|  |
| --- |
| МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ |
| федеральное государственное автономное образовательное учреждение  высшего образования |
| **«Национальный исследовательский ядерный университет «МИФИ»** |
| ИНСТИТУТ ИНТЕЛЛЕКТУАЛЬНЫХ КИБЕРНЕТИЧЕСКИХ СИСТЕМ |
| КАФЕДРА КИБЕРНЕТИКИ (№ 22) |

«УтверждЕН

на заседании кафедры

«\_\_\_»\_\_\_\_\_\_\_\_\_\_20\_\_ г.,

протокол №\_\_\_\_\_\_\_\_\_

зав.каф.22

\_\_\_\_\_\_\_\_\_\_ / А.М.Загребаев /

**ФОНД ОЦЕНОЧНЫХ СРЕДСТВ**

по дисциплине

**«Логическое программирование»**

|  |  |
| --- | --- |
| Направление подготовки (специальность) | 09.03.04 Программная инженерия |
|  |  |
| Профиль подготовки |  |
|  |  |
| Наименование образовательной программы | Математическое и программное обеспечение вычислительных машин и компьютерных сетей |
|  |  |
| Квалификация (степень) выпускника | Бакалавр |
|  |  |
| Форма обучения | очная |

г. Москва, 2019 г.

**ПАСПОРТ**

**фонда оценочных средств по дисциплине**

**«Логическое программирование»**

(наименование дисциплины)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Семестр** | **Интерактив** | **Трудоемкость, кред.** | **Общий объем курса, час.** | **Лекции, час.** | **Практич. занятия, час.** | **Лаборат. работы, час.** | **СРС, час.** | **КСР, час.** | **Форма(ы) контроля, экз./зач./КР/КП** |
| 8 |  | 2 | 72 | 10 | 20 | 0 | 42 | 0 | З |
| ИТОГО | 0 | 2 | 72 | 10 | 20 | 0 | 42 | 0 |  |

Группа: Б19-504, Б19-514

**Модели контролируемых компетенций**

В результате освоения дисциплины у выпускника формируются следующие компетенции:

ОПК-1 – Способен применять естественнонаучные и общеинженерные знания, методы математического анализа и моделирования, теоретического и экспериментального исследования в профессиональной деятельности

ОПК-3 – Способен решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности

ПК-3 – владением навыками использования различных технологий разработки программного обеспечения

УК-1 – Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач

**Программа оценивания контролируемых компетенций**

Формирование у студентов компетенций контролируется в течение всего времени освоения дисциплины в рамках:

* текущего контроля;
* рубежного контроля;
* промежуточного контроля.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№ п.п.** | **Наименование раздела учебного курса** | **Не- дели** | **Лекции час.** | **Практ. зан./ семи-нары, час.** | **Лаб. рабо-ты, час.** | **Обязат. текущий контроль (форма\*, неделя)** | **Аттеста-ция раздела (форма\*, неделя)** | **Макси-мальный балл за раздел \*\*** | **Компетенции по разделам, проверяемые при текущем и рубежном контроле** | **Компетенции, проверяемые на зач. /экз.** |
| 8 семестр | | | | | | | | | | |
| 1 | История создания ЛП и языка Пролог; математические основы ЛП и Пролога. Основы Пролога. Методы и технология программирования на Прологе | 1-5 | 5 | 10 |  | ДЗ-5 | КИ, 5 | 25 | УК-1,  ОПК-1, ОПК-3,  ПК-3 |  |
| 2 | Применение Пролога для синтаксического анализа формальных языков. Пролог и базы данных. | 6-10 | 5 | 10 |  | ДЗ-10 | КИ, 10 | 25 | УК-1,  ОПК-1, ОПК-3,  ПК-3 |  |
|  | Зачет |  |  |  |  |  | З | 50 |  | УК-1,  ОПК-1, ОПК-3,  ПК-3 |
|  | Итого за 8 семестр |  |  |  |  |  |  | 100 |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Недели** | **Содержание / Темы занятий** | **Лек., час.** | **Пр./сем., час.** | **Лаб., час.** | **Компетенции по темам, проверяемые при текущем контроле** | **Виды тек.контроля по проверке компетенций** | **Компетенции по темам, проверяемые на зач. /экз.** |
| 1 семестр | | | | | | | |
| 1 - 3 | **Математические основы логического программирования.** Парадигма логического программирования. История вопроса об автоматизации логического вывода и представления знаний с использованием логики предикатов. Автоматическое доказательство теорем методом резолюций. Стратегия линейной резолюции. | 3 | 6 |  | УК-1,  ОПК-1, ОПК-3,  ПК-3 | КТР4 | УК-1,  ОПК-1,  ОПК-3,  ПК-3 |
| 4 - 5 | **Основы программирования на Прологе.** Синтаксис и семантика Пролога. Основные механизмы Пролога как языка логического программирования. Основные приёмы программирования на Прологе и встроенные средства современных Пролог систем. Использование системы Win Prolog (LPA 4200) для решения задач на Прологе в среде Microsoft Windows. | 2 | 4 |  | УК-1,  ОПК-1, ОПК-3,  ПК-3 |  | УК-1,  ОПК-1,  ОПК-3,  ПК-3 |
| 6 - 7 | **Пролог для синтаксического анализа** Нисходящий синтаксический разбор контекстно-свободных языков. «Наивный» подход к использованию Пролога в синтаксическом разборе. Встроенный механизм DCG - Definite Close Grammar. Учёт контекстной зависимости. Работа со словарями. Анализ и вычисление арифметических и логических выражений. Перевод с ограниченного естественного языка на язык исчисления предикатов первого порядка. Примеры морфологического анализа. Синтаксический анализ программ, представленных на абстрактном языке программирования операторного типа | 2 | 4 |  | УК-1,  ОПК-1, ОПК-3,  ПК-3 |  | УК-1,  ОПК-1,  ОПК-3,  ПК-3 |
| 8 - 10 | **Пролог и базы данных** Программы как данные. Связь Пролога с реляционными базами данных. Реализация операций реляционной алгебры. Модели искусственного интеллекта на Прологе. Редукционные модели. Эвристический поиск на примере программирования игр. Игра двух лиц с полной информацией. Минимаксная процедура. Альфа-бета процедура. Пролог и распознавание образов. Анализ изображений тел с плоскими гранями. Пролог как инструментальное средство разработки прототипов экспертных систем | 3 | 6 |  | УК-1,  ОПК-1, ОПК-3,  ПК-3 |  | УК-1,  ОПК-1,  ОПК-3,  ПК-3 |

Семинарские занятия (8 семестр)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Недели** | **Темы занятий** | **Компетенции по темам, проверяемые при текущем контроле** | **Виды тек.контроля по проверке компетенций** | **Компетенции по темам, проверяемые на зач. /экз.** |
| 1 | **Тема 1.** Примеры и задачи на основные понятия "чистого" логического программирования: логический вывод, унификацию, полные и частичные вычисления, значения и означивания логической программы |  |  | УК-1,  ОПК-1, ОПК-3,  ПК-3 |
| 2 | **Тема 2.** Примеры и задачи на понятия логического программирования, связанные со списками, экстенсиональными и интенсиональными отношениями, полнотой и корректностью логических. | УК-1,  ОПК-1, ОПК-3,  ПК-3 | КТР4 | УК-1,  ОПК-1, ОПК-3,  ПК-3 |
| 3 - 4 | **Тема 3.** Основы программирования на языке Пролог. Отличия Пролога от "чистого" логического программирования. Использование "встроенных" предикатов. |  |  | УК-1,  ОПК-1, ОПК-3,  ПК-3 |
| 5 | **Тема 4.** Основы программирования на Прологе. Методы входящей и исходящей рекурсии, применение предиката отсечения. |  | ТДЗ8 | УК-1,  ОПК-1, ОПК-3,  ПК-3 |
| 6 | **Тема 5.** Программирование синтаксических анализаторов на Прологе. Механизм DCG. Программирование анализаторов для примеров абстрактных контекстно-свободных и контекстно-зависимых языков. |  |  | УК-1,  ОПК-1, ОПК-3,  ПК-3 |
| 7 | **Тема 6.** Программирование синтаксических анализаторов на Прологе. Программирование анализаторов для содержательных примеров контекстно-свободных и контекстно-зависимых языков. |  |  | УК-1,  ОПК-1, ОПК-3,  ПК-3 |
| 8 | **Тема 7.** Программирование синтаксических анализаторов на Прологе. Программирование анализаторов для примеров ограниченных естественных языков. | УК-1,  ОПК-1, ОПК-3,  ПК-3 | КТР13 | УК-1,  ОПК-1, ОПК-3,  ПК-3 |
| 9 | **Тема 8.** Программирование моделей искусственного интеллекта на Прологе - эвристический поиск. |  | ТДЗ15 | УК-1,  ОПК-1, ОПК-3,  ПК-3 |
| 10 | **Тема 9.** Программирование моделей искусственного интеллекта на Прологе - распознавание изображений. | УК-1,  ОПК-1, ОПК-3,  ПК-3 |  | УК-1,  ОПК-1, ОПК-3,  ПК-3 |

**Соответствие оценочных средств видам контроля**

|  |  |
| --- | --- |
| **Вид контроля** | **Наименование оценочного средства (способ оценки: устно/ письменно /комп.технолог.)** |
| КТР | Выполнение контрольно-тестовой работы (письменно) |
| ТДЗ | Выполнение тематического домашнего задания (дистанционно) |
| КИ | Контроль по итогам выполнения (интегральная оценка без проведения дополнительного контроля) |
| З | Зачет по курсу «Логическое программирование» (устно, с применением персональных компьютеров) |

|  |
| --- |
| МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ |
| федеральное государственное автономное образовательное учреждение  высшего образования |
| **«Национальный исследовательский ядерный университет «МИФИ»** |
| ИНСТИТУТ ИНТЕЛЛЕКТУАЛЬНЫХ КИБЕРНЕТИЧЕСКИХ СИСТЕМ |
| КАФЕДРА КИБЕРНЕТИКИ (№ 22) |

**Задачи, предлагаемые на зачете по дисциплине**

**«Логическое программирование»**

|  |  |  |
| --- | --- | --- |
| Составитель | Волченков Н.Г. | К.т.н., доцент |
| Год | 2019 |  |

**Вариант 1.**

Доказать, что определение предиката:

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

**max(X, Y, Y) :- !.**

некорректно.

**Вариант 2.**

В БД Пролога есть факты:

**w\_list("anna, boris, vladimir").**

**g\_list("eyuioa").**

1. Определить предикат ug(X), который выдаёт гласную, которой нет в первом списке. (Использовать предикат name/2 для того чтобы выдавались буквы, а не их коды.)

2. С помощью предикатов setof/3 и length/2 получить весь список таких гласных и их количество:

**| ?- …**

**L = [e,u,y] ,**

**N = 3**

**Вариант 3.**

В БД Пролога есть факты:

**w\_list("anna, boris, vladimir").**

**s\_list("qwrtpsdfghjklzxcvbnm").**

1. Определить предикат **ns(X)**, который выдаёт согласную, которая есть в первом списке. (Использовать предикат **name/2** для того чтобы выдавались буквы, а не их коды.)

2. С помощью предикатов **setof/3** и **length/2** получить весь список таких согласных и их количество:

**| ?- …**

**L = [b,d,l,m,n,r,s,v] ,**

**N = 8**

**Вариант 4.**

Есть программа на Прологе:

**input\_list([если, х, животное, и, х, ест\_сырое\_мясо, то, х, хищник]).**

**task(Res) :- input\_list(L), an\_s(Res,L,[]).**

**an\_s((L:-R)) --> [если], an\_r(X, R), [то], an\_l(X, L).**

**an\_r(X,( (Z1,Z2))) --> [х,A,и,х,B], {Z1=..[A,X], Z2=..[B,X]}.**

**an\_l(X, Z) --> [х,A], {Z=..[A,X]}.**

Что выдаёт эта программа по запросу

**?- task(Res).**

Контекстно-свободная или контекстно-зависимая грамматика реализована в данном анализаторе?

**Вариант 5.**

Есть программа на Прологе:

**input\_list([если, х, животное, и, х, ест\_сырое\_мясо,**

**то, х, хищник]).**

**task(Res) :- input\_list(L), an\_s(Res,L,[]).**

**an\_s((L:-R)) --> [если], an\_r(X, R), [то], an\_l(X, L).**

**an\_r(X,( (Z1,Z2))) --> [х,A,и,х,B], {Z1=..[A,X], Z2=..[B,X]}.**

**an\_l(X, Z) --> [х,A], {Z=..[A,X]}.**

Что нужно добавить в правую часть правила **task(Res) :- …**

чтобы результат – значение переменной **Res –**

был занесён в БД Пролога?

**Вариант 6.**

Есть программа на Прологе:

**input\_list([если, х, животное, и, х, ест\_сырое\_мясо,**

**то, х, хищник]).**

**task :- input\_list(L), an\_s(Res,L,[]), assert(Res).**

**an\_s((L:-R)) --> [если], an\_r(X, R), [то], an\_l(X, L).**

**an\_r(X,( (Z1,Z2))) --> [х,A,и,х,B], {Z1=..[A,X], Z2=..[B,X]}.**

**an\_l(X, Z) --> [х,A], {Z=..[A,X]}.**

Что нужно добавить в БД Пролога (какие факты), чтобы можно было проверить, что Васька – это хищник? И как это сделать

**Вариант 7.**

Есть программа на Прологе:

**input\_list([если, х, животное, и, х, ест\_сырое\_мясо,**

**то, х, хищник]).**

**кот(Васька).**

**ест\_сырое\_мясо(Васька).**

**task :- input\_list(L), an\_s(Res,L,[]) , assert(Res).**

**an\_s((L:-R)) --> [если], an\_r(X, R), [то], an\_l(X, L).**

**an\_r(X,( (Z1,Z2))) --> [х,A,и,х,B], {Z1=..[A,X], Z2=..[B,X]}.**

**an\_l(X, Z) --> [х,A], {Z=..[A,X]}.**

Что нужно добавить в БД Пролога, чтобы проверить, что Васька – это хищник? С помощью какого запроса можно это сделать?

**Вариант 8.**

Входная цепочка хранится в БД Пролога в виде факта:

**input\_list(“кобра123, дракон75, пантера201,…”).**

(Аргумент – список юникодов, содержащий коды запятой, которая разделяет буквенные идентификаторы, вслед за которыми идут цифры.)

Написать грамматику и анализатор на DCG Пролога, результатом работы которого являются факты в БД Пролога:

**кобра(123). дракон(75). пантера(201).** И т.д.

(Идентификаторы превращаются в функторы, а числа – в аргументы. )

**Вариант 9.**

  Язык **L = {afibon(n) bn cfibon(n)},**

где fibon(n) = 1, 1, 2, 3, 5, 8, … (числа Фибоначчи)

для n = 1, 2, 3, 4, 5, 6, …

Написать анализатор, используя механизм DCG, результатом работы которого является значение **n**. Результат должен быть, например, таким:

**| ?- an\_s(N, “aaaaaaaabbbbbbcccccccc”, []).**

**N = 6**

*Указание*: по ходу разбора выясняем значения M и N – длину, соответственно, подцепочки из символов **a** (или равной ей по длине подцепочки из символов **c**) и подцепочки из символов **b**. Затем вычисляем K = fibon(N).

Если K = M, разбор завершается успехом. (Между делом, надо вспомнить определение предиката **fibon/2**.)

**Вариант 10.**

 Написать определение предиката **перестановка(L1, L2)**,

где **L1** – произвольный список, например: [a, b, c];

**L2** – одна из перестановок элементов этого списка, например: [b, a, c].

С помощью предиката **setof/3** выдать все решения:

L = [[a,b,c], [a,c,b], [b,a,c], [b,c,a], [c,a,b], [c,b,a]].

*Указание*: использовать предикат **member\_delete(X, L, M)**, где X – элемент списка ; M – список L после удаления из него этого элемента.

Алгоритм такой: рекурсивно удаляем из списка *произвольный* элемент и применяем предикат **перестановка/2** к оставшемуся после удаления списку.

**Вариант 11.**

Один список состоит из нескольких (например, из 5) согласных букв, а второй список состоит из нескольких

(например, тоже из 5) гласных букв.

Определить на Прологе генератор всех слов, состоящих из 4-х букв – чередующихся согласных и гласных, например:

**ВАСЯ, МАША, МАМА, ПАПА** и т.д. (всего 54 слов).



**Вариант 12.**

Множество слов, состоящих из 9 букв, задано списком Пролога, например:

**L = [ богомолов, водовозов, коновалов, перевалов, … ]**.

**Вариант 13.**

  Язык L = {an! bn c(n+1)!}, где n! = 1, 2, 6, 24, 120, … для n = 1, 2, 3, 4, 5, …

Написать анализатор, используя механизм DCG, результатом работы которого является значение n. Результат должен быть, например, таким:

**| ?- an\_s(N, “aaaaaabbbcccccccccccccccccccccccc”, []).**

**N = 3**

*Указание*: по ходу разбора выясняем значения N1, N2 и N3 – длину, соответственно, подцепочки из символов a, подцепочки из символов b и подцепочки из символов c.

Затем проверяем: N1 = factor(N2); N3 = factor(N2+1).

Если это так, разбор завершается успехом.

(Между делом, надо вспомнить самое простое (необратимое) определение предиката factor/2.)

**Вариант 14.**

Входная цепочка хранится в БД Пролога в виде факта:

**input\_list([кобра, 123, дракон, 75, пантера, 201, …]).**

(Аргумент – список, на нечетн**input\_list([кобра, 123, дракон, 75, пантера, 201, …]).**

ых местах которого идентификаторы, а на четных – числа.)

Написать программу на Прологе, результатом работы которой являются факты в БД Пролога:

**кобра(123). дракон(75). пантера(201).** И т.д.

(Идентификаторы превращаются в функторы, а числа – в аргументы. )

**Вариант 15.**

  Язык L = {β reverse(β)}, где β = {a, b, c}+, reverse(β) – реверс цепочки β.

Написать анализатор, используя механизм DCG, результатом работы которого является значение n. Результат должен быть, например, таким:

**| ?- an\_s(R, “cbbacbaaaabcabbc”, []).**

R = s(…) – дерево синтаксического разбора.

**Вариант 16.**

Написать определение предиката **генератор(N, L, R)**,

где целое число N ϵ [L, R] – выходной параметр;

L, R – целые числа: L < R.

Пример обработки вызова:

**?- setof(N, генератор(N, -2, 3), List).**

**List = [-2, -1, 0, 1, 2, 3].**

**Вариант 17.**

Язык L = {ambnck}, где m = n2, k = n3.

Написать анализатор, используя механизм DCG, результатом работы которого является значение n. Результат должен быть, например, таким:

**| ?- an\_s(N, “aaaabbcccccccc”, []).**

**N = 2**

*Указание*: по ходу разбора выясняем значения m, n и k – длину подцепочек из символов a, b и c. Затем вычисляем n2 и n3. Если m = n2, k = n3, разбор завершается успехом.

**Вариант 18.**

Написать определение предиката

**пересечение(L1, L2, L3)**,

где первые два аргумента – произвольные списки, каждый из которых не содержит дубликатов,

например:

L1 = [a, b, c]; L2 = [c, d, a].

L3 – список из общих элементов: [a, c].

**Вариант 19.**

 Язык L = {an! b3n cn!}, где n! = 1, 2, 6, 24, 120, … для n = 1, 2, 3, 4, 5, …

Написать анализатор, используя механизм DCG, результатом работы которого является значение n. Результат должен быть, например, таким:

**| ?- an\_s(N, “aaaaaabbbbbbbbbcccccc”, []).**

**N = 3**

*Указание*: по ходу разбора выясняем значения N1 и N2 – длину, соответственно, подцепочки из символов **a** (и равной ей по длине подцепочки из символов **c**) и подцепочки из символов **b**. Затем проверяем: N1 = factor(N2/3).

Если это так, разбор завершается успехом.

(Между делом, надо вспомнить самое простое (необратимое) определение предиката factor/2.)

**Вариант 20.**

Написать определение предиката

**объединение(L1, L2, L3)**,

где первые два аргумента – произвольные списки, каждый из которых не содержит дубликатов,

например:

L1 = [a, b, c]; L2 = [c, d, a].

L3 – список из элементов как первого, так и второго списков: L3 = [a, b, c, d].

|  |
| --- |
| МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ |
| федеральное государственное автономное образовательное учреждение  высшего образования |
| **«Национальный исследовательский ядерный университет «МИФИ»** |
| ИНСТИТУТ ИНТЕЛЛЕКТУАЛЬНЫХ КИБЕРНЕТИЧЕСКИХ СИСТЕМ |
| КАФЕДРА КИБЕРНЕТИКИ (№ 22) |

**Варианты тематических домашних заданий (ТДЗ)**

**по дисциплине**

**«Логическое программирование»**

|  |  |  |
| --- | --- | --- |
| Составитель | Волченков Н.Г. | К.т.н., доцент |
| Год | 2019 |  |

**ТДЗ № 1**

*Часть 1. «Декларативные свойства языка Пролог»*

*База данных – для всех вариантов:*

родитель(Дарья,Софья).

родитель(Дарья,Галина).

родитель(Дарья,Георгий).

родитель(Иван,Софья).

родитель(Иван,Галина).

родитель(Иван,Георгий).

родитель(Галина,Николай).

родитель(Галина,Сергей).

родитель(Галина,Наталья).

родитель(Геннадий,Николай).

родитель(Геннадий,Сергей).

родитель(Ирина,Елена).

родитель(Николай,Елена).

родитель(Татьяна,Александр).

родитель(Сергей,Александр).

родитель(Сергей,Евгения).

родитель(Елизавета,Владимир).

родитель(Елизавета,Ольга).

брат(Геннадий,Елизавета).

list\_man([Иван,Георгий,Геннадий,Николай,Сергей,Владимир,Александр]).

list\_woman([Дарья,Софья,Галина,Елизавета,Наталья,Ирина,Татьяна,Ольга,Елена,Евгения]).

мужчина(X):-list\_man(L), member(X,L).

женщина(X):-list\_woman(L), member(X,L).

Примечание: порядок аргументов предиката **родитель(X, Y)** такой:

X – кто родитель; Y – чей родитель и т.д.

В каждом из следующих вариантов определить соответствующие родственные отношения (**C1/2, C2/2, C3/2**) и составить запрос в виде: **?- C1(X,Y), C2(Y,Z), C3(Z,V).** (Значение переменной **X** или **Y** должно быть задано.) Продемонстрировать получение ответа на данный запрос.

1. Найти племянника зятя бабушки Александра.
2. Найти двоюродного брата сына внука Дарьи.
3. Найти тетю внука отца Софьи.
4. Найти племянницу дедушки дочери Николая.
5. Найти предков двоюродного брата сына Елизаветы.
6. Найти правнука мужа матери Георгия.
7. Найти жену племянника дочери Дарьи.
8. Найти правнучку тестя отца Сергея.
9. Найти прадедушку двоюродного брата дочери Николая.
10. Найти отца правнучки тестя Геннадия.
11. Найти двоюродную сестру правнука матери Георгия.
12. Найти мать правнука тещи Геннадия.
13. Найти тещу дедушки сына Татьяны.
14. Найти отца двоюродной сестры племянника Николая.
15. Найти внука зятя матери Галины.
16. Найти мать двоюродного брата племянницы Сергея.
17. Найти жену дяди двоюродной сестры Александра.
18. Найти жену племянника сына Дарьи.
19. Найти жену внука прадедушки Елены.
20. Найти жену племянника брата Софьи.
21. Найти племянника внука прабабушки Александра.
22. Найти брата свекрови матери Александра.
23. Найти дядю правнука тёщи Геннадия.
24. Найти предков двоюродной сестры сына Татьяны.
25. Найти предков племянницы сына Геннадия.
26. Найти потомков дедушки племянника Софьи.
27. Найти потомков прабабушки племянницы Сергея.

*Часть 2. «Процедурные свойства Пролога»*

**Написать программу на Прологе, решающую задачу:**

1. В заданном списке слов русского языка найти такое слово, которое содержит четное число гласных букв.
2. В заданном списке слов русского языка найти такое слово, которое содержит нечетное число согласных букв.
3. В заданном списке слов русского языка найти такое слово, которое содержит, по крайней мере, на две согласных больше, чем гласных.
4. В заданном списке слов русского языка найти такую пару, для которой первая буква одного слова совпадает с последней буквой другого и наоборот.
5. В заданном списке слов русского языка найти такое слово, длина которого вдвое меньше длины самого длинного слова этого списка.
6. В заданном списке слов русского языка найти такое, число гласных в котором кратно заданному числу.
7. В заданном списке слов русского языка найти такое слово, длина которого в точности равна среднему арифметическому длины самого длинного и самого короткого слова.
8. Заданный список слов русского языка разбить на два подсписка: в одном должны быть слова, начинающиеся с гласной, а во втором — с согласной буквы.
9. Найти не самое длинное и не самое короткое слово в заданном списке слов русского языка.
10. Найти слово, вдвое или втрое длиннее самого короткого слова заданного списка слов русского языка.
11. Из заданного списка слов русского языка создать два списка: в первом из них должны быть слова, начинающиеся с гласной буквы, а во втором — заканчивающиеся на гласную букву.
12. В заданном списке слов русского языка найти такое, у которого число букв А равно числу букв В (А и В — входные параметры).
13. Найти букву, наиболее часто встречающуюся в заданном списке слов русского языка.
14. Найти букву, не встретившуюся в заданном списке слов русского языка.
15. В примере выполнения данной части ДЗ дан список пар городов, между которыми есть авиарейсы. Найти маршрут перелета из города А в город В, не содержащий города С.
16. В примере выполнения данной части ДЗ дан список пар городов, между которыми есть авиарейсы. Найти циклический маршрут, длина которого (по количеству промежуточных пунктов) не меньше заданного числа.
17. В примере выполнения данной части ДЗ дан список пар городов, между которыми есть авиарейсы. Найти два маршрута из города А в город В, в которых нет совпадающих промежуточных городов.
18. В примере выполнения данной части ДЗ дан список пар городов, между которыми есть авиарейсы. Найти маршрут, который длиннее самого короткого маршрута в два раза (по количеству промежуточных пунктов).
19. В примере выполнения данной части ДЗ дан список пар городов, между которыми есть авиарейсы. Найти суммарную длину всех возможных маршрутов из города А в город В (по количеству промежуточных пунктов).
20. В примере выполнения данной части ДЗ дан список пар городов, между которыми есть авиарейсы. Найти два города, между которыми есть по крайней мере два разных маршрута.
21. В примере выполнения данной части ДЗ дан список пар городов, между которыми есть авиарейсы. Найти маршрут, длина которого (по количеству промежуточных пунктов) больше кратчайшего маршрута из города А в город В в три раза.
22. В примере выполнения данной части ДЗ дан список пар городов, между которыми есть авиарейсы. Найти разность между самым длинным и самым коротким (по количеству промежуточных пунктов) маршрутом из города А в город В.
23. В примере выполнения данной части ДЗ дан список пар городов, между которыми есть авиарейсы. Найти не самый длинный и не самый короткий путь из города А в город В.
24. Дан список взвешенных дуг ориентированного графа. Найти путь с заданным весом, начинающийся из заданной вершины.
25. Дан список взвешенных дуг ориентированного графа. Найти путь с заданным весом, заканчивающийся в заданной вершине.
26. Дан список взвешенных дуг ориентированного графа. Найти путь в графе, вес которого принадлежит заданному интервалу, и проходящий через заданную вершину.
27. Дан список взвешенных дуг ориентированного графа. Найти путь в графе, вес которого принадлежит заданному интервалу, и не содержащий заданную вершину.

ТДЗ № 2

*Вариант 1.*

1. Построить синтаксический анализатор для языка

L = { (ab)ncc }  { cfibon(m)(ba)m }, n ≥ 0, m ≥ 1.

2. Реализовать разбор и вычисление значений арифметических выражений, содержащих функции: {sin, cos, tan, log, ln, exp, sqrt}. Для простоты считать, что круглые скобки в выражении используются только для обрамления аргумента функции, а из арифметических операций используются только сложение (+) и умножение (\*).

Пример:

Исходная цепочка: [log, ’(’, 100, ’)’, ’+’, sin, ’(’, 90, ’)’, ’\*’, 5]

Результат: 7.0

*Вариант 2.*

1. Построить синтаксический анализатор для языка

L = { amb(m-1)! }  { a(k+1)!bk }, m≥1, k≥0.

2. Составить грамматику и реализовать анализатор целых чисел от 0 до 1000, записанных с помощью римских цифр, с преобразованием их в арабские, например, так:

?-rim\_arab(“DLXVIII”, Arab).

Arab = 568

?-rim\_arab(“CMXCIX”, X).

Arab =999

?-rim\_arab(“”, X).

Arab = 0 (римляне не знали нуля, но мы-то знаем!)

*Указание:* использовать встроенный предикат name/2, преобразующий атом или число в список ASCII-кодов или наоборот.

*Вариант 3.*

1. Построить синтаксический анализатор для языка

L = { banbban-1bbban+1 }  { c(2k)! }, n≥1, k≥0.

2. Построить синтаксический анализатор для распознавания «квадратов» — цепочек типа “uuuurrrddddlll”, “rrrdddlllluuu”, … (u — стрелка единичной длины, идущая вверх, r — вправо, d — вниз, l — влево). Длины подцепочек, содержащих символы только одного из этих видов, должны быть *примерно* одинаковой длины (отличаться от средней их длины не более чем на заданное число). Например:

?-an\_s(0.8, Size, “uurrdddll”, []).

Size=2.25 (это «средняя длина» цепочки: Size = 9/4 = 2.25)

yes (цепочка признаётся «правильной», так как 0.25<0.8, 0.25<0.8, 0.75<0.8, 0.25<0.8).

*Вариант 4.*

1. Построить синтаксический анализатор для языка

L = { an+1(bb)ncn+1 }  { (bac)m^2 }, n≥0, m≥0.

2. Составить грамматику и реализовать разбор «утверждений» — фраз языка, представленного примерами:

* Большая книга лежит под столом.
* Красная книга находится на столе.
* Маленькая книга красная.
* Большая книга синяя.

В результате разбора в базе данных Пролога должны появиться факты типа:

* st(location(object(book, ch(size(big), color(X))), under(table))).
* st(location(object(book, ch(size(X), color(red))), on(table))).
* st(book(size(big), color(blue))).
* st(book(size(little), color(red))).

*Вариант 5.*

1. Построить синтаксический анализатор для языка

L = { (AB)n }  { (BA)m }, n≥1, m≥1,

A — гласная буква, B — согласная буква. (Алфавит английский.)

(Примеры цепочек данного языка: «banana», «ananas», «kukuruza», и т.д.)

1. Реализовать грамматический разбор фраз на ограниченном естественном языке типа:

«Если А — кот, то существует собака B, которая гоняется за А»,

«Если А — юноша, B — девушка, C — вещь, которой владеет А,

B нравится юноше А, C нравится девушке B, то А дарит C девушке B»

На выходе — формулы исчисления предикатов 1-го порядка.

*Вариант 6.*

1. Построить синтаксический анализатор для языка

L = { (aa)2k-1c(aa)2m }  { ccbfibon(n) }, k≥1, m≥0, n≥1.

1. Реализовать разбор цепочек, представляющих вопросы типа:

[Кто, брат, Ивана, ’?’]

[Чей, брат, Пётр, ’?’]

[Иван, брат, Петра, ’?’]

В базе данных Пролога есть такие факты:

br(Сергей, Иван). br(Иван, Пётр). br(Пётр, Андрей). br(Иван, Андрей). Это данные о братьях.

После анализа каждой цепочки необходимо сформировать запрос к этой БД типа:

?- br(X, Иван).

Должен быть получен ответ: X=Сергей; X=Пётр; и т.д.

*Вариант 7.*

1. Построить синтаксический анализатор для языка

L = { ck(abc)p(k) }  { ambm!am }, k≥1, m≥0.

p(k) – число, k-е в ряду простых чисел: 2, 3, 5, 7, 11 и т.д.

2. Реализовать разбор и вычисление значений логических выражений, содержащих функции:

{&, V, ~, =>}

Пример: [’~’, bob, ’&’, john, ’=>’, mary]

Атом считается истинным, если он есть в БД Пролога в виде факта. (Например, если в БД есть факт john., но нет фактов bob. и mary., то данное выражение будет иметь значение «ложь».)

*Вариант 8.*

1. Построить синтаксический анализатор для языка

L = { anbn+kcndn+k } { amb(m-2)!}, n≥0, k≥1, m>2.

1. Реализовать разбор и вычисление значений арифметических выражений, атомы которого — целые 16-ричные числа. Использовать только операции сложения и умножения и скобки, например:

(7A + C2) \* (55 + D4)

Должен получиться результат: 93852.

Данная цепочка представляется в виде списка:

[ ’(’, ’7A’, ’+’, ’C2’, ’)’, ’\*’, ’(’, ’55’, ’+’, ’D4’, ’)’].

*Вариант 9.*

1. Построить синтаксический анализатор для языка

L = { anbanbban }  { ak-3cs(k)}, n≥0, k>2, s(k) – сумма арифметической прогрессии: s(k) = 1+2+…+k.

1. Числовые значения атомов заданы в виде фактов в БД Пролога, например:

{ aaa(5), bbb(8), ccc(7), ...}.

Построить синтаксический анализатор бесскобочных арифметических выражений, в которых вместо чисел — атомы, например: aaa + bbb \* ccc.

(Для простоты, пусть в выражениях используются только операции сложения и умножения.) На выходе — числовое значение. (Например, в данном случае, 61.)

*Вариант10.*

1. Построить синтаксический анализатор для языка

L = { cp(k)(ab)3k }  { f2mdem }, k≥1, m≥1, p(k) – k-e простое число в ряду: 2, 3, 5, 7, ...

2. Реализовать разбор морфологии глаголов: {прибежал, подбежала, бегали, убегал, ...}.

*Корни* (бег, беж, ...) — в самой грамматике,

*приставки* (при, под, у, ...) — в «словаре» приставок, представленного в виде факта:

приставки([при, под, у, …]).

*окончания* (ал, ала, ало, али) — в самой грамматике.

Результат должен содержать сведения о роде и числе. Пример:

Исходная цепочка: "подбежала".

Результат: morf(prist(под), kor(беж), rod(жен), chislo(един)).

*Вариант 11.*

1. Построить синтаксический анализатор для языка

L = L1·L2  L2·L1, L1 = B+, L2 = N+,

B — строчные латинские буквы, N — цифры.

Например: «cobra123», «485dragon» — цепочки языка.

1. Построить преобразователь активных и пассивных форм типа:

[Пётр, и, Иван, любят, рок] и

[рок, любим, Петром, и, Иваном]

в глубинные структуры типа:

likes(agent(Пётр), object(рок)). likes(agent(Иван), object(рок)).

*Вариант 12.*

1. Построить синтаксический анализатор для языка

L = { anb2nc3n }  { cfibon(m)b2m }, n≥0, m≥1.

2. Составить грамматику и реализовать разбор фраз языка, представленного примерами:

* "Петя читает";
* "Маша и Миша рисуют и разговаривают";
* "Ваня, Вася, Катя и Маруся читают и пишут".

В результате разбора в базе данных Пролога должны появиться факты типа:

читает(Петя). читает(Ваня). пишет(Ваня).

и т.д.

*Вариант 13.*

1. Построить синтаксический анализатор для языка

L = { anbn+1cn+2 }  { c2m(ba)m! }, n≥0, m≥1.

2. Составить грамматику и реализовать разбор фраз языка, представленного примерами:

* "Кто читает газету ?";
* "Что читают Ваня, Вася, Катя и Маруся ?";
* "Где лежит книга ?";
* "Кто сидит за столом ?".

В результате разбора в базе данных Пролога должны появиться факты типа:

читать(agent(X), object(газета)).

читать(agent(Ваня), object(X)).

читать(agent(Вася), object(X)).

лежать(object(книга), loc(X)).

и т.д.

*Вариант 14.*

1. Построить синтаксический анализатор для языка

L = { anb2ncnd2n }  { x reverse(x)}, n≥0, x – произвольная цепочка из символов словаря {a,b,c}.

2. Составить грамматику и реализовать разбор фраз языка, представленного примерами:

* "Петя любит футбол, но не любит баскетбол и регби";
* "Маша не любит бананы, но любит апельсины, лимоны и мандарины";
* "Ваня не любит пепси и пиво, а любит чай и кофе".

В результате разбора каждой фразы списки L1 и L2 фактов likes(L1) и rejects(L2) в базе данных Пролога должны пополниться новыми элементами, например:

likes([(Петя, футбол)|…]).

rejects([(Ваня, пепси), (Ваня, пиво)|…]).

Изначально, списки в этих двух фактах пустые.

*Вариант 15.*

1. Построить синтаксический анализатор для языка

L = { (ab)ncnbn }  { (ba)m!cm^2 }, n≥0, m≥1.

2. Составить грамматику и реализовать разбор фраз языка, представленного примерами:

* "150000, г. Ярославль, ул. Циммервальде, д. 1, кв. 10, Собинову И.И.";
* "127591, Москва, ул. 800-летия Москвы, д. 99, кор. 3, кв. 118, Собак Фиме";
* "171273, Тверская обл, Конаковский р-н, пос. Зюзино, ул. К.Маркса, д. 5,

Пульхерии Петровне Подберёзкиной".

Для простоты считать, что элементами исходного списка являются атомы, полученные из исходной цепочки путём объединения символов между разделителями: " " (пробелом) или ", " (запятой с пробелом) или ". " (точкой с пробелом).

В результате разбора должна быть получена структура типа:

address(index(150000), city(’Ярославль’), street(’Циммервальде’), house(’1’),

apartment(’10’), human(’Собинову И.И.’))

*Вариант 16.*

1. Построить синтаксический анализатор для языка

L = { cbna2n+1bnc }  { cm!(ba)(m-1)! }, n≥0, m≥1.

2. Составить грамматику и реализовать разбор фраз языка, представленного примерами:

* "Детали, лежащие на столе, красные.";
* "Ваза, стоящая на столе, зелёная и большая.";
* "Книга, лежащая под столом, толстая.".

В результате разбора в базе данных Пролога должны появиться факты типа:

char(object(детали), loc(at(стол)), value([color(красный)])).

char(object(ваза), loc(at(стол), value([color(зелёный), size(большой)])).

char(object(книга), loc(under(стол), value([size(толстый)])).

и т.д.

*Вариант 17.*

1. Построить синтаксический анализатор для языка

L = { anbn+2cn-2dn }  { bfibon(m-1)am }, n≥2, m≥1.

2. Составить грамматику и реализовать разбор фраз языка, представленного примерами:

* "Учитель входит в класс .";
* "Ученики сидят за партами .";
* "Сидят ли ученики за партами ?";
* "Входит ли учитель в класс ?".

В результате разбора в базе данных Пролога должны появиться факты типа:

входить(status(statement), agent(учитель), loc(класс)).

сидеть(status(statement), agent(ученики), loc(парты)).

входить(status(question), agent(ученики), loc(парты)).

сидеть(status(question), agent(учитель), loc(класс)).

*Вариант 18.*

1. Построить синтаксический анализатор для языка

L = { (ab)2+k!ck }  { d3me2mfm }, k≥0, m≥1.

2. Реализовать разбор морфологии глаголов:

{приедет, подъедет, ехали, уехал, ...}.

Корни (ед, ех, ...) — в самой грамматике,

приставки (при, под, у, ...) — в «словаре» приставок, представленных в виде факта:

приставки([при, под, у, …]).

окончания (ал, ала, ало, али) — в самой грамматике.

Результат должен содержать сведения о роде и числе.

*Пример:* Исходная цепочка: "подъехала"

Результат: morf(prist(под), kor(ех), rod(жен), chislo(един)).

*Вариант 19.*

1. Построить синтаксический анализатор для языка

L = { edk }  { (bab)2m cm! (bab)3m}, k≥1, m≥0.

2. Составить грамматику и реализовать разбор фраз языка, представленного примерами:

* *Животное птица, если оно имеет перья или если оно умеет откладывать яйца и имеет крылья или умеет летать.*
* *Млекопитающее хищник, если оно умеет есть мясо и — либо имеет острые зубы, либо имеет когти, либо имеет смотрящие вперёд глаза.*

В результате разбора в базе данных Пролога должны появиться правила типа:

* *птица(Х) :- животное(Х), имеет(Х, перья).*
* *птица(Х) :- животное(Х), умеет(откладывать\_яйца), (умеет(Х, летать); имеет(крылья)).*
* *хищник(Х) :- млекопитающее(Х), умеет(есть\_мясо), (имеет(острые\_зубы); имеет(когти); имеет(смотрящие\_вперёд\_глаза)).*

*Вариант 20.*

1. Построить синтаксический анализатор для языка

L = { (aa)m!cb2m+1 }  { b2n-1 cn b2n-1 }, m≥0, n≥1.

2. Составить грамматику и реализовать разбор фраз языка, представленного примерами:

* *Если некое животное есть птица и если оно не умеет летать и если оно умеет плавать и если оно имеет чёрно-белый цвет, то это животное пингвин.*
* *Если некто есть хищник и если оно имеет рыже-коричневый цвет и если оно имеет чёрные полосы, то это тигр.*

В результате разбора в базе данных Пролога должны появиться правила типа:

* *пингвин(Х) :- животное(Х), птица(Х), не(умеет(Х, летать)), умеет(плавать), имеет(чёрно-белый\_цвет).*
* *тигр(Х) :- хищник(Х), имеет(чёрные\_полосы), имеет(рыже-коричневый\_цвет).*

*Вариант 21.*

1. Построить синтаксический анализатор для языка

L = { (ca)2m+1cfibon(m) }  { bk+1ak-1bk+1 }, k≥1, m≥0.

2. Составить грамматику и реализовать разбор фраз языка, представленного примерами:

* *Млекопитающее — это животное, которое имеет шерсть или кормит детёнышей молоком («умеет кормить\_детёнышей\_молоком»).*
* *Гепард — это хищник, который имеет рыже-коричневый цвет и имеет чёрные пятна.*

В результате разбора в базе данных Пролога должны появиться правила типа:

* *млекопитающее(Х) :- животное(Х), имеет(шерсть).*
* *млекопитающее(Х) :- животное(Х), умеет(кормить\_детёнышей\_молоком).*
* *гепард(Х) :- хищник(Х), имеет(чёрные\_пятна), имеет(рыже-коричневый\_цвет).*

*Вариант 22.*

1. Построить синтаксический анализатор для языка

L = { anb(n-1)!c(n+1)! }  { bfibon(m+1)am }, n>0, m≥0.

2. Составить грамматику и реализовать анализатор целых арабских чисел от 0 до 999 с преобразованием их в «римские», например, так:

?-arab\_rim(“568”, X).

X=’DLXVIII’

?-arab\_rim(“999”, X).

X=’CMXCIX’

?-arab\_rim(“0”, X).

X=’’ (римляне не знали нуля) – это не двойная кавычка, а две одинарные кавычки (!)

?-arab\_rim(“1000”, X).

X=’?’ (программа не умеет анализировать числа > 999)

*Указание:* использовать встроенный предикат name/2, преобразующий атом или число в список ASCII-кодов или наоборот.

*Вариант 23.*

1. Построить синтаксический анализатор для языка

L = { anb2nc(ca)n }  { bm!am^2 }, n≥1, m≥0.

2. Составить грамматику и реализовать разбор «вопросов» — фраз языка, представленного примерами:

* *Где находится синяя книга?*
* *Где лежит маленькая книга?*
* *Что лежит под столом?*
* *Что лежит на столе?*
* *Каков размер синей книги?*
* *Каков цвет маленькой книги?*
* *Под чем лежит большая книга? (согласовано с вар. 4)*

В результате разбора в базе данных Пролога должны появиться факты типа:

* *q(location(object(book, ch(size(X), color(blue))), Y)).*
* *q(location(object(book, ch(size(little), color(X))), Y)).*
* *q(location(X, under(table))).*
* *q(location(X, on(table))).*
* *q(book(size(X), color(blue))).*
* *q(book(size(little), color(X))).*
* *q(location(object(book, ch(size(big), color(X))), under(Y))).*

*Вариант 24.*

1. Построить синтаксический анализатор для языка

L = { (ab)2kbk(ba)2k } {a4mcm! }, m≥0, k≥1.

1. В базе данных Пролога есть факты вида:

родитель(Марья, Петр).

родитель(Марья, Иван). и т.д.

Запрос к базе данных имеет вид

?-q(X, Y).

Значение 1-го аргумента — одна из цепочек вида:

[Кто, брат, Ивана, ’?’]

[Чей, брат, Пётр, ’?’]

[Иван, брат, Петра, ’?’]

На выходе нужно получить: Y = Пётр, Y = Иван, Y = yes.

*Вариант 25.*

1. Построить синтаксический анализатор для языка

L = { a2nbnc3n }  { (ba)(2m-1)! }, n≥0, m≥1.

2. Составить грамматику и реализовать разбор фраз языка, представленного примерами:

* "Кто любит футбол, но не любит баскетбол и регби?";
* "Что Маша любит и чего она не любит?";

В результате разбора в базе данных Пролога должны появиться факты типа:

person(X, likes([футбол]), rejects([баскетбол, регби])).

person(Маша, likes(X), rejects(Y)).

*Вариант 26.*

1. Построить синтаксический анализатор для языка L = { anb2nc3n }  { cfibon(m)b2m }, n≥0, m≥1.

2. Построить преобразователь активных и пассивных форм типа:

[Петя, и, Саша, любят, футбол] и

[футбол, любим, Петей, и, Сашей]

в глубинные структуры типа:

likes(agent(Петя), object(футбол)). likes(agent(Саша), object(футбол)).

*Вариант 27.*

1. Построить синтаксический анализатор для языка L = L1·L2  L3·L4, где

L1 = B+, L2 = N+, L3 = N+, L4 = B+,

B — строчные русские буквы, N — цифры.

Например: «кобра999», «272дракон» — цепочки языка.

2. Составить грамматику и реализовать разбор фраз языка, представленного примерами:

* "Саша читает";
* "Петя и Миша рисуют";
* "Ваня и Маруся читают и пишут".

В результате разбора в базе данных Пролога должны появиться факты типа:

читает(Саша). читает(Ваня). пишет(Ваня).

и т.д.