов, представленных списками одночленов
merge([],L,L).
merge([X|L1],L2,L):-
    insert(X,L2,L3),
```

```
merge(L1,L3,L).
```

```
insert(X,[],[X]).
insert(X,[Y|T],[Z|T]):-
    equal(X,Y,Z),Z \= 0,!.
insert(X,[Y|T],T):-
    equal(X,Y,0),!.
insert(X,[Y|T],[X,Y|T]):-
    low(X,Y).
insert(X,[Y|T],[Y|T1]):-
    not(low(X,Y)),
    insert(X,T,T1).
```

/\* equal(X,Y,Z) - из двух мономов X, Y при приведении подобных получаем Z

low(X,Y) - моном X предшествует моному Y \*/

% приведение полинома к более "читабельному" виду

```
canon([],0).
canon([X],X).
canon([X,Y],Y+X).
canon([X,Y|T],Z+Y+X):-
    length(T,M),M>0,
    canon(T,Z).
```

% показать полином

```
show(X):-
    pol(X,P),
    canon(P,P1),
    write('Полином с именем '),write_ln(X),write_ln(P1).
```

% сложение полиномов

```
plus_pol(X,Y,Z):-
    pol(X,P1),
    pol(Y,P2),
    merge(P1,P2,P),
    retractall(pol(Z,_)),
    assert(pol(Z,P)),
    show(Z).
```

Протокол:

?- enter(a).

Введите полином с именем a

Введите очередной одночлен или end  
 $-8 * x^5$ .

Введите очередной одночлен или end  
 9.

Введите очередной одночлен или end  
 $10 * x^5$ .

Введите очередной одночлен или end  
 $x^3$ .

Введите очередной одночлен или end  
 $-7 * x^3$ .

Введите очередной одночлен или end  
 end.

Yes

?- show(a).

Полином с именем a

$2 * x^5 + -6 * x^3 + 9$

Yes

?-plus\_pol(a,a,b).

Полином с именем b

$4 * x^5 + -12 * x^3 + 18$

Yes

### 3. Игра "Суммируйте до 20"

Два игрока по очереди называют какое-либо число в интервале от 1 до 3 включительно. Названные числа суммируются, выигрывает тот игрок, сумма чисел после хода которого равна 20. Напишите программу на SWI-Prolog, выигрывающую данную игру на стороне компьютера.

Указания. Используйте запоминающие функции (см. "Курс лекций по логическому программированию"). Напишите программу в таком виде, чтобы ее можно было легко модернизировать для решения более общей задачи. Более общая задача: перед началом игры компьютер спрашивает "Какой список допустимых ходов (какие числа можно называть) и какая предельная (выигрышная) сумма?"

### 4. Задача Прима-Краскала ("жадные" алгоритм) на Прологе

*Дана плоская страна и в ней городов. Нужно соединить все города телефонной связью так чтобы общая длина телефонных линий была минимальной.*

Уточнение задачи. В задаче речь идет о телефонной связи, т. е. подразумевается *транзитивность* связи: если *i*-й город связан с *j*-ым, а *j*-ый с

$k$ -ым, то  $i$ -й связан с  $k$ -ым. Подразумевается также, что телефонные линии могут разветвляться только на телефонной станции, а не в чистом поле. Наконец, требование минимальности (вместе с транзитивностью) означает, что в искомом решении не будет циклов. В терминах теории графов задача Прима-Краскала выглядит следующим образом:

*Дан граф с  $n$  вершинами; заданы длины ребер. Найти остовное дерево минимальной длины.*

Как известно, дерево с  $n$  вершинами имеет  $n-1$  ребер. Оказывается, каждое ребро надо выбирать жадно (лишь бы ни возникали циклы).

Алгоритм Прима-Краскала  
(краткое описание)

В цикле  $n-1$  раз делай:

"выбрать самое короткое еще не выбранное ребро при условии, что оно не образует цикл с уже выбранными".

Выбранные таким образом ребра образуют искомое остовное дерево. Напишите программу для решения задачи Прима-Краскала на SWI-Prolog.

### **5. Задача Прима-Краскала ("жадный" алгоритм) на Лиспе**

Решите задачу Прима-Краскала (см. предыдущую тему) на языке XLisp.

### **6. Упрощение арифметических выражений**

Назовем арифметическим выражением терм, при конструировании которого используются только атомы, числа, скобки и знаки арифметических операций. Напишите программу для упрощения арифметических выражений на SWI-Prolog.

В целом, задача упрощения выражений является достаточно сложной и в каком-то смысле неконкретизованной, т. к. единого верного решения для этой задачи нет. Если арифметическое выражение имеет несколько вариантов более простого представления, то какой из них выбрать в качестве решения? Это зависит от того, для каких целей нам необходимо упрощение.

Задачу упрощения выражения поставим следующим образом. Необходимо найти эквивалентное выражение, форма записи которого является более короткой, чем форма записи исходного выражения. Для упрощения выражений используйте различные рекурсивные "правила переписывания", каждое из которых "упрощает" какое-нибудь подвыражение в исходном выражении. Правила переписывания должны соответствовать обычным математическим преобразованиям, как-то: приведению подобных и т. п., и представляются правилами Пролога (см. программу "дифференцирование выражений" из курса лекций).

Ваша программа должна быть некоторым компромиссом между желательной простотой написания и той сложностью, которой от нее требует поставленная задача.

Литература (не обязательная): Лорьер Ж.-Л. Системы искусственного интеллекта. - М., Мир, 1991.- С. 129-131, 471.

### **7. Определение связности графа на Прологе**

Напишите программу на SWI-Prolog, определяющую является ли данный неориентированный граф связным.

Указание: запрограммируйте предварительно предикат `path(+X,+Y)`, проверяющий, существует ли путь из вершины X в вершину Y.

### **8. Определение связности графа на Лиспе**

Напишите программу на языке XLisp, определяющую, является ли данный неориентированный граф связным.

Указание: запрограммируйте предварительно предикат `(path X Y)`, проверяющий, существует ли путь из вершины X в вершину Y.

### **9. Определение эйлерова пути на Прологе**

Напишите программу на SWI-Prolog, определяющую эйлеровый путь, начинающийся с заданной вершины в неориентированном графе. Путь называется эйлеровым, если проходит через все ребра графа по одному разу. Теорема Эйлера утверждает, что такой путь всегда существует, если количество вершин в графе с нечетной степенью равно 0 или 2. Степень вершины - это количество ребер, которые инцидентны данной вершине. Если количество вершин с нечетной степенью равно 2, то эйлеровый путь всегда начинается в одной из таких вершин.

### **10. Определение эйлерова пути на Лиспе**

Напишите программу на языке XLisp, определяющую эйлеровый путь, начинающийся с заданной вершины в неориентированном графе (см. предыдущую тему).

### **11. Определение компонент связности на Прологе**

Напишите программу на SWI-Prolog, определяющую компоненты связности данного неориентированного графа. Каждая компонента связности - список вершин, следовательно, решением задачи должен быть список списков.

Указание: запрограммируйте предварительно предикат `path(+X,+Y)`, проверяющий, существует ли путь из вершины X в вершину Y.

## 12. Определение компонент связности на Лиспе

Напишите программу на языке XLisp, определяющую компоненты связности данного неориентированного графа. Каждая компонента связности - список вершин, следовательно, решением задачи должен быть список списков.

Указание: запрограммируйте предварительно предикат (path X Y), проверяющий, существует ли путь из вершины X в вершину Y.

## 5. КАК ВЫПОЛНЯТЬ КУРСОВУЮ РАБОТУ И ОФОРМЛЯТЬ ПОЯСНИТЕЛЬНУЮ ЗАПИСКУ

### Общие положения

Основные задачи и цели курсового проектирования:

- 1) приобретение навыков и методов программирования достаточно сложных задач на языках логического программирования;
- 2) подготовка к выполнению дипломного проекта.

В курсовой работе должна быть разработана тема в соответствии с заданием, одобренным кафедрой.

### Общие требования к построению пояснительной записки (ПЗ)

#### Структура построения ПЗ

ПЗ к работе должна содержать следующие разделы:

- 1) титульный лист;
- 2) аннотацию;
- 3) задание на проектирование;
- 4) содержание;
- 5) введение;
- 6) основная часть работы;
- 7) заключение;
- 8) список литературы;
- 9) приложения.

#### Титульный лист

Титульный лист оформляется согласно ГОСТ 2.105-79, форма титульного листа приведена в приложении 1.

#### Реферат

Реферат - краткая характеристика работы с точки зрения содержания, назначения, формы и других особенностей. Перечисляются ключевые слова

работы, указывается количество страниц и приложений. Реферат размещают на отдельной странице. Заголовком служит слово "Реферат", написанное прописными буквами.

### Задание на проектирование

Форма задания заполняется студентом в соответствии с полученным заданием. Форма задания приведена в приложении 2.

### Содержание

Содержание включает наименования всех разделов, подразделов и пунктов, если они имеют наименование, а также список литературы и приложения с указанием номера страниц, на которых они начинаются. Слово "Содержание" записывается в виде заголовка, симметрично тексту, прописными буквами. Пример оформления содержания приведен в приложении 3.

### Введение

Введение содержит основную цель курсовой работы, область применения разрабатываемой темы.

### Заключение

Заключение должно содержать краткие выводы по выполненной работе. Также следует указать, чему программист научился на примере этой задачи (на этот вопрос легко ответить, если сформулировать его в виде: "Что я в следующий раз сделаю иначе?").

### Список литературы

В список литературы входят все те и только те источники литературы, на которые имеются ссылки в ПЗ. Примеры библиографических описаний источников, помещаемых в список литературы, приведены в приложении 4.

### Приложения

Приложения содержат вспомогательный материал: листинг программы и листинг тестов.

Программа должна быть самодокументированная, т.е.

- программа должна иметь простую и понятную структуру,
- в программе должны быть прокомментированы используемые структуры данных,
- для каждой функции должно быть указано, что она делает, что является входными данными и результатом,
- должен быть прокомментирован используемый алгоритм.

## Основная часть курсовой работы

В основной части должно быть решение поставленной задачи, в частности:

- анализ задачи;
- обоснование выбора алгоритма;
- обоснование выбора структур данных;
- описание алгоритма;
- обоснование набора тестов.

### Об анализе задачи

Разработка алгоритма представляет собой задачу на построение. Поэтому, как обычно в таких случаях (можно, например вспомнить о методе решения геометрических задач на построение), необходим этап анализа задачи. Он позволяет установить, что является входом и выходом будущего алгоритма, выделить основные необходимые отношения между входными и выходными объектами и их компонентами, выделить подцели, которые нужно достичь для решения задачи, и как следствие этого, выработать подход к построению алгоритма. Результатом этапа анализа задачи должна быть спецификация алгоритма, т. е. формулировка в самом общем виде того, что (в рамках выбранного подхода) должен делать алгоритм, чтобы переработать входные данные в выходные.

### Об описании алгоритма

Прежде всего, нужно иметь в виду, что такое описание предназначено не для машины, а для человека. Другими словами, речь идет не о программе, а о некотором тексте (т. е. о словесном описании), по которому можно получить представление об общей структуре разрабатываемого алгоритма, о смысле его отдельных шагов и их логической взаимосвязи. Сохранение достаточно высокого уровня описания алгоритма также облегчает его обоснование. Поэтому шаги алгоритма должны описываться в терминах тех объектов и отношений между ними, о которых идет речь в формулировке задачи. Например, для "геометрической" задачи шаги алгоритма следует описывать как действия над точками, прямыми и т. п., как проверки свойств типа принадлежности трех точек одной прямой и т. п. Но не должно быть работы с кодами этих объектов, например с матрицей координат точек некоторого множества.

При программировании на Прологе описание предикатов должно заключаться в указании, для каких отношений между сущностями (объектами предметной области) они введены. Какие аргументы предиката являются входными, а какие выходными? При программировании на Лиспе для описания функций должно быть указано, что функция вычисляет, а не как она это делает.

Нужно подобрать набор тестов, достаточный для демонстрации работы программы и ее реакции на экстремальные ситуации и неправильное обращение.

### **Правила оформления ПЗ к курсовой работе**

ПЗ пишется в редакторе MS Word шрифтом Times New Roman, размером 12, на формате А4. Нумерация страниц должна быть сквозной, первой страницей является титульный лист. Номер страницы проставляется вверху посередине. Заголовки разделов пишутся прописными буквами по середине текста. Заголовки подразделов пишутся с абзаца строчными буквами, кроме первой прописной. В заголовке не допускаются переносы слов. Точку в конце заголовка не ставят. Если заголовок состоит из двух предложений, то их разделяют точкой.

## 6. ЭКЗАМЕНАЦИОННЫЕ ВОПРОСЫ

Данные вопросы предлагаются для компьютерной автоматизированной проверки знаний студентов по дисциплинам "Логическое программирование" и "Функциональное программирование".

### Вопрос 1

Дана цель для интерпретатора Пролога

?- parent(X,pat).

Какое из следующих двух предложений правильно передает логический смысл этого запроса:

- 1) "Все ли X являются родителями Pat?"
- 2) "Существует ли X, который является родителем Pat?"

Варианты ответов:

- 1) первое предложение;
- 2) второе предложение;
- 3) оба предложения не передают логический смысл запроса;
- 4) оба предложения правильны.

Введите номер правильного ответа.

### Вопрос 2

Дана цель для интерпретатора Пролога

?- not parent(X,pat).

на которую был получен отрицательный ответ. Какое из следующих трех предложений правильно передает логический смысл этого ответа:

- 1) "Не существует родитель у Pat."
- 2) "Не все X являются родителями Pat."
- 3) "Есть родители, но не у Pat."

Варианты ответов:

- 1) первое предложение;
- 2) второе предложение;
- 3) третье предложение;
- 4) все три предложения не передают логический смысл ответа.

Введите номер правильного ответа.

### Вопрос 3

Пусть имеется правило на Прологе

'имеет ребенка'(X):-'родитель'(X, Y).

Какое из следующих двух предложений правильно передает логический смысл этого правила:

- 1) "Для всех X и Y, если X - родитель Y, то X имеет ребенка."

- 2) "Для всех  $X$ ,  $X$  имеет ребенка, если существует некоторый  $Y$ , такой, что  $X$  - родитель  $Y$ ."

Варианты ответов:

- 1) первое предложение;
- 2) второе предложение;
- 3) оба эти предложения передают логический смысл правила;
- 4) оба эти предложения не передают логический смысл правила.

Введите номер правильного ответа.

#### Вопрос 4

Пусть имеются следующие два варианта правила для предиката 'сестра'( $X, Y$ ) - "X есть сестра Y" на Прологе (мы вкладываем естественный смысл для используемых в правиле предикатов):

- 1) 'сестра'( $X, Y$ ):-'родитель'( $Z, X$ ), 'родитель'( $Z, Y$ ), 'женщина'( $X$ ).
- 2) 'сестра'( $X, Y$ ):- 'родитель'( $Z, X$ ), 'родитель'( $Z, Y$ ), 'женщина'( $X$ ), 'различны'( $X, Y$ ).

Какое из этих правил более правильно с точки зрения логики?

Варианты ответов:

- 1) первое;
- 2) второе;
- 3) оба неправильны;
- 4) правила логически эквивалентны.

Введите номер правильного ответа.

#### Вопрос 5

Пусть дано отношение 'родитель' на Прологе. Определим отношение "X - родственник Y" следующим образом :

'родственник'( $X, Y$ ):-'родитель'( $X, Y$ ).

'родственник'( $X, Y$ ): - 'родитель'( $Y, X$ ).

'родственник'( $X, Y$ ):-'родственник'( $X, Z$ ), 'родственник'( $Z, Y$ ).

Какие из следующих утверждений верны?

1. Одно из первых двух правил лишнее.
2. Третье правило потенциально опасное - оно может привести в некоторых запросах к бесконечной рекурсии.
3. Любой запрос к данной программе приводит к конечной работе интерпретатора Пролога.
4. Правила неправильно задают отношение 'родственник'.

Введите через пробел номера утверждений (в порядке возрастания), которые вы считаете истинными.

#### Вопрос 6

Пусть, используя отношение 'предок' (см. курс лекций), введено отношение 'родственник':

'родственник'(X,Y):-'предок'(X,Y).  
 'родственник'(X,Y):-'предок'(Y,X).  
 'родственник'(X,Y):-'предок'(Z,X),'предок'(Z,Y).  
 'родственник'(X,Y):-'предок'(X,Z),'предок'(Y,Z).

Правильно ли определено отношение?

Варианты ответов:

- 1) да;
- 2) нет;
- 3) правильно, но при положительном ответе на некоторые ваши запросы интерпретатор будет повторять одинаковые ответы.

Введите номер правильного ответа.

### Вопрос 7

Пусть дана программа на Прологе:

'родитель'('пам', 'боб').  
 'родитель'('том', 'боб').  
 'родитель'('том', 'лиз').  
 'родитель'('боб', 'энн').  
 'родитель'('боб', 'пат').  
 'родитель'('пат', 'джим').  
 'женщина'('пам').  
 'женщина'('лиз').  
 'женщина'('энн').  
 'женщина'('пат').  
 'мать'(X, Y):- 'родитель'(X, Y), 'женщина'(X).  
 'родитель родителя'(X, Y):- 'родитель'(X, Z), 'родитель'(Z, Y).

Постарайтесь понять, как Пролог выводит ответы на указанные ниже вопросы. Будут ли встречаться возвраты при выводе ответов на какие-либо из этих вопросов?

- 1) ?-'родитель'('пам', 'боб').
- 2) ?-'мать'('пам', 'боб').
- 3) ?-'родитель родителя'('пам', 'энн').
- 4) ?-'родитель родителя'('боб', 'джим').

Введите последовательность из четырех ответов через пробелы в порядке нумерации (варианты ответов: да, нет).

### Вопрос 8

Следующие выражения представляют собой правильные объекты в смысле Пролога:

- 1) Diana; 2) diana; 3) 'Diana'; 4) 'Диана едет на юг'; 5) 'едет'('диана', 'юг'); 6) 45;
- 7) +(X,Y); 8) three(black (Kat)).

Что это за объекты (атомы, числа, переменные, структуры)?

Введите последовательность из 8 ответов через пробелы (ответы кодируйте числами: атом -1, число - 2, переменная- 3, структура - 4).

### Вопрос 9

Пролог. Будут ли следующие операции унификации успешными или неуспешными?

- 1)  $\text{point}(A, B) = \text{point}(1, 2)$
- 2)  $\text{point}(A, B) = \text{point}(X, Y, Z)$
- 3)  $\text{plus}(2, 2) = X$
- 4)  $+(2, D) = +(E, 2)$
- 5)  $t(\text{point}(-1, 0), P2, P3) = t(\text{point}(P1, \text{point}(1, 0), \text{point}(0, Y)))$

Введите последовательность из пяти ответов через пробелы в порядке нумерации (варианты ответов: да, нет).

### Вопрос 10

Пролог. При выполнении цели

?-  $P = \text{point}(2, X)$ .

система унифицирует  $X = G692$ ,  $P = \text{point}(2, G692)$ . Имя G692 - законное имя прологовской переменной, которое система построила сама во время вычислений. Ей приходится генерировать новые имена, для того чтобы переименовывать введенные пользователем переменные в программе.

В качестве объяснения этому можно предложить:

- 1) одинаковые имена обозначают в разных предложениях разные переменные;
- 2) при последовательном применении одного и того же предложения используется каждый раз его "копия" с новым набором переменных.

Вам требуется выбрать верное объяснение.

Варианты ответов:

- 1) первая причина правильная;
- 2) вторая причина правильная;
- 3) верны обе причины;
- 4) обе причины неправильны.

Введите номер правильного ответа.

### Вопрос 11

Пролог. Рассмотрим следующую программу:

$f(1, \text{one})$ .

$f(s(1), \text{two})$ .

$f(s(s(1)), \text{three})$ .

$f(s(s(s(X))), N) :- f(X, N)$ .

Сколько ответов пролог-система даст на каждый из следующих вопросов?

- 1) ?-  $f(s(1), A)$ .
- 2) ?-  $f(s(s(1)), \text{two})$ .

3) ?- f(s(s(s(s(s(1)))))), C).

4) ?- f(D, three).

Введите последовательность из четырех ваших ответов через пробелы в порядке нумерации (каждый ваш ответ - натуральное число, представляющее количество ответов пролог-системы на вопрос).

### Вопрос 12

Дана программа на Прологе.

'большой'('медведь').

'большой'('слон').

'маленький'('кот').

'коричневый'('медведь').

'черный'('кот').

'серый'('слон').

'темный'(Z):-'черный'(Z).

'темный'(Z):-'коричневый'(Z).

В каком из следующих двух случаев системе приходится производить большую работу для нахождения ответа?

1) ?- 'большой'(X), 'темный'(X).

2) ?- 'темный'(X), 'большой'(X).

Указание. Время работы оценивайте в количестве шагов; на каждом шаге вычисляется новый список целей, список целей меняется после каждой успешной унификации или возврата.

Введите номер запроса.

### Вопрос 13

В Прологе функтор "." с двумя аргументами можно использовать для представления списков наряду с более удобным представлением с квадратными скобками. Представьте список [1,2,3,4] с помощью функтора ".". Введите ответ - строку символов без пробелов.

### Вопрос 14

Пролог. Какое из следующих представлений списка эквивалентно представлению [a, b, c]?

1) [a|[b, c]]

2) [a, b|[c]]

3) [a, b, c|[]]

4) [a, .(b|[c])]

Введите последовательность из четырех ответов через пробелы в порядке нумерации (варианты ответов: да, нет).

**Вопрос 15**

Пролог. Какую конкретизацию получит переменная L в результате вычисления цели

?- L1 = [a,b,z,z,c,z,z,z,d,e], append(L, [z,z,z|\_],L1).

Введите ответ - строку символов без пробелов.

**Вопрос 16**

Пролог. Какую конкретизацию получит переменная L в результате вычисления следующей цели?

?- append(L, [\_,\_,\_],[a,b,c,d]).

Введите ответ - строку символов без пробелов.

**Вопрос 17**

Пролог. Какую конкретизацию получит переменная L в результате вычисления следующей цели?

?- append([\_,\_,\_]L, [\_,\_,\_],[0,1,2,3,4,5,6,7]).

Введите ответ - строку символов без пробелов.

**Вопрос 18**

Определен следующий прологовский предикат

$p(X, [X])$ .

$p(X, [H|T])$ :-  $p(X, T)$ .

Какую конкретизацию получит переменная L в результате вычисления следующей цели?

?-  $p(L, [1,2,3,4,5])$ .

Введите ответ - строку символов без пробелов.

**Вопрос 19**

Определен следующий прологовский предикат

$p([], [])$ .

$p([H|T], L)$ :-  $p(T, L1)$ , append(L1, [H], L).

Какую конкретизацию получит переменная L в результате вычисления следующей цели?

?-  $p([1,2,3], L)$ .

Введите ответ - строку символов без пробелов.

**Вопрос 20**

Определен следующий прологовский предикат

$p([H|T], L)$ :- append(T, [H], L).

Какую конкретизацию получит переменная L в результате вычисления следующей цели?

?-  $p([1,2,3], L)$ .

Введите ответ - строку символов без пробелов.

### Вопрос 21

Дана прологовская программа

t(zero, 0).

t(one, 1).

t(two, 2).

p([], []).

p([H|T], [H1|L]) :- t(H, H1), p(T, L).

Какую конкретизацию получит переменная L в результате вычисления следующей цели?

?- p([one, zero, two], L).

Введите ответ - строку символов без пробелов.

### Вопрос 22

Пролог. Дано определение предиката

'разбиение списка'([], [], []).

'разбиение списка'([X], [X], []).

'разбиение списка'([X, Y|T], [X|T1], [Y|T2]) :- 'разбиение списка'(T, T1, T2).

Какой список получится длиннее при выполнении цели

?- 'разбиение списка'([1,2,3,4,5], L1, L2).

L1 или L2?

Варианты ответов:

1) список L1;

2) список L2;

3) списки равны по длине.

Введите номер правильного ответа.

### Вопрос 23

Пролог. Определим предикат length для вычисления длины списка:

length([], 0).

length([\_|T], N) :- length(T, N1), N is 1+N1.

Если во втором правиле в его теле поменять две цели местами, то при вызове

?- length([1, 2, 3], N).

произойдет следующее:

1) интерпретатор не сможет вычислить цель, а сообщит об ошибке;

2) N получит значение равное 3;

3) цель успешно вычислится, но N в качестве значения получит не число;

4) интерпретатор ответит: No.

Введите номер правильного ответа.

**Вопрос 24**

Пролог. Определим предикат `length` для вычисления длины списка:

`length([], 0).`

`length([_|T], N) :- length(T, N1), N = 1+N1.`

При вызове

?- `length([1, 2, 3], N).`

произойдет следующее:

- 1) интерпретатор не сможет вычислить цель, а сообщит об ошибке;
- 2) N получит значение равное 3;
- 3) цель успешно вычислится, но N в качестве значения получит не число;
- 4) интерпретатор ответит: No.

Введите номер правильного ответа.

**Вопрос 25**

Пролог. Определим предикат `length` для вычисления длины списка:

`length([], 0).`

`length([_|T], N) :- N = 1+N1, length(T, N1),`

При вызове

?- `length([1, 2, 3], N).`

произойдет следующее:

- 1) интерпретатор не сможет вычислить цель, а выдает сообщение об ошибке;
- 2) N получит значение равное 3;
- 3) цель успешно вычислится, но N в качестве значения получит не число;
- 4) интерпретатор ответит: No.

Введите номер правильного ответа.

**Вопрос 26**

Пролог. Пусть мы определили некоторый предикат `p(X, Y)`, для которого X является входным параметром, а Y - выходным параметром.

Можем ли мы вызывать этот предикат в обратной ситуации, когда Y - конкретизованный аргумент, а X - переменная?

Варианты ответов

- 1) да,
- 2) нет,
- 3) достаточно часто, зависит от конкретного предиката.

Введите номер правильного ответа.

**Вопрос 27**

Пусть имеется прологовская программа

`p(1).`

`p(2):-!.`

`p(3).`

Какие будут конкретизации пролог-системы на следующий запрос?

?- p(X).

Введите последовательность конкретизаций переменной X - строку символов из чисел разделенных пробелами.

### Вопрос 28

Пусть имеется прологовская программа

p(1).

p(2):-!.

p(3).

Какие будут конкретизации пролог-системы на следующий запрос?

?- p(X),p(Y).

Варианты ответов:

1) X=1, Y=1

2) X=1, Y=1; X=1, Y=2; X=2, Y=1; X=2, Y=2

3) X=1, Y=1; X=1, Y= 2

Введите номер правильного ответа.

### Вопрос 29

Пролог. Дано определение предиката

'различны'(X, Y):- !, fail.

'различны'(X, Y).

который выполняется, если X и Y не совпадают.

В каком смысле понимаются "различны":

1) X и Y не совпадают буквально;

2) X и Y не унифицируемы;

3) значения арифметических выражений X и Y не равны?

Введите номер правильного ответа.

### Вопрос 30

В языке Пролог имеются следующие дополнительные средства управления выполнением программы:

1) предикат отсечения;

2) предикат fail;

3) предикат true;

4) предикат отрицания not;

5) предикат call;

6) предикат repeat.

Какие средства управления могут менять декларативную семантику программы и потенциально опасны?

Введите через пробел номера предикатов (в порядке возрастания), которые вы считаете ответами на вопрос.

**Вопрос 31**

Какие следующие предикаты относятся к "чистому" Прологу, т. е. описываются в рамках логической теории предикатов первого порядка?

- 1) отсечение;
- 2) fail;
- 3) true;
- 4) отрицание not;
- 5) call;
- 6) repeat;
- 7) findall;
- 8) assert;
- 9) retract.

Введите через пробел номера предикатов (в порядке возрастания), которые вы считаете ответами на вопрос.

**Вопрос 32**

Следующий текст взят из учебника по Прологу.

"Внесение изменений в программу при помощи предикатов ... может сделать поведение программы значительно менее понятным. В частности, одна и та же программа на одни и те же вопросы будет отвечать по-разному в разные моменты времени. В таких случаях, если мы захотим повторно воспроизвести исходное поведение программы, нам придется предварительно убедиться в том, что ее исходное состояние, нарушенное при обращении к предикатам ..., полностью восстановлено.

О каких предикатах из следующего списка идет речь?

- 1) отсечение;
- 2) fail;
- 3) true;
- 4) отрицание not;
- 5) call;
- 6) repeat;
- 7) findall;
- 8) assert;
- 9) retract.

Введите через пробел номера предикатов (в порядке возрастания), которые вы считаете ответами на вопрос.

**Вопрос 33**

Пролог. Последовательность чисел Фибоначчи имеет вид

1, 1, 2, 3, 5, 8, 13,...

Каждый член последовательности, за исключением первых двух, представляет собой сумму предыдущих двух членов. Какой метод программирования

позволяет написать предикат  $\text{fib}(N, F)$ , эффективно (т. е. линейно по времени) вычисляющий  $N$ -ое число Фибоначчи  $F$ ?

Варианты ответов:

- 1) использование запоминающих функций;
- 2) такого метода нет, рекурсия всегда не эффективна;
- 3) использование отсечений;
- 4) изменение порядка целей и предложений;
- 5) использование накапливающих параметров.

Введите через пробел номера ответов (в порядке возрастания), которые вы считаете правильными.

### Вопрос 34

Пролог. Пусть предикат умножения двух натуральных чисел  $p(+M, +N, ?R)$  определен следующим образом

$p(0, \_, 0).$

$p(\_, 0, 0).$

$p(X, 1, X).$

$p(X, N, R):- N>0, N1 \text{ is } N-1, p(X, N1, R1), R \text{ is } R1+X.$

Сколько ответов даст вызов

?-  $p(0, 2, X).$

- 1) один ответ  $X=0$ ;
- 2) один неправильный ответ;
- 3) два правильных ответа;
- 4) три правильных ответа;
- 5) программа заикнется;
- 6) более трех правильных ответов.

Введите номер правильного ответа.

### Вопрос 35

Пролог. Пусть предикат перемножения всех натуральных чисел от 1 до  $N$   $\text{factorial}(+N, ?R)$  определен следующим образом

$\text{factorial}(1, 1).$

$\text{factorial}(2, 2).$

$\text{factorial}(N, R):- N>1, N1 \text{ is } N-1, \text{factorial}(N1, R1), R \text{ is } R1*N.$

Сколько ответов даст вызов

?-  $\text{factorial}(3, X).$

- 1) один ответ  $X=6$ ;
- 2) один неправильный ответ;
- 3) два правильных ответа;
- 4) три правильных ответа;
- 5) программа заикнется;
- 6) более трех правильных ответов.

Введите номер правильного ответа.

**Вопрос 36**

Пролог. Какие из следующих утверждений правильные.

1. Отсечение отбрасывает все расположенные после него предложения. Если цель  $p$  унифицирована с предложением, содержащим отсечение, и отсечение выполнено, то эта цель не может быть использована для построения решений с помощью предложений, расположенных ниже данного.
2. Отсечение отбрасывает все альтернативные решения конъюнкции целей, расположенных в предложении левее отсечения, т. е. конъюнкция целей, стоящих перед отсечением, приводит не более чем к одному решению.
3. Отсечение влияет на цели, расположенные правее его. В случае возврата они не могут порождать другие решения.

Введите через пробел номера утверждений (в порядке возрастания), которые вы считаете правильными.

**Вопрос 37**

Пролог. Предикат для вычисления минимума двух чисел определен следующим образом

`minimum(X,Y,X):- X=<Y,!.`

`minimum(X,Y,Y):- X>Y,!.`

По какой причине поставлено отсечение после второго предложения?

Варианты ответов:

- 1) увеличивает эффективность программы;
- 2) излишне с любой прагматической точки зрения и поставлено только ради симметрии;
- 3) есть какая-то польза от этого отсечения, но я не знаю.

Введите номер вашего ответа.

**Вопрос 38**

Пролог. Какие из приведенных цепочек символов являются атомами:

- 1) человек(георгий)
- 2) 'человек(георгий)'
- 3) man(georg)
- 4) b
- 5) A
- 6) 7
- 7) very\_long\_string\_of\_symbol
- 8) a+b
- 9) ++

Введите через пробел номера термов (в порядке возрастания), которые вы считаете атомами.

**Вопрос 39**

Пролог. Определите, будет ли каждая из следующих пар термов унифицируемой:

- 1) 'книга'('название'('Ферма животных'), 'автор'('Джордж Оруэлл')) и 'книга'('название'(T), Avtor);
- 2) 'дата'('день недели'('среда'), 'число'(12), 'месяц'(M), 'год'(1986)) и 'дата'(W,D,X,Y);
- 3) 'праздник'('рождество', 'дата'('день'(25), 'месяц'('декабрь'), 'год'(Y))) и 'праздник'(H, 'дата'(D,M, 'год'(1986)));
- 4) 'праздник'('Первомай'(1, 'май')) и 'праздник'('Первомай', 1, 'май').

Введите через пробел номера пар (в порядке возрастания), для которых, вы считаете, проходит унификация.

**Вопрос 40**

Пролог. Какие следующие утверждения истинны?

1. В Прологе единственная структура данных - термы.
2. Структура характеризуется своим функтором. Не допускается использовать структуры с одинаковым функтором и разной местностью (арностью).
3. Факт - это структура, которая используется в программе для представления объекта или отношения в задаче из конкретной предметной области.
4. Мы задаем вопросы, используя термы в качестве целей.
5. Структура относится к рекурсивному типу данных.
6. Переменная - это "забронированное" место. Любой терм может заменить переменную, но для разных вхождений переменной в структуру не обязательно должна иметь место одна и та же замена.
7. Когда два терма унифицируются, переменные в них заменяются на некоторые значения.

Введите через пробел номера утверждений (в порядке возрастания), которые вы считаете правильными.

**Вопрос 41**

В каком порядке Пролог ищет утверждения программы для унификации с целью?

Варианты ответов:

- 1) в порядке размещения клауз (предложений) в тексте программы - сверху вниз;
- 2) сначала просматриваются факты в программе, потом - правила сверху вниз;
- 3) Пролог сам устанавливает порядок, исходя из эффективности программы.

Введите номер варианта.

**Вопрос 42**

Пролог. Какие следующие утверждения истинны?

1. Правило имеет заголовок (голову) и тело.
2. Унификация цели может вызвать необходимость унификации подцелей.
3. Пролог всегда совершает возврат для повторной унификации любой цели.
4. Пытаясь согласовать цель повторно, Пролог начинает поиск снова с начала программы.
5. Пользователь может заставить Пролог совершить возврат, отвергнув полученный ответ.

Введите через пробел номера утверждений (в порядке возрастания), которые вы считаете правильными.

### Вопрос 43

Пролог. Установите, какие из представленных пар термов унифицируются:

- 1) [['серый','зеленый'],'черный','голубой'] и [H|T];
- 2) [['георгий','мария']] и [H|[]];
- 3) [['уильям','мери']|Any] и [Fist,Second] ;
- 4) ['дом','осел','лошадь'] и [H,T];
- 5) [['брак'('георгий','мария')]] и [A];
- 6) [[1805],1815] и [[A|B],C|D];
- 7) ['жак','жиль'] и [A,B|C].

Введите через пробел номера пар (в порядке возрастания), для которых, вы считаете, проходит унификация.

### Вопрос 44

Пролог. К чему приведет следующий вызов предиката

?- max(4+7, 8\*9, N).

если предикат max/3 определен следующим образом:

max(X,Y,X):-X>=Y.

max(X,Y,Y):-Y>X.

Варианты ответов:

- 1) Пролог ответит Yes и выдаст N=72;
- 2) Пролог ответит Yes и выдаст N=8\*9;
- 3) Пролог ответит No;
- 4) Пролог сообщит об ошибке в операциях сравнения.

Введите номер ответа.

### Вопрос 45

Пролог. Для каждого варианта вызова выберите свой результат.

Вызовы:

- 1) X is 7+2;
- 2) X =:= 7+2;
- 3) X = 72;
- 4) 8+1 is 7+2;
- 5) 8+1 =:= 7+2;

6)  $8+1 = 7+2$ .

Результаты:

- 1) отказ: термы не сопоставимы;
- 2) унифицируется: подстановка  $X \leftarrow 9$ ;
- 3) унифицируется: значения выражений одинаковы;
- 4) ошибка Пролога:  $X$  - не арифметическое выражение;
- 5) отказ: термы не сопоставимы;
- 6) унифицируется: подстановка  $X \leftarrow \dots$ .

Введите через пробел номера результатов, соответствующих вызовам в порядке 1), 2),...6).

**Вопрос 46**

Пролог. Для каждого списка в стандартном синтаксисе, использующем символы '!', '[' и ']':

- 1)  $\cdot(x, \cdot(y, \cdot(z, [])))$ ;
- 2)  $\cdot(x(y), [])$ ;
- 3)  $\cdot([x, y], \cdot([z, y], []))$ ;
- 4)  $\cdot([x, y], \cdot(z, \cdot([], \cdot(x, []))))$ ,

выберете вариант в сокращенном синтаксисе:

- 1)  $[x(y)]$ ;
- 2)  $[[x, y], z, [], x]$ ;
- 3)  $[x, y, z]$ ;
- 4)  $[[x, y], [z, y]]$ .

Введите через пробел номера представлений в сокращенном синтаксисе, соответствующих спискам в первом представлении в порядке 1), 2), 3), 4). Если для какого-то списка из первого представления отсутствует правильный вариант в сокращенном представлении, введите 0.

**Вопрос 47**

Пролог. Имеются три варианта определения предиката max:

- 1)  
 $\text{max}(N1, N2, N2):- N2 \geq N1.$   
 $\text{max}(N1, N2, N1):- N2 < N1.$
- 2)  
 $\text{max}(N1, N2, N2):- N2 \geq N1, !.$   
 $\text{max}(N1, N2, N1).$
- 3)  
 $\text{max}(N1, N2, N3):- N2 \geq N1, !, N3 = N2.$   
 $\text{max}(N1, N2, N1).$

Какие ответы даст Пролог в этих вариантах на вызов?

?- max(3, 7, 3).

Введите через пробел три соответствующих ответа (каждый ответ - слова yes или no).

**Вопрос 48**

Имеется программа

s(1).

s(2).

p(X):-retract(X), fail.

p(\_).

Какое будет значение у X при вызове

?- p(s(X)).

Варианты ответов:

1) 1;

2) 2;

3) X будет унифицирована с внутренней переменной;

4) будет получен отказ - No.

Введите номер правильного ответа.

**Вопрос 49**

Пролог. Даны следующие утверждения:

1) каждое небесное тело, заслуживающее внимание, представляет собой либо звезду, либо планету, либо комету;

2) у комет, расположенных недалеко от Солнца, есть хвосты;

3) Венера недалеко от Солнца, но у нее нет хвоста.

Выразим их на языке хорновских клауз:

1)

'небесное тело'(X):- 'заслуживает внимание'(X),  
( 'звезда'(X); 'планета'(X); 'комета'(X)).

2)

'есть хвост'(X):- 'недалеко от Солнца'(X), 'комета'(X).

3)

'недалеко от Солнца'('Венера').

Какие переводы на языке хорновских дизъюнктов правильны? Помните, что в Прологе мир замкнут.

Введите через пробел номера правильных переводов (в порядке возрастания).

**Вопрос 50**

Пролог. Даны следующие утверждения:

1) ни один дракон, который живет в зоопарке, не является счастливым;

2) любой зверь, который встречает добрых людей, является счастливым;

3) люди, которые посещают зоопарк, - добрые;

4) звери, которые живут в зоопарке, встречают людей, которые посещают зоопарк.

Выразим их на языке хорновских клауз:

1)

'счастливый'(X):- 'живет в зоопарке'(X), not 'дракон'(X).

2)

'счастливый'(X):- 'зверь'(X), 'встречает человека'(X,Y), 'добрый человек'(Y).

3)

'добрый человек'(X):- 'посещает зоопарк'(X).

4)

'встречает человека'(X,Y):- 'посещает зоопарк'(Y), 'зверь'(X),  
'живет в зоопарке'(X).

Какие переводы на языке хорновских дизъюнктов правильны?

Введите через пробел номера правильных переводов (в порядке возрастания).

### Вопрос 51

Пролог. Даны следующие утверждения:

- 1) каждому кто-то нравится;
- 2) каждому нравится каждый;
- 3) кому-то нравится каждый;
- 4) никому не нравятся все;
- 5) никому не нравится некто;
- 6) кому-то не нравится никто.

Используя предикат 'нравится'(x,y) - "иксу нравится игрек", дан перевод этих утверждений на язык логики предикатов (не обязательно в этом порядке):

- 1)  $\exists x \forall y \neg \text{'нравится'}(y, x)$ ;
- 2)  $\exists y \forall x \neg \text{'нравится'}(y, x)$ ;
- 3)  $\forall x \exists y \text{'нравится'}(x, y)$ ;
- 4)  $\neg \exists x \forall y \text{'нравится'}(x, y)$ ;
- 5)  $\exists x \forall y \text{'нравится'}(x, y)$ ;
- 6)  $\forall x \forall y \text{'нравится'}(x, y)$ .

Введите через пробел номера переводов, соответствующих утверждениям в порядке 1), 2),...6).

### Вопрос 52

Лисп. Дано определение функции, вычисляющей полную длину списка, т. е. суммарную длину списка и всех его подсписков:

```
(defun fulllength(s)
  (if (atom s) 1
      (+ (fulllength (car s)) (fulllength (cdr s)))))
```

Чему будет равно значение (fulllength '(1)).

Введите ответ.

### Вопрос 53

Лисп. Функция ident определена следующим образом:

```
(defun ident (x) x)
```

Какие из приведенных ниже выражений являются правильными (т. е. Лисп не сообщит об ошибке):

- 1) (ident (quote a));
- 2) (quote (ident a));
- 3) (length a b);
- 4) (length (quote (a b)));
- 5) (quote (length a b));
- 6) (length (ident a b));
- 7) (ident (quote (a b)));
- 8) (quote (ident (a b)));
- 9) (length (ident (quote (a b))));
- 10) (length (quote (ident (a b)))).

Введите через пробел номера правильных выражений (в порядке возрастания).

#### Вопрос 54

Лисп. Какая ошибка в определении функции, проверяющей, является ли данный список одноуровневым?

```
(defun f(s)
  (if (not (atom (car s))) nil (f (cdr s))))
```

Варианты ответов:

- 1) без ошибок;
- 2) перепутаны случаи "то" и "иначе" в условной функции;
- 3) надо пользоваться предикатом `or`;
- 4) нет окончания рекурсии.

Введите номер ответа.

#### Вопрос 55

Лисп. Следующая функция удаляет n-ый элемент из списка s.

```
(defun f(n s)
  (cond ((null s) nil)
        ((= n 1) (rest s))
        (t (cons (first s) (1- (rest s))))))
```

Чему равно значение (length (f 3 '(x y)))?

Введите ответ (если вы считаете, что Лисп выдаст ошибку, введите `error`).

#### Вопрос 56

Лисп. Даны три различные функции с аргументами s - список и x - символьное выражение, выполняющие следующие действия:

- 1) удаляет первое вхождение x из списка s только на верхнем уровне;
- 2) удаляет все вхождения x из списка s;
- 3) удаляет все вхождения x из одноуровневого списка s;

Следующие три определения реализуют эти функции (необязательно в том же порядке):

1)

```
(defun f (s x)
  (cond ((null s) nil)
        ((equal x (first s)) (f (rest s) x))
        ((listp (first s)) (cons (f (first s) x) (f (rest s) x)))
        (t (cons (first s) (f (rest s) x)))))
```

2)

```
(defun f (s x)
  (cond ((null s) nil)
        ((equal x (first s)) (rest s))
        (t (cons (first s) (f (rest s) x)))))
```

3)

```
(defun f (s x)
  (cond ((null s) nil)
        ((equal x (first s)) (f (rest s) x))
        (t (cons (first s) (f (rest s) x)))))
```

Введите через пробел номера определений, соответствующих действиям в порядке 1), 2), 3).

### Вопрос 57

Лисп. Сколько элементов самого верхнего уровня в следующих списках:

- 1) ((1 2 3));
- 2) ((a b) c (d (e)));
- 3) (a ((( ))) nil nil);
- 4) (((((a (b (c d) e) f) g) h ((i (j) k) l) m) n)

Введите последовательность ответов через пробел (в порядке нумерации).

### Вопрос 58

Лисп. Определите значения следующих выражений:

- 1) (length '(+ 2 (\* 3 4)));
- 2) (length (+ 2 '(\* 3 4)));
- 3) (length (+ 2 (\* 3 4)));
- 4) (length (+ 2 (\* 3 '4)));
- 5) (length (quote 'quote));
- 6) (length (quote 2));
- 7) (length '(quote nil)).

Введите последовательность ответов через пробел (в порядке нумерации).  
Возможные ответы: число или слово error (ситуация ошибки).

### Вопрос 59

Лисп. Может ли функция принимать разные значения на одних и тех же аргументах?

Варианты ответов:

- 1) нет, если "чистая" функция;
- 2) да, если есть побочные эффекты у других функций;
- 3) да, если используются глобальные переменные.

Введите последовательность номеров правильных ответов через пробел (в порядке нумерации).

### Вопрос 60

Лисп. Какие из следующих вызовов возвращают значение t?

- 1) (atom '(cdr nil));
- 2) (equal '(a b) (cons '(a) '(b)));
- 3) (atom (\* 2 (+ 2 3)));
- 4) (null (null t));
- 5) (eq nil (null nil))
- 6) (eq 2.0 2)
- 7) (equal 2.0 2)
- 8) (= 2.0 2)
- 9) (equal (atom nil) (caar '((t))))

Введите последовательность номеров правильных ответов через пробел (в порядке нумерации).

### Вопрос 61

Лисп. Дана последовательность выражений:

- 1) (length (cons nil '(nil)));
- 2) (length (cons nil nil));
- 3) (length (cons '(nil) '(nil)));
- 4) (length (cons (car '(a b)) (cdr '(a b))));
- 5) (length (car '(car (a b c))));
- 6) (length (cdr (car (cdr '(a b c))))).

Введите последовательность ответов через пробел (в порядке нумерации).  
Возможные ответы: число или слово error (ситуация ошибки).

### Вопрос 62

Лисп. Вычислите значения следующих выражений:

- 1) (length ((lambda (x) (cons x nil)) 'y));
- 2) (length ((lambda (x y) (list y x)) 'x y));
- 3) (length ((lambda (x) (list x)) (list nil)));
- 4) (length ((lambda (x) (list x)) ((lambda (x) (list x)) (list nil)))).

Введите последовательность ответов через пробел (в порядке нумерации).  
Возможные ответы: число или слово error (ситуация ошибки).

### Вопрос 63

Лисп. Вычислите значения следующих выражений:

- 1) (length ((lambda (x y) (list x y)) '(+ 2 3) 'c));

- 2) (length ((lambda (x y) ((lambda (z) (list x y z)) 'c)));
- 3) (length ((lambda (x y) (list x y))  
((lambda (z) z) 'a) 'b)).

Введите последовательность ответов через пробел (в порядке нумерации).  
Возможные ответы: число или слово error (ситуация ошибки).

#### Вопрос 64

Лисп. Вычислите значения следующих вызовов:

- 1) (length (apply 'list '(a b)));
- 2) (length (funcall 'list '(a b)));
- 3) (length (funcall 'apply 'list '(a,b)));
- 4) (length (funcall 'list 'apply '(a b))).

Введите последовательность ответов через пробел (в порядке нумерации).  
Возможные ответы: число или слово error (ситуация ошибки).

#### Вопрос 65

Лисп. Какова неподвижная точка функции (lambda (x) (+ 1 (/ x 2)))?

Введите значение.

#### Вопрос 66

Следующие предложения на Лиспе должны следовать друг за другом. В каждом случае укажите, имеет ли результат какой-либо смысл. (Например, если за (setq x 'y) следует (setq z (+ 6 x)), то второе из этих предложений - очевидная бессмыслица, потому что оно пытается сложить 6 с символом y.)

- 1) (setq a (+ 3 7))
- 2) (setq a (+ a 7))
- 3) (set a (+ a 7))
- 4) (setq b 'alpha)
- 5) (set b 'beta)
- 6) (setq c (+ 6 alpha))
- 7) (setq c (+ 6 'alpha)).

Введите последовательность номеров через пробел тех выражений, которые имеют смысл (в порядке возрастания).

#### Вопрос 67

Лисп. Имеется два определения символьного выражения.

1. Символьное выражение составляется из атомов (которые могут быть константами или переменными) и скобок таким образом, что если мы представим себе счетчик, установленный вначале в нуль, и будем увеличивать его на единицу при движении слева направо каждый раз, когда встречается левая скобка, и уменьшать на единицу каждый раз, когда встречается правая скобка, то его конечное значение будет нуль. Атомы между собой должны разделяться пробелами.

2. Символьное выражение - это либо атом, либо левая круглая скобка, за которой следует последовательность символьных выражений, отделенных друг от друга пробелами, и следующая за нею правая круглая скобка.

Правильны ли эти определения?

Варианты ответов:

- 1) оба правильны;
- 2) оба неправильны;
- 3) первое - правильное, второе - неправильное;
- 4) второе - правильное, первое - неправильное.

Введите номер правильного ответа.

### Вопрос 68

Лисп. Каково общее число подписков в подписках (т. е. списки уровня 3; исходный список имеет уровень 1) в следующих списках:

- 1) (6 (3 6 (7 (4 5) 8)) (9 3));
- 2) (((a 9( (b 7) c) (d 5)));
- 3) (quote (s (3 5) (7 (9 8))));
- 4) (d (q 2 7 4) (+ 1 (\* 7 (+ 3 (- 2)))))?

Введите последовательность чисел через пробелы в порядке перечисления исходных списков.

### Вопрос 69

Лисп. Дан лямбда-терм

$(\lambda h. (\lambda x. h(x\ x))(\lambda x. h(x\ x))) ((\lambda x. x) (+\ 1\ 5))$

и некоторые его подтермы:

- 1)  $\lambda x. h(x\ x)$ ;
- 2)  $\lambda h. (\lambda x. h(x\ x))(\lambda x. h(x\ x))$ ;
- 3)  $(\lambda x. h(x\ x))(\lambda x. h(x\ x))$ ;
- 4)  $(\lambda h. (\lambda x. h(x\ x))(\lambda x. h(x\ x))) ((\lambda x. x) (+\ 1\ 5))$ ;
- 5)  $((\lambda x. x) (+\ 1\ 5))$ ;
- 6)  $(\lambda x. x)$ ;
- 7)  $(+\ 1\ 5)$ .

Какие из этих подтермов являются редексами?

Введите номера соответствующих подтермов через пробел (в порядке возрастания).

### Вопрос 70

Лисп. Дан лямбда-терм

$(\lambda x. \lambda y. x (\lambda z. y\ z))(((\lambda x. \lambda y. y)\ 8)(\lambda x. (\lambda y. y)\ x))$

и некоторые его подтермы:

- 1)  $x (\lambda z. y\ z)$ ;
- 2)  $\lambda x. \lambda y. x (\lambda z. y\ z)$ ;

- 3)  $\lambda x. \lambda y. y$ ;
- 4)  $(\lambda x. \lambda y. y) 8$ ;
- 5)  $(\lambda y. y) x$ ;
- 6)  $\lambda x. (\lambda y. y) x$ ;
- 7)  $(\lambda x. \lambda y. x (\lambda z. y z))(((\lambda x. \lambda y. y) 8)(\lambda x. (\lambda y. y) x))$ ;
- 8)  $((\lambda x. \lambda y. y) 8)(\lambda x. (\lambda y. y) x)$ .

Какие из этих подтермов являются редексами?

Введите номера соответствующих подтермов через пробел (в порядке возрастания).

### Вопрос 71

Лисп. Дан лямбда-терм

$(\lambda h. (\lambda x. h(x x))(\lambda x. h(x x))) ((\lambda x. x) (+ 1 5))$

и некоторые его подтермы:

- 1)  $\lambda x. h(x x)$ ;
- 2)  $\lambda h. (\lambda x. h(x x))(\lambda x. h(x x))$ ;
- 3)  $(\lambda x. h(x x))(\lambda x. h(x x))$ ;
- 4)  $(\lambda h. (\lambda x. h(x x))(\lambda x. h(x x))) ((\lambda x. x) (+ 1 5))$ ;
- 5)  $((\lambda x. x) (+ 1 5))$ ;
- 6)  $((\lambda x. x))$ ;
- 7)  $(+ 1 5)$ .

Какой из этих подтермов является самым левым из самых внешних редексов и самым левым из самых внутренних редексов?

Введите номера соответствующих подтермов через пробел (сначала номер самого левого из самых внешних редексов, потом номер самого левого из самых внутренних).

### Вопрос 72

Лисп. Дан лямбда-терм

$(\lambda x. \lambda y. x (\lambda z. y z))(((\lambda x. \lambda y. y) 8)(\lambda x. (\lambda y. y) x))$

и некоторые его подтермы:

- 1)  $x (\lambda z. y z)$ ;
- 2)  $\lambda x. \lambda y. x (\lambda z. y z)$ ;
- 3)  $\lambda x. \lambda y. y$ ;
- 4)  $(\lambda x. \lambda y. y) 8$ ;
- 5)  $(\lambda y. y) x$ ;
- 6)  $\lambda x. (\lambda y. y) x$ ;
- 7)  $(\lambda x. \lambda y. x (\lambda z. y z))(((\lambda x. \lambda y. y) 8)(\lambda x. (\lambda y. y) x))$ ;
- 8)  $((\lambda x. \lambda y. y) 8)(\lambda x. (\lambda y. y) x)$ .

Какой из этих подтермов является самым левым из самых внешних редексов и самым левым из самых внутренних редексов?

Введите номера соответствующих подтермов через пробел (сначала номер самого левого из самых внешних редексов, потом номер самого левого из самых внутренних).

### Вопрос 73

Лисп. Какие из следующих утверждений верны?

1.  $\lambda$ -исчисление - это исчисление безымянных функций. Оно включает в себя нотацию для записи выражений и набор правил преобразований этих выражений.
2.  $\lambda$ -исчисление может быть дополнено произвольным набором констант, таких, как целые числа, и связанных с ними функций, называемых  $\delta$ -правилами.
3. Когда есть возможность выбрать редекс для преобразования, выбор определяется порядком редукций. Двумя альтернативными стратегиями выбора являются аппликативный порядок редукций и нормальный порядок редукций, тесно связанные с ленивым вычислением и энергичным вычислением, соответственно.
4. Удалив  $\delta$ -правила, мы получим чистое  $\lambda$ -исчисление. В нем нельзя выразить некоторые функции.

Введите номера правильных утверждений через пробел (в порядке возрастания).

### Вопрос 74

Лисп. Что является наименьшей неподвижной точкой выражения  $\lambda x. * x x$  ?

Введите число.

### Вопрос 75

Лисп. Какие из следующих утверждений верны?

1. Язык XLisp - функциональный язык с ленивыми вычислениями.
2. Язык XLisp - функциональный язык с энергичными вычислениями.
3. Язык XLisp - функциональный язык с энергичными и ленивыми вычислениями.
4. Язык C++ - функциональный язык.

Введите номера правильных утверждений через пробел (в порядке возрастания).

### Вопрос 76

Лисп. Пусть  $M \equiv \lambda x. ((\lambda y. y y) x)$ .

Различные цепочки редукций выражения  $M M$  могут привести только к четырем различным термам. Найдите в следующем списке терм, к которому мы не можем редуцировать  $M M$  :

- 1)  $M M$ ;
- 2)  $\lambda x. M$ ;

- 3)  $(\lambda x. x x) M$ ;
- 4)  $M (\lambda x. x x)$ ;
- 5)  $(\lambda x. x x) (\lambda x. x x)$ .

Введите номер терма.

### Вопрос 77

Лисп. Пусть  $M \equiv \lambda x. \lambda y. x y y$ . Различные цепочки редукций выражения  $W W W$  могут привести только к трем различным термам. Найдите в следующем списке терм, к которому мы не можем редуцировать  $W W W$  :

- 1)  $W W W$ ;
- 2)  $(\lambda y. W y y)W$ ;
- 3)  $(\lambda y. y y y) W$ ;
- 4)  $(\lambda y. y y W) W$ .

Введите номер терма.

### Вопрос 78

Лисп. Пусть  $IF \equiv \lambda p. \lambda q. \lambda r. p q r$ ,  $FALSE \equiv \lambda x. \lambda y. y$ . Сколько шагов редукции требуется для преобразования  $IF FALSE A B$  к нормальной форме?

Введите число.

### Вопрос 79

Лисп. Пусть  $IF \equiv \lambda p. \lambda q. \lambda r. p q r$ ,  $TRUE \equiv \lambda x. \lambda y. x$ . Сколько шагов редукции требуется для преобразования  $IF TRUE A B$  к нормальной форме?

Введите число.

### Вопрос 80

Лисп. Пусть  $FALSE \equiv \lambda x. \lambda y. y$ ,  $TRUE \equiv \lambda x. \lambda y. x$  и  $OR \equiv \lambda x. \lambda y. (x TRUE) y$ . Сколько шагов редукции требуется для преобразования  $TRUE OR FALSE$  к нормальной форме?

Введите число.

### Вопрос 81

Лисп. Пусть  $FALSE \equiv \lambda x. \lambda y. y$ ,  $TRUE \equiv \lambda x. \lambda y. x$  и  $AND \equiv \lambda x. \lambda y. x y FALSE$ . Сколько шагов редукции требуется для преобразования  $TRUE AND FALSE$  к нормальной форме?

Введите число.

### Вопрос 82

Лисп. Пусть  $CONS \equiv \lambda h. \lambda t. \lambda s. s h t$  и  $REST \equiv \lambda x. x FALSE$ . Сколько шагов редукции требуется для преобразования  $REST (CONS A B)$  к нормальной форме?

Введите число.

**Вопрос 83**

Лисп. Дано определение функции

```
(defun rol (f g)
  (function (lambda (x y)
    (funcall f (funcall g x y) (funcall g y x))))))
```

Чему равно значение (funcall (rol '\* '+) 2 3) ?

**Вопрос 84**

Лисп. Даны определения функций

```
(defun twice (f)
  (function (lambda (x) (funcall f (funcall f x)))))
(defun do (x) (funcall (twice 'list) x))
```

Чему равно значение (do '0) ?

**Вопрос 85**

Лисп. Дано определения функций

```
(defun many (f x)
  (mapcar #'(lambda (g) (funcall g x)) f))
(defun f1 (x) (+ x x))
(defun f2 (x) (* x x))
```

Чему равно значение (length (many '(f1 f2) 1)) ?

**Вопрос 86**

Лисп. Дано определение функции

```
(defun f (p s)
  (cond ((null s) t)
        ((funcall p (car s)) (f p (cdr s)))
        (t nil)))
```

Чему равно значение (f 'zerop '(0 0 0 0)) ?

**Вопрос 87**

Лисп. Дано определение функции

```
(defun f (p s)
  (cond ((null s) nil)
        ((funcall p (car s)) t)
        (t (f p (cdr s)))))
```

Чему равно значение (f 'zerop '(1 1 1)) ?

**Вопрос 88**

Лисп. Дано определение функции

```
(defun create (x y)
  (eval (cons 'defun (cons x (cdr y)))))
```

Вызов этой функции приводит к определению некоторой новой функции f.

```
(create 'f '(lambda (x) (* x x)))
```

Чему равно значение (f 2)?

**Вопрос 89**

Лисп. Дано определение функции

```
(defun create (x y)
  (eval (cons 'defun (cons x (cdr y)))))
```

Вызов этой функции приводит к определению некоторой новой функции f.

```
(create 'f '(lambda (x) (+ x x)))
```

Чему равно значение (f 5)?

**Вопрос 90**

Лисп. Дано определение функции

```
(defun create (x y)
  (eval (cons 'defun (cons x (cdr y)))))
```

Вызов этой функции приводит к определению некоторой новой функции f.

```
(create 'f '(lambda (x) (+ x 1)))
```

Чему равно значение (f 2)?

**Вопрос 91**

Лисп. Запишите с помощью функций C...R последовательность вызовов CAR и CDR, выделяющих из приведенных ниже списков символ goal.

1) (1 2 goal 3 4)

2) ((1) (2 goal) (3 (4)))

3) ((1 (2 (3 4 goal))))

Введите последовательность ответов - имен функций через пробел (в порядке нумерации).

**Вопрос 92**

Лисп. Дано определение функции

```
(defun factor (n)
  (if (< n 2) '(1) (append (factor (- n 1)) (list '* n))))
```

Чему равно значение (length (factor 3)) ?

**Вопрос 93**

Лисп. Дано определение функции

```
(defun mix (x y)
  (if (null x) y (cons (car x) (mix y (cdr x)))))
```

Чему равно значение (length (mix '(1 2) '(a b))) ?

#### Вопрос 94

Лисп. Дано определение функции

```
(defun f (s)
  (cond ((null s) nil)
        ((null (cdr s)) s)
        (t (cons (car s) (f (cddr s))))))
```

Чему равно значение (length (f '(a b))) ?

#### Вопрос 95

Лисп. Дано определение функции

```
(defun f (x s)
  (if (= x (car s)) 1 (+ 1 (f x (cdr s)))))
```

Чему равно значение (f 2 '(1 3 2)) ?

#### Вопрос 96

Лисп. Какая из функций, чье имя начинается с C, кончается R и содержит буквы A и D, (например CADDDR) будет:

- 1) давать (3 4), когда применяется к (1 2 (3 4));
- 2) давать (3 4), когда применяется к (1 2 3 4);
- 3) давать (6 3), когда применяется к (((4 (6 3)) 8) 7);
- 4) давать 12, когда применяется к (5 ((12) 23 34));
- 5) давать (12), когда применяется к (5 ((12) 23 34))?

Введите последовательность функций через пробел (в порядке нумерации).

#### Вопрос 97

Лисп. Каково значение каждого из следующих выражений:

- 1) (atom (car (quote ((1 2) 3 4))));
- 2) (null (cddr (quote ((5 6) (7 8)))));
- 3) (equal (car (quote ((7)))) (cdr (quote (5 7))));
- 4) (zerop (cadddr (quote (3 2 1 0)))) ?

Введите последовательность значений через пробел (в порядке нумерации).

#### Вопрос 98

Лисп. Дано определение функции

```
(defun f(s)
  (if (null s) nil (cons (car s) (f (f (cdr s))))))
```

Чему равно значение (f '(1 2 3)) ?

**Вопрос 99**

Лисп. Даны две функции

(defun g (x) (atom (car x))) и (defun f (s) (> (cadr s) (caddr s))).

Чему равно значение (length (append (g '((1) 2)) (f '(1 2 3 4 5)))) ?

**Вопрос 100**

Лисп. Какое значение каждого из следующих выражений:

1) (eval (list (quote -) (+ 3 4)));

2) (eval (list (car (quote (atom x))) 6));

3) (eval (list (quote equal) (+ 2 2) (\* 2 2))) ?

Введите последовательность значений через пробел (в порядке нумерации).

**Вопрос 101**

Лисп. Каково значение следующего выражения

(eval (cons (quote >) (cons 5 (list ((lambda (x y) (- x y)) 3 7))))) ?

Введите значение.

## 7. РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

### Основная литература по логическому программированию

1. Братко И. Программирование на языке Пролог для искусственного интеллекта. - М.: Мир, 1990. - 560 с.
2. Стерлинг Л., Шапиро Э. Искусство программирования на языке Пролог. - М.: Мир, 1990. - 235 с.

### Дополнительная литература по логическому программированию

1. Малпас Дж. Реляционный язык Пролог и его применение. - М.: Наука, 1990. - 464 с.
2. Логический подход к искусственному интеллекту: от классической логики к логическому программированию: Пер.с франц./Тейз А., Грибомон П., Луи Ж. и др. - М.: Мир, 1990. - 432 с.

### Основная литература по функциональному программированию

1. Хювенен Э., Сеппянен Й. Мир Лиспа. В 2-х т. - М.: Мир, 1990.
2. Хендерсон П., Функциональное программирование. Применение и реализация. - М.: Мир, 1983. - 349 с.

### Дополнительная литература по функциональному программированию

1. Филд А., Харрисон П. Функциональное программирование. - М.: Мир, 1993. - 637 с.
2. Бердж В. Методы рекурсивного программирования. - М.: Машиностроение, 1983. - 248 с.
3. Барендрегт Х. Ламбда - исчисление. Его синтаксис и семантика. - М.: Мир, 1985. - 606 с.

**ПРИЛОЖЕНИЕ 1**

**ФОРМА ТИТУЛЬНОГО ЛИСТА К КУРСОВОЙ РАБОТЕ**

Министерство образования Российской Федерации

**ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ  
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)**

Кафедра автоматизированных систем управления (АСУ)

**Нахождение компонент связности графа**

Пояснительная записка к курсовой работе по дисциплине  
"Логическое программирование"

Студент гр. 436-1  
С. С. Лавров  
20.12.2000 г.

**ПРИЛОЖЕНИЕ 2****ФОРМА ЗАДАНИЯ ДЛЯ КУРСОВОЙ РАБОТЫ**

Министерство образования Российской Федерации

**ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ  
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)**

Кафедра автоматизированных систем управления (АСУ)

УТВЕРЖДАЮ  
зав. кафедрой АСУ  
А. М. Кориков  
01. 06. 2000 г.

**ЗАДАНИЕ**

по курсовому проектированию по дисциплине  
"Логическое программирование"

студенту \_\_\_\_\_  
группа \_\_\_\_\_ факультет ФСУ

1. Тема проекта: Нахождение компонент связности графа
2. Срок сдачи студентом законченной работы 25.12.2000 г.
3. Исходные данные к проекту (здесь должен быть текст задания)
4. Дата выдачи задания: 01.6.2000 г.

Задание принял к исполнению  
(ФИО)

**ПРИЛОЖЕНИЕ 3****ПРИМЕР ОФОРМЛЕНИЯ СОДЕРЖАНИЯ****СОДЕРЖАНИЕ**

1. Введение	5
2. Анализ задачи	8
3. Решение задачи	10
3.1. Выбор алгоритма и структур данных	10
3.2. Описание алгоритма	14
3.3. Выбор набора тестов	18
4. Заключение	25
Список литературы	26
Приложение 1. Листинг программы	27
Приложение 2. Распечатки тестов	29

**ПРИЛОЖЕНИЕ 4****ПРИМЕРЫ БИБЛИОГРАФИЧЕСКИХ ОПИСАНИЙ ИСТОЧНИКОВ,  
ПОМЕЩАЕМЫХ В СПИСОК ЛИТЕРАТУРЫ**

1. Хендерсон П. Функциональное программирование. Применение и реализация. - М.: Мир, 1983. - 349 с.
2. Филд А., Харрисон П. Функциональное программирование. - М.: Мир, 1993. - 637 с.
3. Шеховцов А. С. Квазисинхронное регулирование гистерезисных электродвигателей // Тез. докл. на науч.-техн. конф. 21-23 дек. 1998 г. - Т. 4 - Томск: Издательство Томского государственного педагогического университета, 1999. - 133 с.
4. Калянов Г. Н. CASE-технологии проектирования программного обеспечения // Кибернетика и системный анализ. - 1993. - №5. - С. 152-164.