Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение «Томский государственный университет систем управления и радиоэлектроники»

(ТУСУР)

УТВЕРЖДАЮ

Заведующий кафедрой ЭМИС

_____ И.Г. Боровской

«____» _____ 2010 г.

Методическое пособие к лабораторным работам и самостоятельной работе студентов по дисциплине «Представление знаний в информационных системах»

Составлено кафедрой экономической математики, информатики и статистики

Для студентов, обучающихся по направлению подготовки 230200 – Информационные системы

Форма обучения – очная

Составитель:

Должность, место работы _____ А.А. Матолыгин

«____» _____ 2010 г.

Томск 2010 г.

ОГЛАВЛЕНИЕ

Аннотация

Методические указания к самостоятельной работе студентов составлены в соответствии с требованиями государственного образовательного стандарта высшего профессионального образования по направлению подготовки 230200 – Информационные системы.

.

Методические указания направлены на приобретение студентами необходимых навыков программирования в Visual Prolog.

Лабораторная работа 1
Создание проекта в Visual Prolog

Цель работы: Знакомство с интерфейсом Visual Prolog, получение навыков создания новых проектов в Visual Prolog.

Создать новый пустой проект просто. Выберите команду *Project/New* из панели задач, как показано на рис. 1. Затем, заполните диалоговое окно *Project Settings* как на рис. 2. Нажмите кнопку *Create,* и перед вами появится окно дерева проекта (рис. 3).

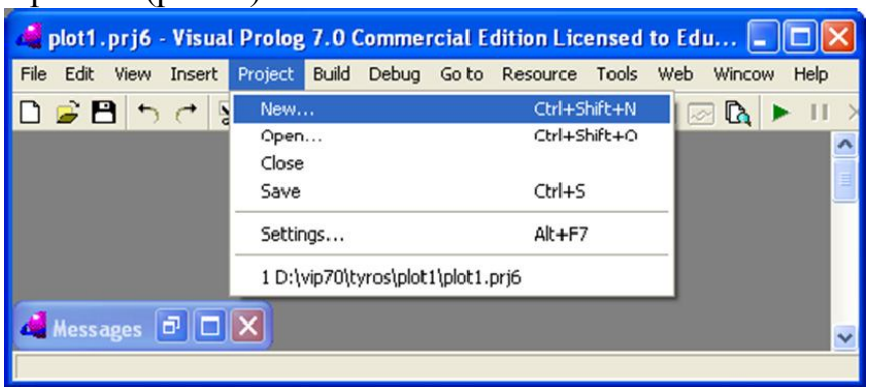

Рис. 1

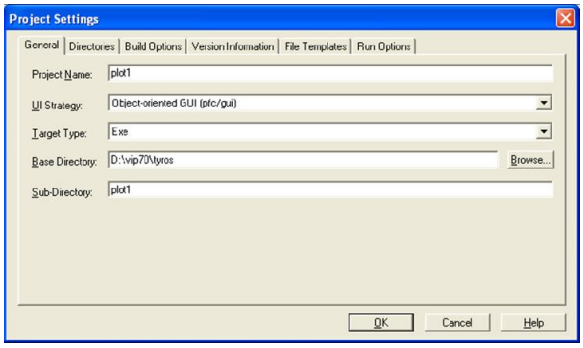

Рис. 2

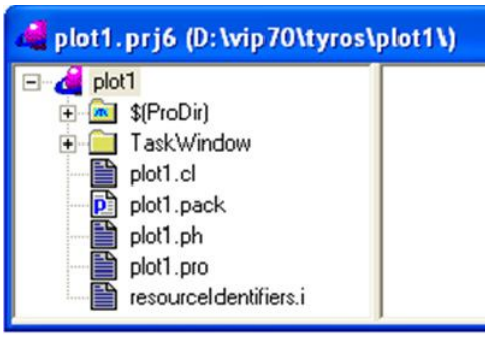

Рис. 3

Чтобы откомпилировать программу необходимо выбрать команду *Build/Build* из панели задач (рис. 4). Для запуска программы нужно выбрать *Build/Execute* из панели задач, и на экране появится окно, похожее на изображенное на рис. 5.

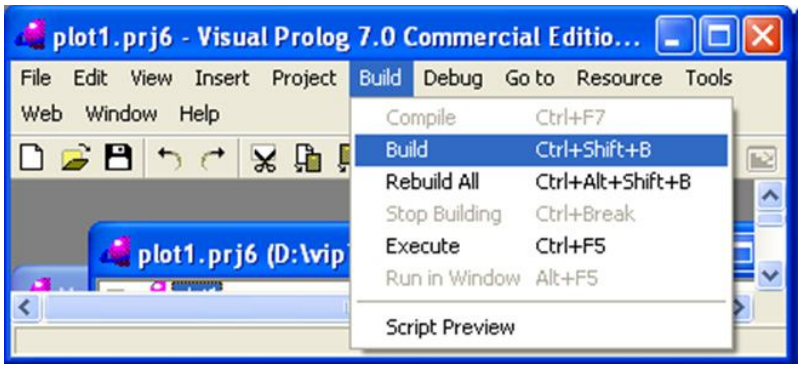

Рис. 4

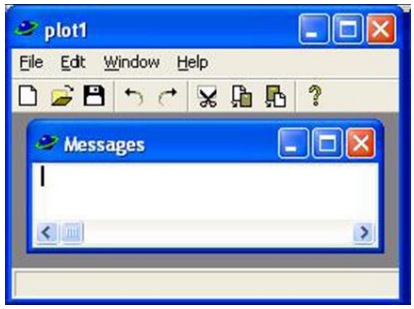

Рис. 5

Задание

1. Создайте пустой проект в Visual Prolog.
2. Откомпилируйте и запустите программу, предложенную преподавателем, в этом проекте.

Лабораторная работа 2
Формы в Visual Prolog

Цель работы: получение практических навыков добавления функциональности к пустому проекту в Visual Prolog.

Чтобы создать форму, выберите команду *File/New* из панели задач, как на рис. 1. Выберите на левой панели пункт *Form* и заполните диалоговое окно *Create Project Item* как на рис. 2. Название новой формы – *query*. Удостоверьтесь, что вы поместили форму в корне дерева проекта. Это произойдет, если вы выбрали корневой каталог в дереве проекта, прежде чем нажать *File/New*.

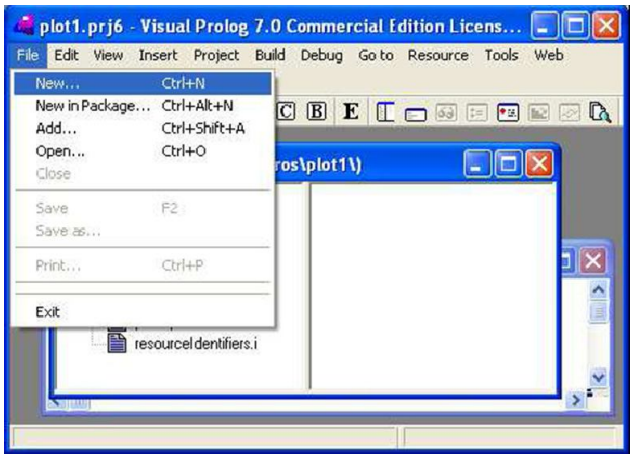

Рис. 1

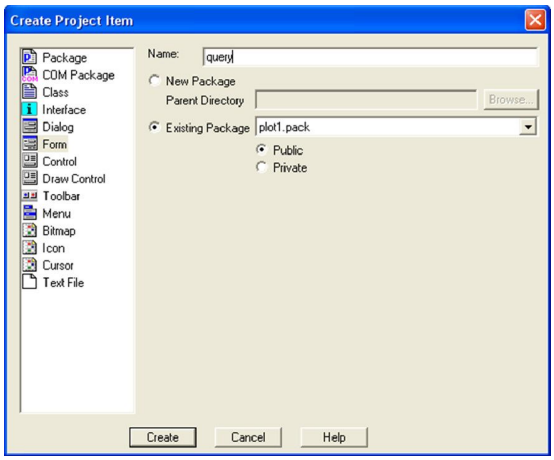

Рис. 2

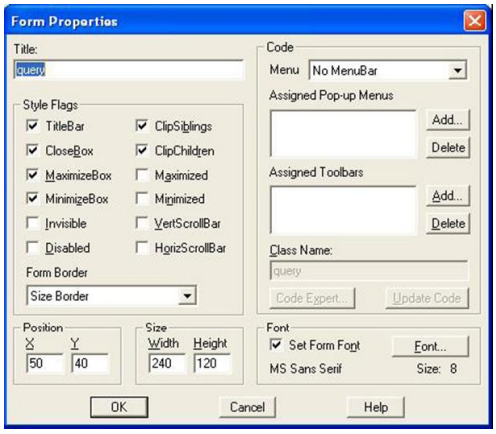

Рис. 3

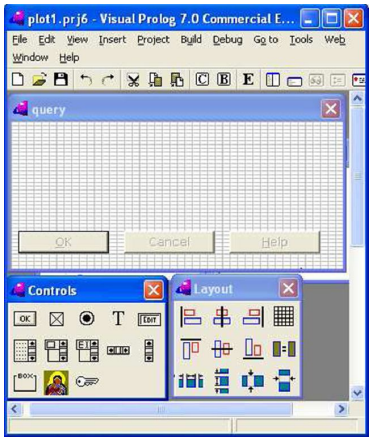

Рис. 4

Когда вы запрашиваете новую форму, Visual Prolog показывает вам диалоговое окно под названием *Form Properties* (рис. 3). Вы можете принять

установки по умолчанию в этом диалоговом окне. Далее появиться прототип новой формы (рис. 4). Вы можете изменить размеры прямоугольника окна, сделав его немного больше прототипа. Чтобы сделать это, нажмите мышкой на нижнем правом углу формы и тяните его, как вы делаете, когда изменяете размеры обычного окна.

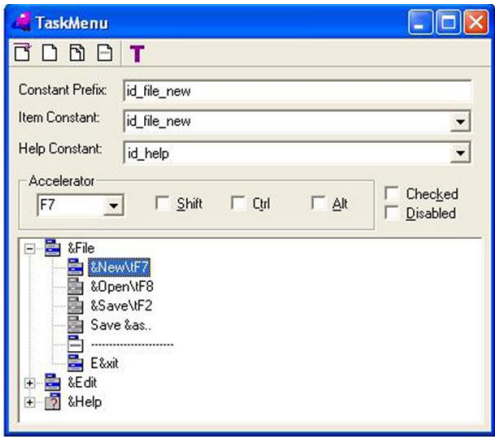

Рис. 5

При запуске пустого приложения пункт меню *File/New* отключен. Чтобы включить его, дважды щелкните по ветке дерева проекта под названием *TaskMenu.mnu*. Затем, разверните дерево, находящееся в нижней части диалогового окна *TaskMenu*, и уберите галочку *Disabled*, относящуюся к пункту *&New/tF7*, как на рис. 5.

Чтобы добавить код к пункту *File/New*, нажмите на ветке дерева проекта *TaskWindow.win* правой кнопкой мыши, которая откроет контекстное меню. Выберите пункт *Code Expert* (рис 6). Как на рис. 7, дважды щелкните на

Наконец, нажмите кнопку *Add* (ориентируйтесь по рис. 7). Это откроет текстовый редактор, со следующим фрагментом:

```
clauses
    onFileNew(_Source, _MenuTag).
```

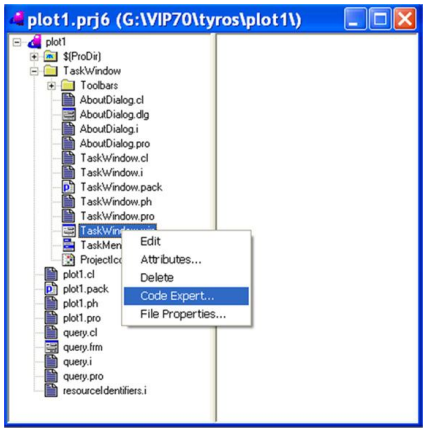

Рис. 6

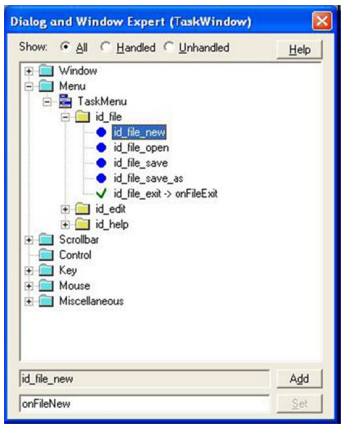

Рис. 7

Откомпилируйте программу, затем измените этот фрагмент так:

```
clauses
onFileNew(W, _MenuTag) :-
X := query::new(W),
```

X:show().

Откомпилируйте программу снова, выбрав пункт *Build/Build* на панели задач. Запустите программу, и вы увидите, что, когда вы нажимаете *File/New*, создается новая форма.

Задание

Создайте новую форму выбором пункта *File/New и* заполните диалоговое окно *Create Project Items*.

Лабораторная работа 3

Дерево проекта. Панель задач

Цель работы: Получение практических навыков работы с файлами и ресурсами.

Диалоговое окно создания нового проекта имеет шесть форм: *General, Directories, Build Options, Version Information, File Templates* и *Run Options*. В большинстве случаев заполняется только форма *General*.

General
Project Name: plot2
UI Strategy: Object-oriented GUI (pfc/gui)
Target Type: Exe
Base Directory: E:\VIP7\Tyros

Когда вы найдете шаг, указывающий: *Создайте новый GUI проект: plot2*, то вам следует войти в диалоговое окно *Project Settings* (выбрав пункт *Project/New* на панели задач и заполнить форму *General* как указано выше.

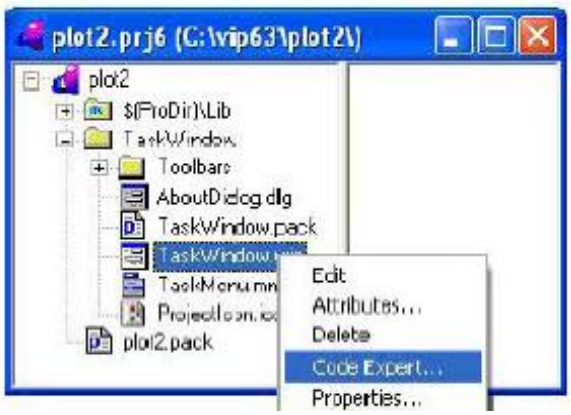

Рис. 1.

Самый лёгкий способ ориентироваться в файлах и ресурсах это щелкать мышкой по соответствующим элементам Дерева проекта: Если вы дважды щёлкнете по папке, она откроется и отобразит свое содержимое. Если вы щёлкнете по элементу правой кнопкой мыши, откроется контекстное меню, как на рис. 1. Для того, чтобы добавить фрагмент кода в *TaskWindow.win*, необходимо обратиться в дерево проекта и сделайть следующее:

1. Дважды щелкните по папке *TaskWindow*, чтобы открыть её, если она закрыта.

2. Щелкните правой кнопкой мыши по ветке дерева проекта *TaskWindow.win*, чтобы раскрыть контекстное меню, в котором будут следующие пункты:

Edit
Attribute
Delete
Code Expert

3. Наконец, выберите пункт *Code Expert.*

Эксперт кода тоже имеет форму дерева. Соответствуя названию, эксперт кода используется для вставки кода во многие файлы проекта. Чтобы добраться до эксперта кода, вы должны щелкнуть правой кнопкой мыши по элементу дерева проекта, к которому вы хотите добавить код. Затем, выберите пункт *Code Expert* из контекстного меню. Чтобы перемещаться по дереву эксперта кода и добираться до точек, куда вы желаете вставить код, щелкайте по нужным веткам дерева. Если вы хотите, чтобы эксперт кода добавил прототип к «листку» дерева, нажмите на этот листок и затем нажмите на кнопку *Add*, которая появится внизу диалогового окна. Затем дважды щелкните по листку снова, чтобы перейти к коду.

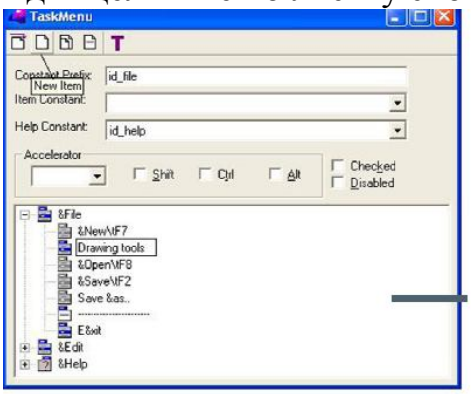

Рис. 2

Для добавления нового элемента к дереву проекта выберите команду *File/New* из панели задач. Следите за тем, чтобы размещать создаваемые элементы в нужных папках.

Если дважды щёлкнуть по *ProjectTree/TaskMenu.mnu,* то получим диалоговое окно. Можно развернуть дерево спецификации меню, щелкая по его ветвям. Можно создать новый пункт меню (рис. 2).

Задание

1. Создайте новый проект: factorial.
2. Добавьте новый пакет к дереву проекта: factorial/forms.
3. Создайте новую форму: forms/query. Добавьте edit field [строку ввода] (с названием edit_ctl) и кнопку (с названием factorial_ctl) на неё.
4. Включите пункт TaskMenu File/New.
5. Откомпилируйте приложение

Лабораторная работа 4

Класс. Создание класса.

Цель работы: Получение практических навыков создания новых классов.

Чтобы создать новый класс и вложить его, например, в пакет *forms*, выберите папку *forms* в дереве проекта и выберите пункт *File/New* в панели

задач VDE. Убедитесь, что убрали галочку *Create Objects*, как показано на рис. 1.

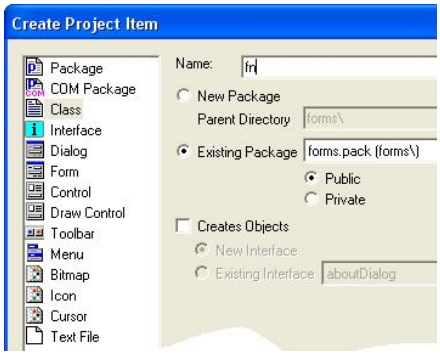

Рис. 1

На языке объектно-ориентированного программирования, файл ***fn.cl*** содержит интерфейс методов ***setVal:(string) procedure*** и ***calculate:().*** Он информирует предполагаемых пользователей о том, что метод ***setVal/1*** это процедура (об этом позже), и требует один аргумент, который должен быть строкой; он также говорит, что метод ***calculate/0*** не имеет аргументов. С другой стороны, файл ***fn.pro*** содержит определение обеих процедур.

```
%файл fn.pro
implement fn
      open core
constants
      className = “forms/fn”.
      classVersion = “”.
class facts
      nVal:integer := 0.
class predicates
      fact:(integer, integer) procedure (i,o).
clauses
      classInfo(className, classVersion).
setVal(X) :-
      nVal:=toterm(X),
      stdio::write(“fact(“, nVal, “)=”).
fact(0,1) :- !.
fact(N, N*F) :-
      fact(N-1, F).
calculate() :-
      fact(nVal, F),
      stdio::write(F, “\n”).
end implement fn
```

Следующий шаг – вызвать предикат **setVal/1**, когда пользователь введёт что-нибудь в строку ввода. Отправляйтесь в дерево проекта и примените эксперт кода к ***query.frm***. В эксперте кода откройте папку *Control/edit_ctl* (дважды щелкнув на ней), щелкните на *LoseFocus*, и нажмите кнопку *Add*, чтобы создать прототип кода. Дважды щелкните на *LoseFocus→onEditLoseFocus*, и добавьте следующий фрагмент кода:

```
predicates
      onEditLoseFocus : window::loseFocusListener.
clauses
      onEditLoseFocus(S) :-
            fn::setVal(S:getText()).
```

В Visual Prolog есть классы и объекты, принадлежащие классам. Объект имеет набор методов, которые являются программами, которые работают с объектом. Они также называются *event driven* predicates – движимые событиями предикаты. Если вы поставите курсор в строку ввода,

она получит фокус. Строка ввода потеряет фокус, если вы обратите свое внимание на другой объект, например, кнопку.

Задание

В новом приложении выберите пункт *File/New*. Создайте форму с кнопками и полями. Сделайте так, чтобы класс **fn** написал в окне сообщений значение, заданное преподавателем.

Лабораторная работа 5

Функции. Предикаты.

Цель работы: Получение практических навыков работы со сложными термами и списками.

Предикаты это функции, чьим образом является множество **{verum, falsum}**, или, если вам не нравятся латинские названия, используемые в логике, вы всегда можете полагаться на английский эквивалент: **{true, false}**. Есть несколько предикатов, известных любому. Вот они:

X>Y возвращает **true** если X больше, чем Y, иначе возвращает **false**.

X<Y возвращает **true** если X меньше, чем Y, иначе возвращает **false**.

X=Y возвращает **true** если X равен Y, иначе возвращает **false**.

Предикат с одним аргументом говорит о свойстве или особенности его аргумента. Можно сказать, что такой предикат работает, как прилагательное. В C, **~X true**, если X **false**, иначе **~X false**.

Предикат более чем с одним аргументом показывает, что между его аргументами существует отношение. В случае **X=Y** это отношение – равенство.

Предположим, что у вас есть предикат **city(Name, Point)**, который определяет координаты города на карте. Предикат **city/2** имеет домен1

city : (string Name, pnt Position).

и может быть определен как база данных фактов:

city(“Salt Lake”, pnt(30, 40)).
city(“Logan”, pnt(100, 120)).
city(“Provo”, pnt(100, 200)).
city(“Yellowstone”, pnt(200, 100)).

Этот предикат проверяет, является ли заданное положение заданного города верным, когда кто-либо не уверен насчет этого. Вот несколько запросов, которые можно задать предикату **city/2**.

city("Salt Lake", pnt(30, 40)) → true
city("Logan", pnt(100, 200)) → false
city("Provo", pnt(100, 200)) → true

Нет вопросов, которым вы можете найти применение для подобного предиката. Хотя, предикат, возвращающий координаты города по заданному названию, может оказаться намного более полезным.

city(“Salt Lake”, P) → P=pnt(30, 40).

В этом новом виде предикатов, символы1, начинающиеся с заглавной буквы, называются переменными.

Задание

Создайте проект. Далее создайте предикат, который:

1. Заменяет в исходном списке первое вхождение заданного значения другим.
2. Заменяет в исходном списке все вхождения заданного значения другим.
3. Порождает по заданному натуральному числу N список, состоящий из натуральных чисел от 1 до N (по возрастанию).
4. Порождает по заданному натуральному числу N список, состоящий из натуральных чисел от N до 1 (по убыванию).
5. Порождает по заданному натуральному числу N список, состоящий из N случайных натуральных чисел из промежутка от 1 до 100.
6. Порождает по заданным числам N, M, K список, состоящий из N случайных натуральных чисел из промежутка от M до K.
7. Удваиваевает значения элементов списка.

Лабораторная работа 6

Предикаты с множественными решениями.

Цель работы: Получение практических навыков создания программ, использующих предикаты с множественными решениями.

Мы знакомы с предикатами, которые использовали свои переменные для получения решения, а не для проверки истинно отношение или ложно. Тем не менее, существуют ситуации, требующие более одного решения. Пусть ***conn/2*** будет предикатом, устанавливающим связь между двумя городами.

```
conn(pnt(30, 40), pnt(100, 120)).
conn(pnt(100, 120), pnt(100, 200)).
conn(pnt(30, 40), pnt(200, 100)).
.
.
.
```

Вы можете использовать его для получения всех соединений между городами, как показано в примере:

```
conn(pnt(30, 40), W). → W=pnt(100, 120)
→ W=pnt(200, 100)
conn(X, Y). → X=pnt(30, 40) / Y=pnt(100, 120)
→ X=pnt(100, 120) / Y=pnt(100, 200)
→ X=pnt(30, 40) / Y=pnt(200, 100)
```

Рассмотрим запрос: *conn(pnt(30, 40), W)?*

Ответом может быть ***W=pnt(100, 120)***, но им также может быть ***W=pnt(200, 100)***.

Задание

Создайте программу, результатом выполнения которой будет определение координат города на карте.

Лабораторная работа 7

Предикаты рисования.

Цель работы: Получение практических навыков работы в создании предикатов рисования.

Предикаты рисования требуют дескриптор (*handle1)* окна, которое будет поддерживать рисунки и чертежи. Вот как вы можете получить дескриптор:

```
clauses
onPaint(S, _Rectangle, _GDIObject) :-
W= S:getVPIWindow(), draw::drawThem(W).
```

Внутри класса ***draw*** будет передаваться дескриптор ***W***. Например, в предложении

drawThem(Win) :- connections(Win), drawCities(Win).

он передается в ***connections/1***, где он является первым аргументом ***drawLine/3****:*

connections(Win) :- conn(P1, P2), drawLine(Win, P1, P2), fail.

Предикат ***drawLine(Win, P1, P1)*** чертит линию из ***P1*** в ***P2*** на окне ***Win***. Как вы знаете, ***P1*** и ***P2*** – точки, как ***pnt(10, 20)***. Иной предикат, с которым вы знакомы, это ***drawEllipse(W, rct(X1, Y1, X2, Y2)),*** который чертит эллипс на окне ***W***. Эллипс вписан в прямоугольник ***rxt(X1, Y1, X2, Y2***), где ***X1, Y1*** – координаты верхнего левого угла, и ***X2, Y2*** – координаты нижнего правого угла.

Задание

Создайте программу, которая рисует изображение городов Томска, Новосибирска, Красноярска по нажатию определенной кнопки.

Лабораторная работа 8

Списки. Строки. Операции со строками.

Цель работы: Получение практических навыков выполнения операций со строками, работы со списками.

Список это упорядоченная последовательность элементов, где ***упорядоченная*** означает, что порядок имеет значение. В Прологе список помещается между квадратными скобками, и подразумевается имеющим голову (*head*) и хвост (*tail*).

Важные схемы программирования списков:

• ***Редукция***. Схема, используемая для вычисления суммы элементов списка, называется *редукцией*, потому что она сокращает размерность входных данных1. Фактически, списки можно считать одномерными данными, а сумму – величиной нулевой размерности. Есть два способа выполнения сокращения: рекурсивный и итеративный. Вот рекурсивная редукция суммы:

```
%Рекурсивная редукция
class predicates
sum:(real*, real) procedure (i, o).
clauses
sum([], 0) :- !.
sum([X|Xs], S+X) :- sum(Xs, S).
```

Термин ***итеративный*** восходит к латинскому слову *iterum*, которое означает ***снова***. Вы также можете вообразить, что он восходит к *iter/itineris* – ***путь, дорога***. Итеративная программа проходит через код снова и снова.

```
%Итеративная редукция
add([], A, A) :- !.
add([X|Xs], A, S) :- add(Xs, X+A, S).
sum(Xs, S) :- add(Xs, 0.0, S).
```

• ***«Молния».*** Это устройство, используемое в сплошной застежке для одежды, изобретенной канадским инженером Гидеоном Сандбэком (*Gideon Sundback*). Это устройство сводит с ума, когда не работает, и является причиной такого количества инцидентов с мальчишками, что врачам и медсёстрам потребовалось специальное обучение, чтобы разбираться с этими устройствами.

```
class predicates
dotproduct:(real*, real*, real) procedure (i, i, o).
clauses
dotproduct([], _, 0) :- !.
dotproduct(_, [], 0) :- !.
    dotproduct([X|Xs], [Y|Ys], X*Y+R) :- dotproduct(Xs,
Ys, R).
```

Задание 1

1. Создайте предикат, осуществляющий перестановку двух элементов списка с заданными номерами.

2. Создайте предикат, генерирующий все перестановки элементов списка, указанного в качестве первого аргумента предиката.

3. Создайте предикат, осуществляющий циклический сдвиг элементов списка на один влево (вправо).

4. Создайте предикат, осуществляющий циклический сдвиг элементов списка на заданное количество шагов влево (вправо).
5. Создайте предикат, осуществляющий поэлементное перемножение соответствующих элементов двух исходных списков.
6. Создайте предикат, вычисляющий скалярное произведение векторов, заданных списками целых чисел.
7. Создайте предикат, осуществляющий подсчет числа вхождений каждого элемента исходного списка. Ответом должен быть список пар, в которых первая компонента - элемент исходного списка, вторая - число его вхождений в первоначальный список.
8. Создайте предикат, определяющий первую позицию подсписка в списке.
9. Создайте предикат, добавляющий элементы одного списка во второй список, начиная с заданной позиции.
10. Создайте предикат, возвращающий по списку и двум числам M и N подсписок исходного списка, состоящий из элементов с номерами от M до N.
11. Создайте предикат, формирующий список простых чисел, не превосходящих данного числа.
12. Создайте предикат, транспонирующий матрицу, заданную списком списков.

Задание 2

1. Создайте предикат, который будет находить последнюю позицию вхождения символа в строку.
2. Создайте предикат, который подсчитает общее количество латинских букв в списке символов.
3. Создайте предикат, который будет подсчитывать количество русских гласных букв в строке.
4. Создайте предикат, который будет удалять из данной строки все вхождения заданного символа.
5. Создайте предикат, который продублирует вхождение каждого символа в строку.
6. Создайте предикат, "переворачивающий" строку (меняющий в строке порядок символов на обратный).
7. Создайте предикат, проверяющий, является ли данная строка палиндромом.
8. Создайте предикат, составляющий список символов, которые входят одновременно в обе данных строки.

ЛИТЕРАТУРА

1. Зюзьков В. М. Искусственный интеллект : учебное пособие - Томск : НТЛ, 2007. - 152 с.
2. Ходашинский И. А.Язык ПРОЛОГ в примерах и задачах : Учебное пособие для вузов /. Томск : ТУСУР, 2006. - 279[1] с. : ил. - **ISBN** 5-86889-291-7.
3. Ходашинский И. А. Методы искусственного интеллекта, базы знаний, экспертные системы : Учебное пособие / - Томск : ТУСУР, 2002. - 140 с.