РЕКУРСИВНО-ЛОГИЧЕСКОЕ ПРОГРАММИРОВАНИЕ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ КОНТРОЛЬНОЙ РАБОТЫ

ВВЕДЕНИЕ

Изучение дисциплины «Рекурсивно-логическое программирование» предусмотрено Государственным образовательным стандартом высшего профессионального образования, регламентирующим процесс подготовки программистов по специальности 010500.65 «Математическое обеспечение и администрирование информационных систем». В соответствии с этими же стандартами данная дисциплина должна быть обеспечена практикумом.

Представляется целесообразным в рамках контрольной работы по дисциплине «Рекурсивно-логическое программирование» изучить основы программирования на языке логического программирования Prolog в среде Visual Prolog. Задания контрольной работы будут способствовать закреплению знаний по соответствующим разделам теоретической части курса, более глубокому пониманию студентами основных вопросов программирования логики.

Задания носят теоретический и практический характер, и заключается в программировании задач на языке Prolog и ответах на вопросы.

Задания выполняются в строгой последовательности: сначала указывается условие, затем ответ. Контрольная работа выполняется в письменном виде в виде распечаток всех созданных документов. Объем контрольной работы не должен превышать 25 страниц ученической тетради или 15 печатных страниц. Работа должна быть грамотно написана, правильно оформлена. Страницы нумеруются, ставится номер варианта, подпись и дата выполнения. В конце работы указывается список используемой литературы.

Контрольную работу необходимо представить в сроки, указанные в учебном графике. Работы, не отвечающие требованиям методических указаний, не засчитываются.

ОБЩИЕ МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Контрольная работа состоит из двух заданий – первое задание состоит в ответе на теоретический вопрос, второе задание состоит в разработке собственных программ на языке Prolog в среде – Visual Prolog.

Prolog - это язык программирования для символических, нечисловых вычислений. Он особенно хорошо приспособлен для решения проблем, которые касаются объектов и отношений между объектами.

Запуск Visual Prolog осуществляется следующим образом: Пуск Программы Visual Prolog 7.3 PE Visual Prolog. Исходный вид окна приложение Visual Prolog представлен на рис. 1.

Консольное приложение – проект HelloCons.

Выберем команду Project/New в меню задач.

В нем задаем:

- имя проекта HelloCons,
- тип - console application.

Построение проекта. Нажимаем OK. ИСР строит шаблон проекта. Компиляция проекта. Теперь командой Build/Build осуществляется компиляция проекта, в дерево проекта добавляются файлы проекта без функциональности:

Для задания функциональности щелчком мыши выбираем main.pro, в шаблон main.pro нужно добавить свои операции.

Добавляем в файл main.pro функциональность. Вставляем в файл команду ожидания ввода Enter,

_=readLine().

Кроме того, во фрагменте заголовка (строка open core) открываем и класс консоли (console), что позволяет в предикате run() ссылку на консоль не упоминать.

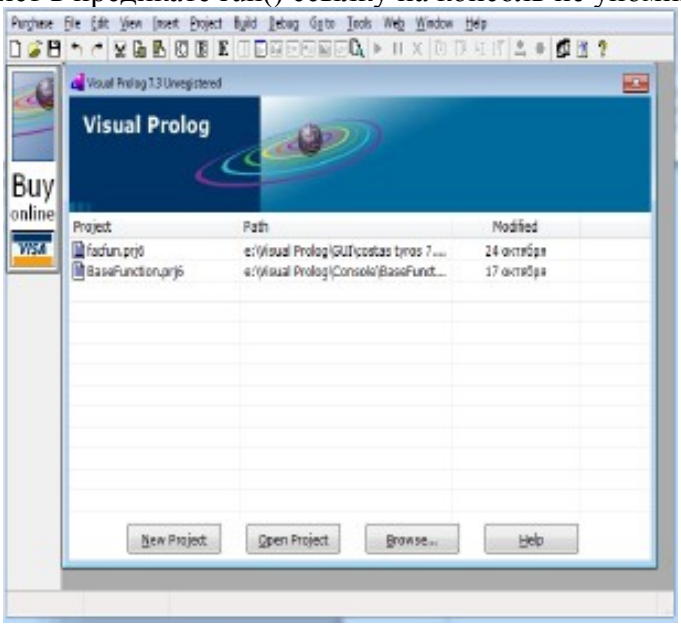

Рис.1

Листинг программы.

```
implement main
open core, console  % Открываем класс console
constants
  className = "main".
  classVersion = "".
clauses
  classInfo(className, classVersion).
clauses
  run():-
    init(),
    write("Hello World!!"),nl,nl,
    write("Нажмите Enter”),
    _=readLine().       % ожидание Enter
end implement main

goal
  mainExe::run(main::run).
```

Запуск программы.

При запуске проекта командой Build/Execute сначала осуществляется повторная компиляция, формируется и запускается исполняемый файл.
Появляется окно консоли с выведенным текстом. Для завершения нажмите Enter.

Проект GUI

Создание проекта. Выберем команду Project/New в меню задач.

Создание проекта. Выберем команду Project/New в меню задач.

В нем задаем:

- имя проекта Prohect1
- тип - GUI application..

Построение проекта. Нажимаем OK. ИСР строит шаблон проекта.

Компиляция проекта. Теперь командой Build/Build осуществляется компиляция проекта, в дерево проекта добавляются файлы проекта без функциональности:

Для задания функциональности щелчком мыши выбираем main.pro, в шаблон main.pro нужно добавить свои операции.

Построение проекта. Нажимаем OK. ИСР строит шаблон проекта. Компиляция проекта. Теперь командой Build/Build осуществляется компиляция проекта, в дерево проекта добавляются файлы проекта без функциональности (рис.2).

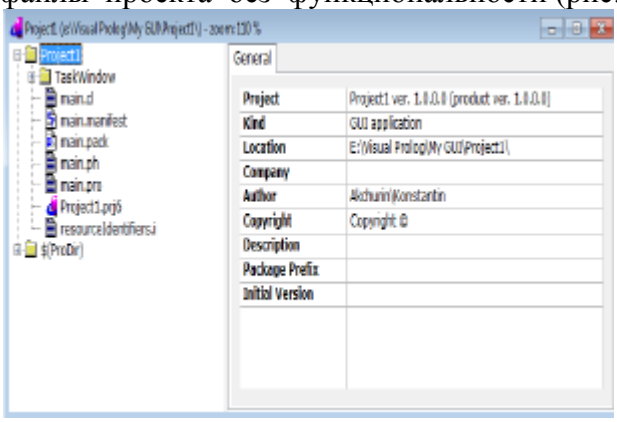

Рис.2

Запуск программы. При запуске проекта командой Build/Execute сначала осуществляется повторная компиляция, формируется и запускается исполняемый файл.

На экране появится окно, содержащее:

- Стандартное графическое окно со строками заголовка и главного меню, панелью инструментов. Меню и панель инструментов содержат компоненты по умолчанию.
- Встроенную консоль сообщений Messages (рис.3). Сюда компилятор выводит свои сообщения.

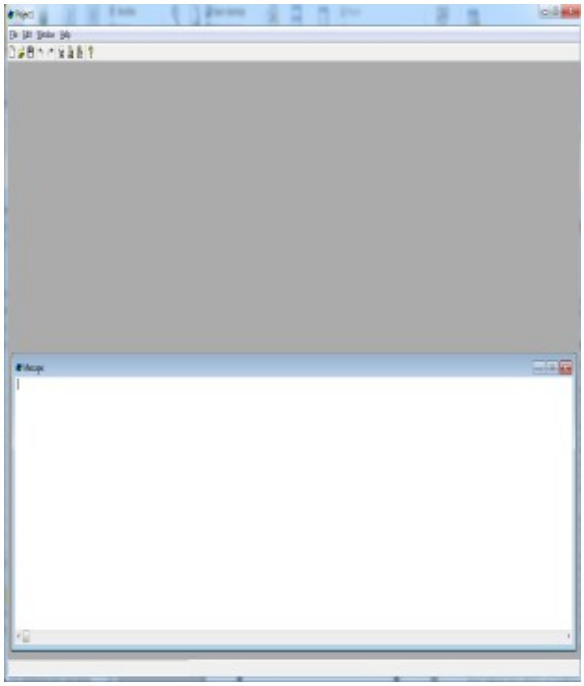

Рис.3

Чтобы выйти из программы, нужно нажать кнопку в виде крестика, которая находится в верхнем правом углу окна.

Для конкретных действий в проекте нужно создать тему. Для этого используем команду File/New in New Package (новый файл в новом пакете) (рис. 4).

Генерируется окно создания темы Create Project Item. Можно создавать темы:

- Package – встроенный пакет,
- Class – класс,
- Interface – интерфейс,
- Dialog – диалог,

- Form – форма,
- Control – элемент управления,
- Draw Control – элемент управления для рисования,
- Toolbar – панель инструментов,
- Menu – меню,
- Bitmap – растровая картинка.
- Icon – иконка,
- Cursor - курсор,
- TextFile – текстовый файл.

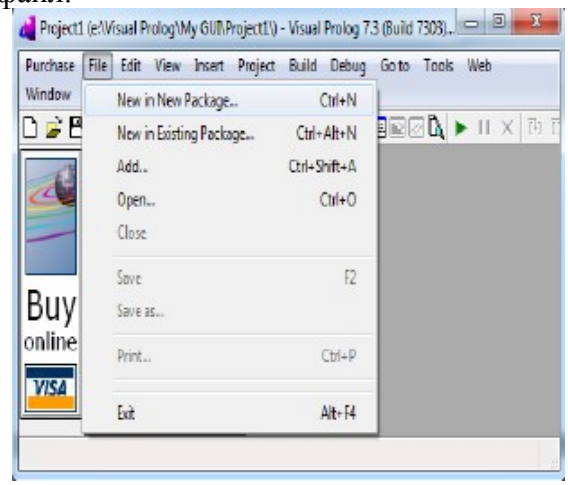

Рис.4

Например, выбираем форму с именем form1.

Нажимаем кнопку create, генерируется окно конструктора формы, которое содержит встроенные окна:

- Form1 -форма, в которую по умолчанию включены 3 часто употребляемые элементы управления (OK, Cancel, Help).
- Properties – свойства,
- Controls – доступные элементы управления, которые можно использовать в форме (путем копирования их туда),
- Layout - средства управления положением компонент в форме.

Для регулирования поведения окна формы в дереве проекта нужно открыть поле TaskWindow.win.

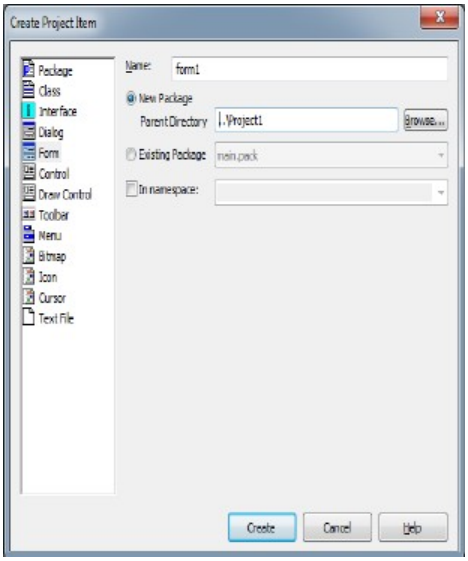

Рис.5

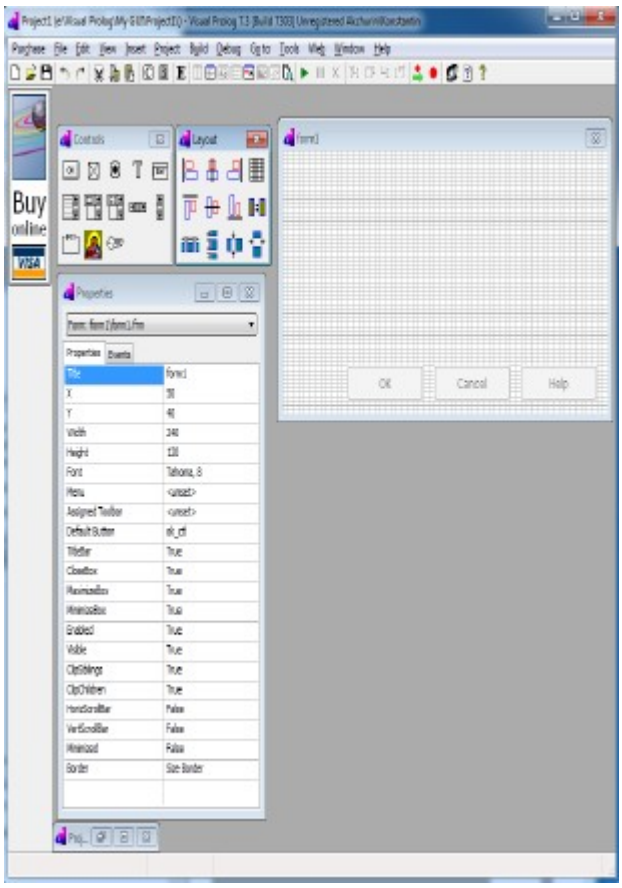

Рис. 6

Открывается окно эксперта диалога и окон Dialog and Window Expert (TaskWindow), содержащее эксперты для компонент:

- Window –окно.
- Menu – меню.
- Scrollbar
- Control – элемент управления.
- Key – клавиша.
- Mouse - мышь
- Miscellaneous – разное.

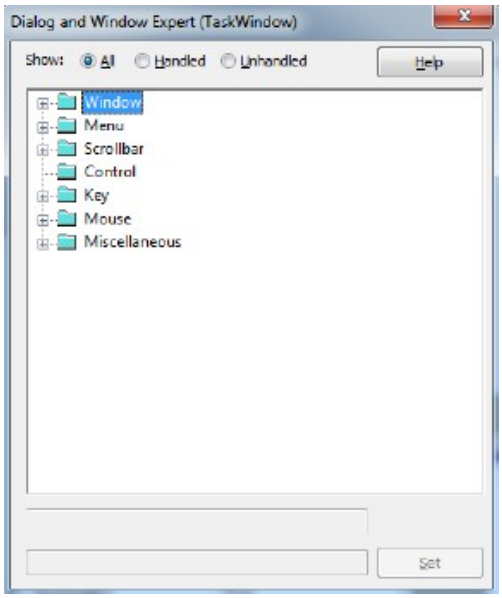

Рис.7

В нем нужно изменить установки, которые по умолчанию запрещают некоторые действия.

Для поля Windows нужно сделать доступными:

- Destroy – уничтожение,
- Show – показать,
- Size – размер.

Для поля Menu нужно сделать доступными:

- in_file_new – новый файл открыть,
- in_file_exit – файл закрыть,
- id_help_about – открыть справку.

Остальные поля менять не надо.

Поля без галочки выключены. Для включения пункта двойной щелчок по нему. Чтобы открывалось окно формы, нужно разрешить открытие файла формы (по умолчанию это запрещено). Открываем TaskMenu двойным щелчком мышью по TaskMenu.mnu.

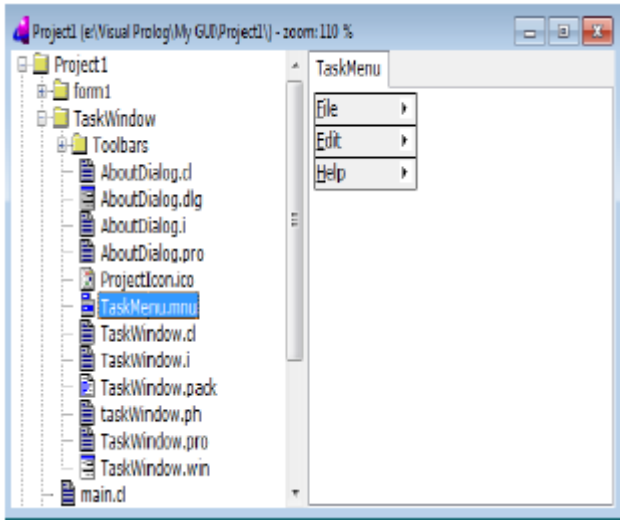

Рис. 8

Открывается окно TaskMenu, в котором для &File\>&New\F7 нужно убрать галочку в Disable (недоступно).

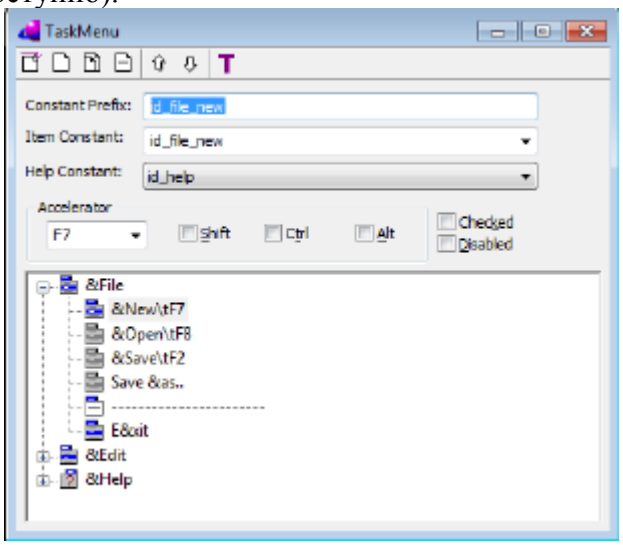

Рис. 9

При закрытии этого окна возникает запрос на сохранение сделанных изменений. Выбираем Save – сохранить.

Осталось прописать ссылку на открываемый файл. Открываем файл TaskWindow.pro двойным щелчком по TaskWindow.pro. Открывается файл TaskWindow,pro. Его содержимое создано автоматически при компиляции. Только не прописано правило реагирования на открытие нового файла – clauses onFileNew…

Рис. 10

Вносим туда правило отображения формы form1 с помощью конструктора формы.

Делаем повторную компиляцию и запуск командой Build/Execute. В результате отображается окно проекта без запущенной формы.

Команда File/New в окне проекта приводит к отображению формы в окне проекта.

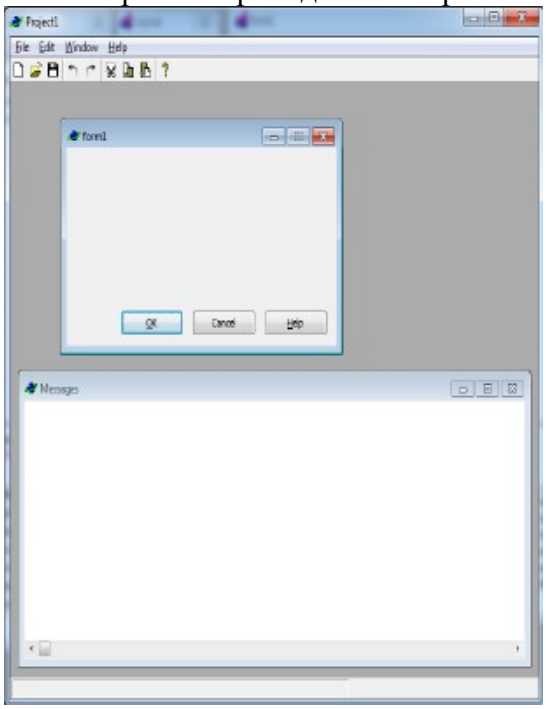

Рис.11

ТЕОРЕТИЧЕСКОЕ ВЫПОЛНЕНИЕ

1. Понятие логического программирования.
2. История возникновения логического программирования.
3. Языки логического программирования.
4. Место и роль логического программирования в подготовке специалиста.
5. Сферы применения средств логического программирования
6. Основные стратегии решения задач искусственного интеллекта в Пролог.
7. Экспертные системы
8. Тенденции и перспективы развития методов и средств логического программирования.
9. Логика предикатов. Фразы Хорна. Принцип резолюции.
10. Законы логики. Исчисление предикатов.

ПРАКТИЧЕСКОЕ ВЫПОЛНЕНИЕ

Написать программу для раскрашивания произвольной плоской карты не более чем четырьмя цветами так, чтобы никакие два соседних региона не были окрашены в один и тот же цвет.

Эту задачу решает классическая программа порождения и проверки. В предикате

test(L) вызовы
generateColor(A), generateColor(B),
generateColor(C), generateColor(D),
generateColor(E), generateColor(F),
порождают цвета для шести регионов карты. Затем test(L) строит карту в виде списка пар соседних стран:
L= [nb(A, B), nb(A, C), nb(A, E), nb(A, F),
nb(B, C), nb(B, D), nb(B, E), nb(B, F),
nb(C, D), nb(C, F), nb(C, F)]

Наконец предикат aMap(L) проверяет, является ли L допустимой картой. Она будет допустимой, если никакие из двух соседних стран не будут окрашены в один цвет. Если предикат aMap(L) является ложным, то предложенные цвета не являются правильными.

Поэтому программа откатывается к вызову generateColor(X) для получения нового набора цветов.

Задача четырёх красок интересна по двум причинам.

1. Схема порождения и проверки — это техника, которая одной из первых применялась в искусственном интеллекте.
2. Существует очень известная теорема, которая утверждает:

Любая карта может быть раскрашена четырьмя цветами так, что никакие две соседние страны не будут окрашены в один цвет.

Эта теорема была доказана в 1976 году Кеннетом Аппелем (Kenneth Appel) и Вольфгангом Хакеном (Wolfgang Haken), двумя математиками из университета Иллинойса.

```
implement main
    open core
domains
    colors= blue; yellow; red; green.
    neighbors= nb(colors, colors).
    map= neighbors*.

class predicates
    aMap : (map) nondeterm anyFlow.
    test : (map) procedure anyFlow.
```

```
    generateColor : (colors) multi (o).

clauses
    classInfo("main", "fourcolors").

    generateColor(R) :-
        R= blue; R= yellow;
        R= green; R= red.

    aMap([]).
    aMap([X|Xs]) :-
        X= nb(C1, C2), not(C1 = C2),
        aMap(Xs).

    test(L) :-
        generateColor(A), generateColor(B),
        generateColor(C), generateColor(D),
        generateColor(E), generateColor(F),
        L= [ nb(A, B), nb(A, C), nb(A, E), nb(A, F),
            nb(B, C), nb(B, D), nb(B, E), nb(B, F),
            nb(C, D), nb(C, F), nb(C, F)],
        aMap(L), !; L= [].

    run():- console::init(), test(L),
  100  100
        stdio::write(L), stdio::nl.
end implement main

goal
    mainExe::run(main::run).
```

Вопросы для самоконтроля

1. Какова структура окон в программе Visual Prolog?
2. Каким образом обрабатываются ошибки в программах Visual Prolog’a?
3. Сформулируйте отличие между атомами и переменными.
4. Какие символы можно использовать в имени переменной?
5. Что такое «правило» с точки зрения Prolog’a?
6. Перечислите требования, предъявляемые к постановке вопросов в Prolog’e.
7. Какие раздела выделяются в предложениях языка Prolog?
8. В каком разделе описываются факты? Правила? Запросы? Каковы особенности их записи?
9. В каких случаях используется запятая при формировании запроса?

СПИСОК СТУДЕНТОВ

Номер варианта	ФИО студента
1	Агутов Евгений Александрович
2	Васильчев Михаил Алексеевич
3	Еремин Юрий Владимирович
4	Кудашова Валентина Сергеевна
5	Любченко Сергей Николаевич
6	Макаров Михаил Юрьевич
7	Николаев Евгений Александрович
8	Платонов Александр Александрович
9	Смирнов Алексей Петрович