Лабораторная работа №1.
Факты и правила.
(5 баллов)

Задание.
Написать программу на языке Пролог, описывающую родственные отношения.

Реализовать следующие правила.

I. Исходные факты – отношения «родитель», «мужчина», «женщина»

1. родитель("Имя_1","Имя_2"). ← человек с именем Имя_1 является родителем человека с именем Имя_2
2. мужчина("Имя"). ← человек с именем Имя является мужчиной
3. женщина("Имя"). ← человек с именем Имя является женщиной

II. Правила, задающие родственные отношения

1. отец(X,Y) ← X является отцом Y
2. мать(X,Y) ← X является матерью Y
3. родители(X,Y) ← X и Y являются родителями одного и того же ребёнка
4. брат(X,Y) ← X является братом по отношению к Y
5. сестра(X,Y) ← X является сестрой по отношению к Y
6. дядя(X,Y) ← X является дядей по отношению к Y
7. тётя(X,Y) ← X является тётей по отношению к Y
8. дети_одних_родителей(X,Y) ← X и Y являются детьми одних и тех же родителей
9. родственники(X,Y) ← X и Y являются родственниками

III. «Проверочные» правила

1. отец(X) ← X является чьим-то отцом
2. мать(X) ← X является чьей-то матерью

Возможное именование термов по-английски

родитель()	parent()
мужчина()	man()
женщина()	woman()
отец()	father()
мать()	mother()
родители()	parents()
брат()	brother()
сестра()	sister()
дядя()	uncle()
тётя()	aunt()
дети_одних_родителей()	children_of_common_parents()
родственники()	relatives()

Лабораторная работа №2.
Структурирование данных и рекурсия.
(5 баллов)

Задание.
Написать программу на языке Пролог, описывающую группу военных.

Реализовать следующие правила.

1. Исходные факты - звания военных

```
военный(фамилия("…"),звание("…")).
```

2. Исходные факты для установки субординации

```
следующее_звание(A,B) :- B является следующим званием по отношению к A
```

Например:
```
следующее_звание("рядовой","сержант").
```

Использовать следующий минимальный набор званий (в порядке возрастания старшинства):

- рядовой;
- сержант;
- лейтенант;
- капитан;
- майор;
- полковник;
- генерал.

3. Равенство званий военных

```
одинаковое_звание(A,B) :- военные A и B имеют одинаковое звание
```

Рекомендация.
Для описания этого правила определить вспомогательное правило для проверки неравенства фамилий
```
разные(фамилия(X),фамилия(Y))
```

4. Отношение субординации

```
субординация(младший(Мл), старший(Ст)) :- военный с фамилией Мл младше по званию,
чем военный с фамилией Ст
```

Рекомендация.
Для описания этого правила определить вспомогательное рекурсивное правило для сравнения старшинства званий
```
младшее_звание(R1,R2) :- звание R1 младше, чем звание R2
```

5. Возможное дальнейшее продвижение по службе

```
цепочка_званий(Военный,Маршрут) :- Маршрут – возможное продвижение по службе для
военного Военный
```

Рекомендации.

а) Цепочка званий формируется с использованием структурирования. Например,

```
звание("капитан",звание("майор",звание("полковник",звание("генерал","высшее звание
")))))
```

б) Для формирования цепочки званий определить вспомогательное рекурсивное правило со структурированием

```
продвижение(R1,звание(R2,Маршрут)) :- звание(R2,Маршрут) –цепочка (список)
следующих званий по отношению к R1
```

Для ограничения рекурсии использовать факт

```
продвижение("генерал","высшее звание").
```

Возможное именование термов по-английски

`военный(фамилия("…"),звание("…"))`	`military_man(lastname("…"),rank("…"))`
`следующее_звание(A,B)`	`next_rank(A,B)`
`одинаковое_звание(A,B)`	`same_rank(A,B)`
`разные(фамилия(X),фамилия(Y))`	`noneq(lastname(X),lastname(Y))`
`субординация(младший(Мл), старший(Ст))`	`subordination(minor(Mn),major(Mj))`
`младшее_звание(R1,R2)`	`minor_rank(R1,R2)`
`цепочка_званий(Военный,Маршрут)`	`chain_of_ranks(Military,Route)`
`продвижение(R1,звание(R2,Маршрут))`	`progress(R1,rank(R2,Route))`

Лабораторная работа №3.
Работа со списками.
(5 баллов)

Задание.

Запрограммировать на языке Пролог указанные ниже предикаты. Продемонстрировать различные варианты их использования на примерах.

I. Предикаты работы со списками

Аргументы L1,L2,L3 обозначают списки, E - некоторый элемент списка (тип элементов списка произволен), N - порядковый номер элемента в списке.

1. append (L1, L2, L3) ← список L3 является слиянием (конкатенацией) списков L1 и L2;
2. reverse (L1, L2) ← L2 – перевернутый список L1;
3. delete_first (E, L1, L2) ← список L2 получен из L1 исключением первого вхождения объекта E;
4. delete_all (E, L1, L2) ← L2 – это список L1, из которого удалены все вхождения E;
5. no_doubles (L1, L2) ← L2 – это список, являющийся результатом удаления из L1 всех повторяющихся элементов;
6. sublist (L1, L2) ← L1 – любой подсписок списка L2, т.е. непустой отрезок из подряд идущих элементов L2;
7. number (E, N, L) ← N – порядковый номер элемента E в списке L;
8. sort (L1, L2) ← L2 – отсортированный по неубыванию список чисел из L1.

II. Предикаты работы со множествами

Аргументы M1, M2, M3 обозначают множества, которые представляются в виде списков элементов без повторений, порядок элементов в них не существенен, тип элементов - произволен.

10. subset (M1, M2) ← множество M1 является подмножеством M2;
11. union (M1, M2, M3) ← множество M3 – объединение множеств M1 и M2;
12. intersection (M1, M2, M3) ← M3 – пересечение M1 и M2;
13. subtraction (M1, M2, M3) ← M3 – разность M1 и M2.

Рекомендации.

1. Для реализации предиката number/3 ознакомиться с помощью справки или литературных источников с использованием предиката is.

2. В некоторых заданиях для их выполнения необходимо описать вспомогательные предикаты.

3. Операция сравнения «меньше или равно» записывается как “=<”.

Лабораторная работа №4.
Построение рекурсивных программ.
(5 баллов)

Задание.

Запрограммировать на языке Пролог указанные ниже предикаты. Продемонстрировать различные варианты их использования на примерах.

В отчёте по лабораторной работе продемонстрировать для каждого предиката использование нисходящей методологии разработки и дать декларативную интерпретацию.

1. `div(X,Y,Z)` ← Z – результат целочисленного деления X на Y, где X и Y – натуральные числа, представленные в структурированной форме (например, `div(s(s(s(s(s(0))))),s(s(0)),s(s(0)))`);
2. `gcd(X,Y,Gcd)` ← Gcd – наибольший общий делитель натуральных чисел X и Y;
   процедура должна быть реализована без использования операции деления по модулю;
   при реализации можно использовать по желанию либо структурированное представление (см. пример в п.1), либо обычное представление натуральных чисел.
3. `adjacent(X,Y,Zs)` ← X и Y являются соседними элементами в списке Zs;
   процедура должна быть рекурсивной и не использовать предикат append/3;
4. `last(X,Xs)` ← X является последним элементом списка Xs;
   процедура должна быть рекурсивной и не использовать предикат append/3;
5. `double(Xs,XsXs)` ← XsXs – это список Xs, в котором каждый элемент повторён дважды (например, `double([1,1,5,3,5],[1,1,1,1,5,5,3,3,5,5])`);
6. `sum(Ns,Sum)` ← Sum – сумма элементов списка натуральных чисел Ns;
   при реализации использовать структурированное представление натуральных чисел (см. пример в п.1) и не использовать вспомогательные предикаты;
7. `substitute(X,Y,L1,L2)` ← L2 – это список L1, в котором все вхождения элемента X заменены элементом Y.

Рекомендации.

1. Для реализации предикатов, работающих со структурированным представлением натуральных чисел необходимо предварительно описать предикаты *nat_num/1*, *plus/3* и *ll(X,Y)*. Последний из них истинен, если X<Y.

2. Реализация предиката *gcd/3* основана на многократном вычитании меньшего числа из большего, пока два числа не станут равными).

Лабораторная работа №5.
Множественная рекурсия на графах.
(5 баллов)

Задание.

Запрограммировать на языке Пролог указанные ниже предикаты. Продемонстрировать различные варианты их использования на примерах.

Для каждой из групп предикатов А и Б построить дерево вывода для произвольно выбранного основного вопроса.

А. Предикаты работы с бинарными деревьями

Аргументы T, T1 и T2 обозначают деревья, представляемые в виде термов, записываемых с помощью тернарного функтора `tree (левое поддерево, правое поддерево, метка)` и константы nil, например:

```
tree (tree (nil, nil, f), tree (tree(nil, nil, p), tree (nil, nil, r), k))
```

представляет дерево, изображенное на рисунке 1. В вершинах дерева могут находиться объекты произвольного скалярного типа (в приведенном примере – символы).

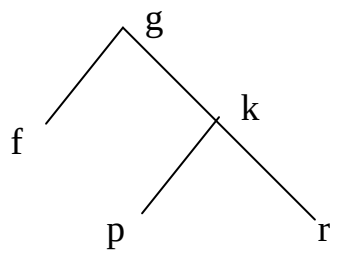

Рисунок 1

1. `tree_depth(T,N)` ← `N` – глубина дерева (т.е. количество ребер в самой длинной ветви дерева);
2. `sub_tree(T1,T2)` ← дерево `T1` является непустым поддеревом дерева `T2`;
3. `flatten_tree(T,L)` ← `L` – список меток всех узлов дерева `T`;
4. `insert(T1,N,T2)` ← `T2` – дерево, полученное путем добавления натурального числа `N` в упорядоченное дерево `T1` с учётом упорядоченности (упорядоченное бинарное дерево – дерево, в котором для каждого узла все элементы левого поддерева меньше, а все элементы правого поддерева больше значения элемента в этом узле). Натуральные числа представлять в обычном виде (1,2,3,…).

Б. Предикаты для работы с графами

Граф может быть представлен либо явно – в виде одной структуры, либо неявно – набором фактов вида `edge(P,R,N)`, устанавливающих наличие ребра (дуги) между вершинами (узлами) `P` и `R` и стоимость `N` (целое неотрицательное число) этого ребра. При явном способе задания графа он представляется термом, включающим в свой состав список ребер (заданных аналогично с помощью тернарного функтора edge) и, возможно, список вершин графа.

Граф, изображенный на рисунке 2, может быть задан неявно фактами `edge(a,c,8)`, `edge(a,b,3)`, `edge(c,d,12)`, `edge(b,d,0)`, `edge(e,d,9)`, а в явной форме – например, термом `graph([edge(a,c,8), edge(a,b,3), edge(c,d,12), edge(b,d,0), edge(e,d,9)], [a,b,c,d,e])`, где `graph` – бинарный функтор.

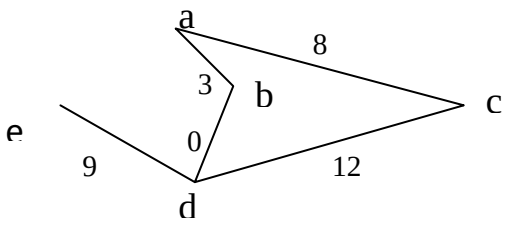

Рисунок 2

В нижеследующем описании предикатов используется первый (неявный) способ задания графа, X и Y обозначают вершины графа, а L – список вершин.

5. `path(X,Y,L)` ← `L` – путь без петель между вершинами `X` и `Y`, т.е. список вершин между этими вершинами;
6. `min_path(X,Y,L)` ← `L` – путь между вершинами `X` и `Y`, имеющий минимальную стоимость (стоимость пути равна сумме стоимостей входящих в него ребер);
7. `short_path(X,Y,L)` ← `L` – самый короткий путь между вершинами `X` и `Y` (длина пути равна количеству ребер, входящих в него);
8. `cyclic` ← граф является циклическим (т.е. не является деревом);
9. `is_connected` ← граф является связным (т.е. для любых двух его вершин существует связывающий их путь).

Замечания:
1) В предикатах 5-8 граф предполагается связным и может содержать циклы;
2) Все вышеописанные предикаты работы с графами для второго способа представления графа должны содержать дополнительный аргумент – сам граф, например: `path(G,X,Y,L)`, где `G` – структура, представляющая граф.

Лабораторная работа №6.
Работа с символьными выражениями.
(5 баллов)

Задание.

1. Написать программу, определяющую, в нормальной ли форме задана арифметическая сумма, т.е. имеет ли она вид A+B, где A – константа, а B – сумма в нормальной форме.
2. Написать определение типа «булева формула».
3. Написать программу, распознающую логические формулы в конъюнктивной нормальной форме, т.е. формулы, являющиеся конъюнкцией дизъюнкций литералов, где литерал – атомарная формула или её отрицание.
4. Написать программу, задающую отношение *negation_inwards(F1,F2)*, которое выполнено, если логическая формула F2 получается из логической формулы F1 внесением всех операторов отрицания внутрь конъюнкций и дизъюнкций.
5. Написать программу приведения логической формулы к конъюнктивному нормальному виду, т.е. к конъюнкции дизъюнкций литералов.

Лабораторная работа №7.
Интерфейс приложений Visual Prolog.
(3 балла)

Задание.

Спроектировать и реализовать средствами Visual Prolog интерфейс оконного приложения для работы со списками.

Функциональные требования.

Приложение должно обеспечить возможность выполнения следующих действий:

1) отображение содержимого списка;
2) добавление элемента в список;
3) удаление элемента из списка;
4) сортировка списка в прямом порядке;
5) сортировка списка в обратном порядке.

Отчёт по лабораторной работе не оформляется. Результаты работы демонстрируются преподавателю на компьютере.

Рекомендации и дополнительные требования.

1. Отображение списка можно реализовать в окне, открываемом при выборе пункта меню «Список|Показать». Для отображения списка в конструкторе форм следует использовать элемент List Box.

2. Возможности добавления и удаления элементов списка должны быть доступны как через меню («Список|Добавить элемент», «Список|Удалить элемент»), так и посредством интерфейсных элементов в окне отображения списка.

3. Возможности сортировки списка должны быть доступны в окне отображения списка. Доступность этих действий через меню не обязательна.

4. При попытках выполнить операции со списком пользователю должно выдаваться сообщение о недоступности данной операции на текущий момент. Непосредственная реализация действий со списками будет выполнена в рамках другой лабораторной работы.

5. В рамках лабораторной работы должны быть использованы процедуры из класса vpiCommonDialogs: note/1, note/2, error/2, messageBox/6.

Лабораторная работа №8 (4 часа).
Объектно-ориентированное программирование в Visual Prolog.
(7 баллов)

Задание.

На основе результатов лабораторной работы №7 реализовать средствами Visual Prolog работу со списком абонентов телефонной компании.

Функциональные требования.

Приложение должно обеспечить возможность выполнения следующих действий:

1) отображение списка абонентов и их телефонных номеров;
2) добавление абонента;
3) удаление абонента;
4) сортировка абонентов по фамилиям в прямом и обратном порядке;
5) сортировка абонентов по номерам телефонов в прямом и обратном порядке.

При разработке приложения для хранения информации об абонентах использовать средства и принципы реализации объектно-ориентированного подхода Visual Prolog.

Рекомендации и дополнительные требования.

1. Интерфейс, созданный в рамках лабораторной работы №7, можно использовать как основу. Однако, очевидно, что он должен быть модифицирован. Возможна также разработка интерфейса «с нуля».

2. При выполнении работы следует ориентироваться на способ работы со списком людей, описанный в презентации Lecture_9.ppt.

3. Номера телефонов допустимо представлять целочисленными значениями (например, 9871234567). При этом необходимо описать и использовать домен phoneNumber, ограничивающий допустимые значения телефонных номеров.

Лабораторная работа №9.
Стандартные предикаты работы со списками в Visual Prolog.
(5 баллов)

Задание.

Выполнить задание лабораторной работы №8 с использованием процедур стандартного класса list.

Дополнить приложение функциями сохранения списка абонентов в файл и загрузки его из файла.

Лабораторная работа №10.
Использование отсечения.
(10 баллов)

Задание.

Запрограммировать на языке Пролог указанные ниже предикаты. Продемонстрировать различные варианты их использования на примерах. Реализацию выполнить в виде консольного приложения, созданного средствами Visual Prolog. Для предикатов 1–7 обязательным требованием является использование отсечения (кроме предиката next_month).

1. subset (S1, S2) ← множество S1 является подмножеством S2;
2. union (S1, S2, M3) ← множество M3 – объединение множеств S1 и S2;
3. intersection (S1, S2, M3) ← M3 – пересечение S1 и S2;
4. subtraction (S1, S2, M3) ← M3 – разность S1 и S2.
5. equal (S1, S2) ← множества S1 и S2 эквивалентны
6. different (S1, S2) ← множества S1 и S2 являются разъединёнными, т.е. не имеют одинаковых элементов.

7. next_month(M1,M2) ← M1 является следующим месяцем по отношению к M2
   earlier_month(M1,M2) ← M1 является более ранним месяцем по отношению к M2

8. test_dialog() ← предикат, реализующий диалог с пользователем через командную строку «в стиле DOS»: программа предлагает пользователю номер варианта, пользователь вводит выбранный номер, после чего выполняются соответствующие действия и возврат к диалогу; выход из диалога является одним из вариантов действий; остальные варианты соответствуют тестам вызовов предикатов 1-7 (для пункта 7 тестируется предикат earlier_month/2).

Рекомендации и дополнительные требования.

1. Вызов предиката test_dialog() осуществляется в предикате run/0 файла main.pro.
2. Использование отсечения в предикате 8 обязательным не является, но может оказаться полезным и даже необходимым.
3. Множества представляются в виде списков с неповторяющимися значениями. Тип этих значений произвольный.
4. Предикат next_month/2 должен позволять утвердительно отвечать на вопрос next_month(“Январь”, “Декабрь”), т.е. подтверждать, что январь следует в календаре за декабрём.
5. При реализации предиката earlier_month/2 необходимо предотвратить эффект «зацикливания», т.е. на вопрос earlier_month(“Январь”, “Декабрь”)? ответ должен быть утвердительным (январь – более ранний месяц, чем декабрь), а на вопрос earlier_month(“Декабрь”, “Январь”)? решений быть не должно. То есть понятие «более ранний месяц» применять только в рамках календарного года. Для реализации данного требования рекомендуется использовать комбинацию «отсечение-fail».
6. Предикат earlier_month/2 должен давать корректные ответы на все возможные для данного предиката вопросы (как основные, так и неосновные). Для реализации данного требования рекомендуется использовать встроенный предикат free/1, который доказан, если его аргументом является переменная, не имеющая значения во время унификации предиката free/1. Например, предикат
   test(A):-free(A).
   на вопрос test(2)? даст результат “No solutions”, а на вопрос test(X)? даст положительный ответ.