# Интерактивная визуальная среда Visual Prolog

**Общие сведения о системе Visual Prolog 5.2**

Программный продукт Visual Prolog (VIP) осуществляет полную поддержку работы с оконными системами, обеспечивая создание современных графических приложений. С использованием VIP можно разрабатывать новое функциональное приложение в короткие сроки, шаг за шагом развивая его от опытного образца до законченного коммерческого проекта.

Система VIP состоит из нескольких компонентов: интерактивной визуальной среды проектирования (VDE), которая включает текстовый и различные графические редакторы; средств генерации кода (Experts); встроенной управляющей логики и расширений языка Пролог в форме визуального интерфейса программирования (VPI). Кроме этого в состав продукта входят компилятор Пролога, различные файлы и библиотеки, компоновщик, а также файлы примеров и справки.

VDE предназначена для облегчения, удобства и быстроты разработки, проверки и изменения приложений, написанных на языке Visual Prolog. Интерфейс VPI обеспечивает доступ к ряду экспертов кода, редакторов ресурсов и дополнительных предикатов, описаний и библиотек.

Редакторы ресурсов используются для описания, размещения и редактирования окон, диалогов, меню, панелей инструментов, списков, иконок, курсоров, рисунков и контекстной справки. Далее средства генерации кода собирают эти компоненты в некотором каркасе (например, в окне), который компилируется, компонуется и в дальнейшем запускается на выполнение.

По запросу программиста средства генерации кода вызывают любую часть данного каркаса в окно редактирования для проверки, заполнения предметной логикой и получения в результате целевого приложения. В окне данного каркаса имеется множество необходимых функций многоуровневого контекстного меню, предназначенных для редактирования, выделения, поиска, замены и вставки кода. Часть кода может быть сгенерирована с использованием пунктов вставки данного меню, позволяющих выбирать цвета, элементы VPI и стандартные шаблоны предикатов Пролога, определяемые пользователем предикаты для диалогов и окон, имен доменов, идентификаторов ресурсов, констант VPI, имен файлов, ключевых слов и т. п.

Библиотека VPI содержит приблизительно 300 предикатов высокого уровня для создания графических пользовательских интерфейсов, что позволяет создавать мобильные приложения.

Широкий диапазон стандартных предикатов обеспечивает все возможности, которые присущи профессиональному языку программирования: процедуры ввода – вывода, арифметику, реализацию функций манипуляции, внешнюю систему базы данных с B+ - деревьями, поддержку сетевого взаимодействия, функции операционной системы. Система включает даже предикаты для манипуляции памятью низкого уровня, а также обработки прерывания от внешних устройств.

Программы Visual Prolog могут быть легко связаны с кодом на любом другом языке, который производит стандартные файлы объектного формата, включая Borland C, C++, Microsoft C и Ассемблер. Более того, Visual Prolog может вызывать готовые фрагменты, написанные на других языках. Готовые модули на других языках также могут вызывать предикаты Пролога, обеспечивая даже программистам, программирующим не на Прологе, доступ к мощной подсистеме логического программирования. Это позволяет использовать любой язык в тех случаях, где этот язык наилучшим образом соответствует вашим задачам.

Все процедуры, написанные на других языках программирования, могут быть объявлены в программе на Прологе как предикаты, а объявления типов данных гарантируют безопасный вызов импортируемых функций. Такие объявления позволяют также формировать заголовочные файлы для координирования работы множества программистов в многоязычном проекте.

Программы, написанные на Visual Prolog, могут как вызывать динамические DLL, так и сами быть оформлены в виде DLL.

Разрабатываемые в системе приложения не привязаны к одной операционной системе или одной графической платформе. С помощью Visual Prolog можно создавать мобильные приложения для DOS-, MS Windows и OS/2–платформ, серверных приложений для платформ SCO Unix и Linux.

Визуальная среда разработки (VDE, Visual Development Environment) объединяет компилятор с редактором, комплектом инструментальных средств разработки ресурсов, экспертами ресурсов и приложений, интерактивной утилитой построения приложений и различными средствами просмотра кода. После интерактивного визуального создания компонентов пользовательского интерфейса автоматически генерируется исполняемая программа. Эксперт приложений автоматически создает все необходимые файлы проекта, а эксперт ресурсов знает, как сгенерировать код Пролога, который будет обеспечивать использование всех выбранных ресурсов.

При создании приложения с помощью команды **Project/New Project** необходимо вызвать Мастер приложений (Application Expert) для ввода имени и местоположения файла базы данных проекта с типом VPR. База данных проекта содержит все установки компиляции, имена и место расположения всех исходных модулей проекта, характеристик интерфейса пользователя и множества других параметров. Все опции, устанавливаемые командами меню **Options/Project**, сохраняются в этой базе данных. Для конвертирования файла проекта в текстовый файл и обратно может быть использована утилита VIPCONV.EXE.

**Окно проекта**



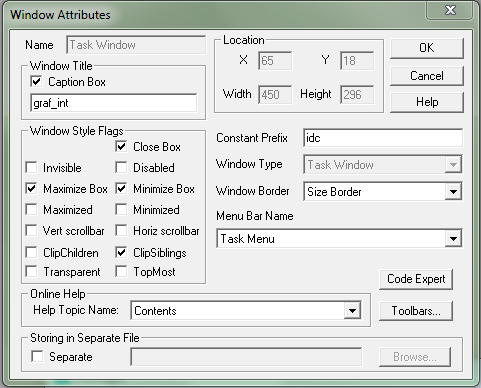
Окно проекта содержит списки всех компонентов приложения, сгруппированных по типам.

В правой части окна проекта расположены кнопки для выполнения операций над компонентами проекта. Двойной щелчок мыши по элементу списка или нажатие кнопки **Edit** вызывает редактор соответствующего компонента. Кнопка **New** позволяет начать создание нового компонента (с заполнения полей окна атрибутов). Кнопка **Delete** предназначена для удаления компонента из проекта, а кнопка **Attribute** позволяет изменить атрибуты компонента. С помощью кнопки **Code Expert** можно вызвать соответствующий эксперт кода для компонента.

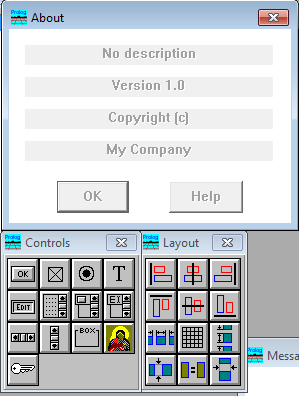
Visual Prolog включает различные редакторы для проектирования пользовательских интерфейсов. Результатом работы редакторов является или создание ресурса, или текст исходного кода Пролога, который при исполнении приложения создаст необходимый интерфейс пользователя. Группировка компонентов по типам соответствует группировке соответствующих редакторов и отображается кнопками в левой части окна проекта.

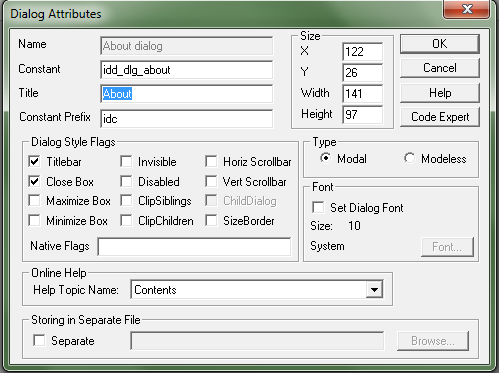
1. Текстовый редактор (**Module**). Visual Prolog включает развитый текстовый редактор, который окрашивает коды Visual Prolog, лексемы и ключевые слова. Окраска в разные цвета позволяет намного легче ориентироваться среди названий (имен) предикатов, параметров, комментариев и т. д. Редактор поддерживает неограниченную отмену / восстановление, поиск, замену, вырезание, копирование, вставку, перетаскивание блока, функциональные возможности гипертекста и т. д.

2. Редактор окон и форм (диалогов) (**Window, Dialog**).



С использованием Редактора форм в Visual Prolog возможно проектирование различных окон и диалогов (форм). При этом можно использовать стандартные средства управления, например кнопки, переключатели и т. д.



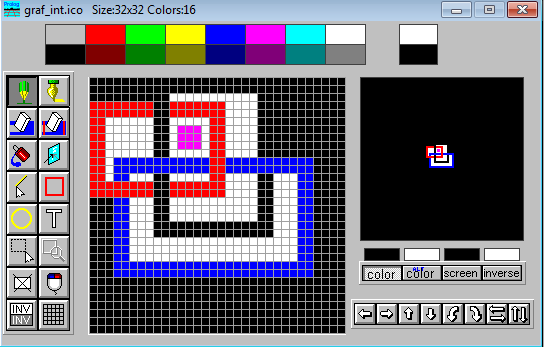


Используются также определяемые элементы (**Custom Control**).

Операции управления позволяют выполнять выравнивание различными способами. Может быть установлен порядок обхода элементов форм при использования клавиши **Tab**.

Каждому элементу управления назначают символическое имя. Эти константы автоматически помещаются в код и используются, когда содержание элемента должно быть найдено или изменено. Кроме того, эти константы записываются в файл, который автоматически включается в исходные файлы.

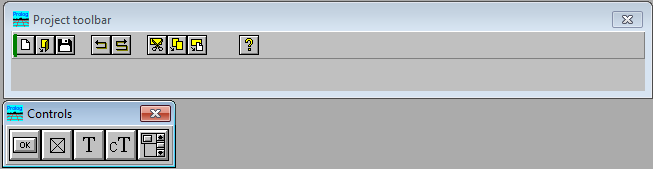
3. Редактор растровых изображений и икон (**Bitmap, Icon, Cursor**). Программы Windows часто используют различные рисунки – иконки, курсоры, кнопки, элементы инструментальных панелей и т. д., что требует разработки таких изображений. Visual Prolog включает редактор изображений.



Редактор изображений может вращать или перемещать изображения. Имеется специальное средство – зеркало, которое позволяет создавать симметричные изображения. Для курсоров могут быть установлены "горячие точки". Однако встроенный редактор изображения не предназначен для редактирования растровых рисунков большого размера. В этом случае должна использоваться другая, более подходящая программа из средств базовой операционной системы.

Все курсоры, иконки и растровые рисунки, которые созданы в проекте, легко могут быть обработаны другими редакторами. На основе созданных растровых рисунков кнопок легко создать инструментальную панель.

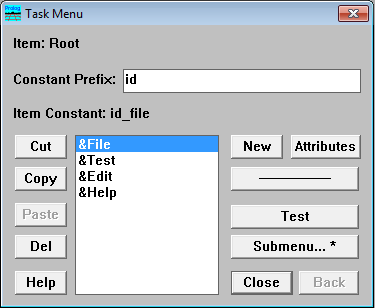
4. Редактор инструментальной панели (**Toolbar**).



Этот редактор позволяет в интерактивном режиме создавать интерфейс для различных видов инструментальных панелей. Стиль инструментальной панели (Toolbar Style) может быть одним из следующих:

* Top – помещена в верхней части окна;
* Left – помещена слева;
* Right – помещена с правой стороны окна;
* Bottom – помещена в нижней части окна;
* Inside – помещена в произвольное место внутри окна;
* Moveable – плавающая инструментальная панель, которую пользователь может перемещать.

5. Редактор меню (**Menu**).

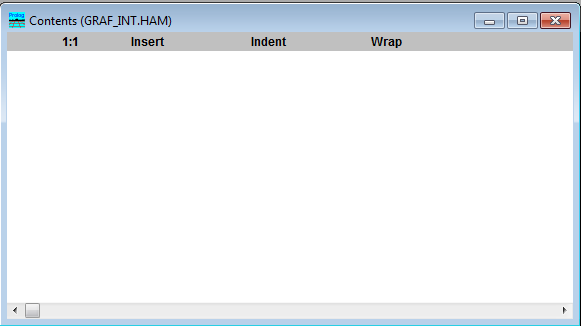


Позволяет облегчить процесс создания объектно-ориентированных интерфейсов пользователя. Для каждого входа в меню в программе инициируется событие в тот момент, когда пользователь нажимает позицию меню.

6. Редактор ресурсных строк (**String**).

Используется, чтобы создавать таблицы строк в ресурсном файле. Таблицы строк могут содержать сообщения об ошибках, подсказки или любой другой текст, который должен отобразиться в приложении. Можно хранить множество таблиц строк в проектном файле.

7. Интегрированный редактор системы помощи (**Help Topics**).



Встроенный редактор для создания систем помощи (Help-система) позволяет легко создавать интерактивные системы помощи к разрабатываемому приложению. Help-система основана на гипертекстовой абстрактной машине. В Help-системе возможны интерактивный ввод содержимого, разметка новых ссылок с помощью мыши и перемещение по ссылкам в фазе редактирования. Help-система может создавать файлы в RTF-формате Windows.

Описания диалогов, меню, строк, иконок и изображений в BMP-формате размещаются в базе данных проекта и в конечном итоге объединяются в главном файле ресурсов с расширением .RC или .RES. Описания окон и панелей инструментов используются при генерации исходного кода с целью создания их на стадии выполнения.

**Эксперты кода и инструментальные средства**

Разнообразные редакторы ресурсов тесно связаны с кодовыми экспертами в интегрированной среде Visual Prolog. Кодовые эксперты обеспечивают автоматическое поддержание соответствия текста программ экранным конструкциям. Таких экспертов несколько.

1. Эксперт приложений (вызывается командой главного меню **Options/Project/Application expert**).



Используется для создания новых проектов и назначения новых свойств уже существующим проектам. Он генерирует коды всех основных частей нового приложения, проектного файла, исходных файлов, меню, инструментальных панелей, иконок, растровых рисунков и даже файла помощи.

Эксперт приложений позволяет выбирать платформу, на которой должны выполняться программа (карточка **Target** в окне диалога эксперта).

С помощью эксперта приложений можно установить правильную структуру для всех используемых инструментальных средств. Например, он обеспечивает обращение к SQL-базе данных через ODBC, работу в текстовом редакторе, представление данных в виде дерева, трехмерное представление элементов интерфейса, использование инструментальных панелей, организацию главной программы на C или C++.

Произведенный экспертом приложений код служит структурой, в которую можно легко добавлять специфические особенности приложения.

2. Эксперт форм (вызывается кнопкой **Code Expert** в окне проекта или командой **Project/Code Expert** из главного меню).



Главная функция эксперта форм (окон и диалогов) заключается в том, чтобы автоматически генерировать тексты программы, которые обрабатывают различные события в среде системы окон. В окне диалога эксперта все типы события могут быть отмечены, а нажатие кнопки **Add Clause** (Добавить правило) добавляет фрагмент кода для выбранного события. Например, для события перемещения окна в другое место будет произведен следующий код:

*%BEGIN mywin, e\_Move*

*win\_mywin\_eh(\_Win,e\_Move(\_X,\_Y),0):-!,*

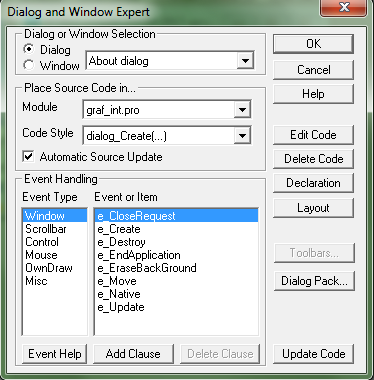
*!.*

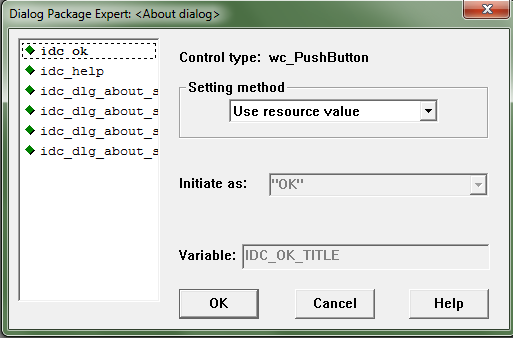
*%END mywin, e\_Move*

Обратите внимание на то, что элемент отмечен парой *%BEGIN - %END*. Она используется экспертом форм для последующих добавлений кода. В частности, как только фрагмент кода создан, кнопка **Add Clause** изменяется на **Edit** **Clause** (Редактирование) и может использоваться для быстрого перехода к фрагменту кода в пределах исходного модуля для редактирования. Всплывающие меню в текстовом редакторе позволяют быстро вставлять тексты вызовов предикатов, констант, имен доменов, ключевых слов и идентификаторов ресурсов.

VPI обеспечивает заданное по умолчанию поведение для всех событий, поэтому можно только вставить код для событий, которые необходимо обработать явно.

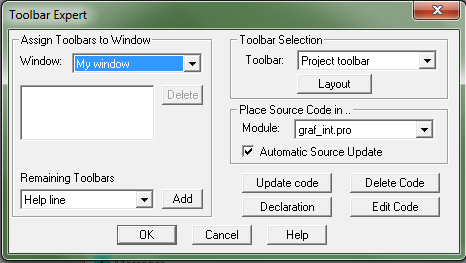
3. Эксперта кода для пакета диалога (вызывается кнопкой **Dialog Pack** в окне эксперта форм при установленной опции **Dialog**).



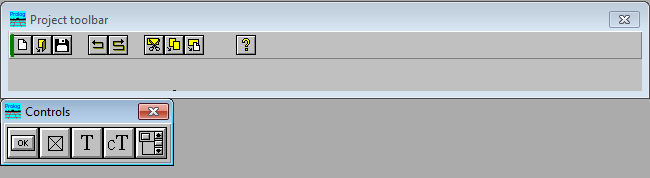


Данный инструмент дает возможность выбрать множество свойств поведения диалога, а также выбрать, какие переменные программы должны быть связаны с управлением. Это позволяет автоматически выполнять различные проверки правильности, например проверку, позволяющую видеть, является ли содержимое поля целым числом, находится ли в пределах данного диапазона или строка превысила максимальную длину и т. д. Единственное, что программист должен делать, – установить значение для каждого поля прежде, чем диалог создан, и когда диалог закончен, возвратить результат.

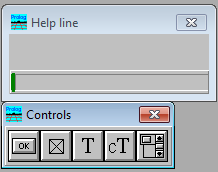
4. Эксперт инструментальной панели (вызывается кнопкой **Code Expert** в окне проекта при выбранном компоненте типа **Dialog** или командой **Project/Code Expert** из главного меню при установленной опции **Toolbar**).



В приложениях могут быть использованы инструментальные панели (**Project toolbar**)



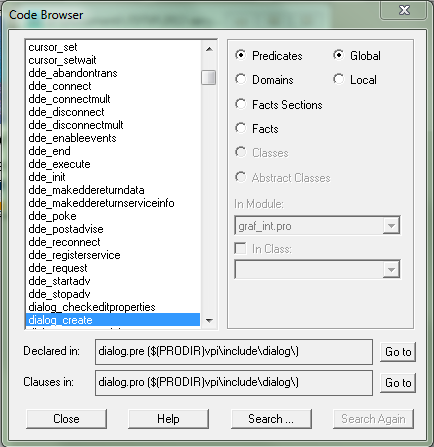
и строки подсказок (**Help line**).



Эксперт инструментальной панели генерирует код для динамического подключения инструментальной панели к соответствующему окну и определения ее свойств. Все курсоры, иконки и растровые изображения, используемые в проекте, легко могут быть изготовлены с помощью любых редакторов. Если растровые изображения кнопок уже созданы, то данный эксперт позволяет легко создать функциональную инструментальную панель.

В Visual Prolog можно импортировать ресурсы из .VPR-, .RES-, .EXE- или .DLL-файлов (команда **Resource/Import**). Существующие в других приложениях или в проектах, созданных в Visual Prolog, меню, диалоги, ресурсные строки, растровые рисунки, иконки и курсоры могут быть импортированы в проект и многократно использоваться.

Visual Prolog имеет мощную систему просмотра кода (**Project/Code Browser**), которая делает очень простым определение расположения доменов, фактов или описаний и правил для предикатов в файлах. В результате компиляции создается файл с расширением .BRO, в котором хранится информация о доменах и предикатах, доступных для просмотра посредством команды **Project/Code Browser**.



С помощью кнопок **Go to** можно перейти к редактированию описания или правила для выбранного в окне Browse предиката.

С помощью команды меню для вывода окна дерева структуры проекта (**Project/Project Tree**) можно получить структуру проекта в обозримом иерархическом виде. При этом возможно с помощью команд контекстного меню в окне структуры добавлять внешние (например, написанные на языке C) файлы к проекту, включать и строить исходные тексты таким образом, что внешние файлы будут автоматически являться частью разрабатываемого приложения.

Visual Prolog автоматически заботится о поддержании целостности проектов, об обновлении файлов ресурсов и соответствующих кодов при изменениях в проекте. Компиляция выполняется только для файлов, претерпевших изменения. Система построения проектов включает поддержку своевременных изменений исходных текстов программ, компиляцию и связывание всех модулей проекта. В основе этой процедуры лежит общий механизм, позволяющий менять правила построения проекта и даже компилировать и связывать С-программы или программы, написанные на ином языке. Система построения проектов тесно связана с представлением проекта в виде дерева.

**Отладчик в системе Visual Prolog**

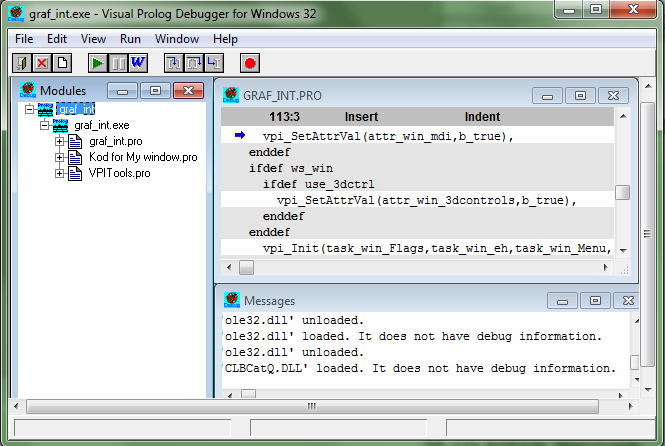
Отладчик Visual Prolog имеет все свойства и средства отладчиков, встречающиеся в других современных средах разработки. Он работает на cкомпилированном коде. Это делает программу очень устойчивой и полностью независимой от среды разработки.

Отладчик позволяет устанавливать точки останова и выполнять программу по шагам с просмотром кода на Прологе и на Ассемблере, отображать значения переменных, просматривать и отказываться от фактов, просматривать состояние динамической памяти, стека и дампов памяти.

Отладчик является самостоятельной исполняемой программой VIPDEBUG.EXE, помещенной в директорию BIN\WIN\32\. Вызов отладчика обычно производится из VDE активизацией меню **Project/Debug** или нажатием соответствующей кнопки инструментальной панели.

Для использования отладчика прежде всего необходимо включить создание отладочной информации в диалоге свойств компилятора (команда **Options/Project/Compiler Options**). Включение флажка **Generate Debug Information** (Генерировать отладочную информацию) на карточке **Output** приведет к генерации отладочной информации компилятором (.DEB-файлы в директории OBJ). DEB-файлы используются отладчиком для связи с исходными файлами и получения информации о переменных, фактах и т. д. в процессе исполнения программы.

Программа может быть запущена с использованием меню (**Run/Run**) или с помощью кнопки инструментальной панели. После загрузки программы отладчиком начальная точка и некоторая другая информация (перечень загруженных модулей, указание, какие из них могут отлаживаться и т. д.) может быть видна в окне сообщений в нижней части экрана.



Рассмотрим основные возможности отладки программ, поддерживаемые средствами меню, панелью инструментов и системой окон среды отладчика.

##### Обзор модулей программы. Когда программа загружена отладчиком, ее структура видна в окне модулей (*Modules*)*,* где она представлена в виде дерева структуры файлов.

В режиме просмотра двойным нажатием мыши на файле, входящем проект, или на определенном предикате можно открыть окно с исходным кодом и текстовым курсором в соответствующем месте.

Исходный код выводится только для программ на языке системы Visual Prolog. Остальные программы могут быть представлены только в ассемблерном представлении в окне дизассемблирования.

В окне отладчика редактировать текст невозможно, поскольку отладчик является самостоятельной программой. Наиболее удобным способом корректировки кода является синхронный поиск предикатов в отладчике и в VDE.

##### Просмотр исходного кода. Команда меню **View/Go to Executing Predicate** **Source** устанавливает курсор в позицию текущей инструкции.

Фон участков кода, где могут быть установлены точки останова, белого цвета.

С помощью правой кнопки мыши или пункта меню **Edit** возможны копирование текста в буфер обмена Windows (командой **Copy**), поиск текста (команда **Search**), переход в нужную позицию или на нужную строку (число в окне диалога команд **Go to position** или в **Go to Line Number** соответственно). Номер текущей позиции может быть получен вызовом команды **Show Position** (Показать позицию).

##### Запуск приложения (**Run/Run**). Пункт меню **Run** или клавиша **F9** используется для управления исполнением отлаживаемой программы.

Можно прервать исполняемую под отладчиком программу с помощью команды **Run/Break Program** или кнопки **Break**. Затем можно проверить состояние стека вызовов, чтобы понять, что программа делает в действительности. Можно также установить новую точку останова для перехвата исполнения.

Команда **Run/Trace Into** или клавиша **F7** используется для пошагового исполнения кода.

Команда **Run/Step Over** или клавиша **F8** используется для пропуска контроля исполнения данного предиката.

Команда **Run/Run to Cursor** или клавиша **F4** работает подобно установке точки останова в текущую позицию и вызову исполнения программы.

##### Установка точек останова. Эта операция является очень важным средством для локализации ошибок или для подтверждения правильности работы программы. При открытом окне с исходным кодом, находясь на нужной строке, возможно нажатием правой кнопки мыши вызвать меню и установить точку останова. Программа будет исполняться только до этой точки.

##### Больше возможностей предоставляет окно **Break Points** (Окно точек останова). Это окно может быть открыто выбором **Break Points** из меню **View**. В этом окне виден список всех точек останова, и их состояние может быть изменено: точка активирована или удалена. Все это делается правой кнопкой мыши. Нажатие на пункте **Remove** удаляет точку останова, нажатие на пункте **Toggle** переключает состояния "активно"/"неактивно". Пункт меню **Goto Code** позволяет перейти к исходному тексту в то место, где установлена точка останова. Другой полезной особенностью является управление свойствами точки останова выбором пункта **Properties** (Свойства). При этом можно наблюдать, сколько раз точка останова была достигнута (в поле **Counter**), и установить, что если она достигается *N* раз, то она активизируется (устанавливается в поле **Current Count**).

##### Просмотр значений переменных. В процессе выполнения программы значения переменных можно наблюдать в окне переменных (вызывается командой **View/Local Variables**). При позиционировании указателя мыши на переменной в исходном тексте текущее значение переменной появляется на некоторое время. Таким образом можно контролировать значения переменных без открытия окна переменных.

##### Представление фактов. Важным моментом при проверке правильности работы программы, написанной на Прологе, является проверка того, что в память записываются правильные факты. Содержимое памяти фактов можно наблюдать в окне фактов. Это окно открывается вызовом пункта **Facts** меню **View**.

##### Стек вызовов. Окно стека вызовов (**View/Call Stack**) показывает список предикатов, которые уже были вызваны в отлаживаемой программе. Двойное нажатие на предикате показывает место в исходном коде, где был вызван этого предикат.

##### Окно точек перехвата. Это окно (**View/Trap Points**) показывает список точек перехвата в отлаживаемой программе. Двойное нажатие мыши на точке перехвата покажет место в коде, где этот перехват предусмотрен.

##### Окно точек отката. Это окно (**View/Backtrack Points**) показывает список точек отката в отлаживаемой программе. Двойное нажатие мыши на точке отката покажет место в коде, где эта точка находится.

##### Представления низкого машинного уровня. Существует возможность ассемблерного представления кода непосредственно в отладчике (**View/Memory** или **View/Registers**); можно видеть состояние регистров и содержимое памяти. Основное назначение этого механизма – контроль кодов, написанных не на Прологе.

Опции отладчика. Опции можно получить, выбрав пункт меню **View/Configuration**:

* опция **Show Startup Code** на карточке **Global** позволяет проверить поведение программы перед вызовом основного запроса Goal;
* опция **Save BreakPoints** позволяет сохранить все установки точек останова для текущего проекта в файле .ini отладчика;
* опция **Do not show Assembler as possible** позволяет не показывать ссылки на ассемблерный код;
* если выбрана опция **Scanning Call Stack from Goal** на карточке **Call** **Stack**, то окно стека вызовов будет содержать только вызовы, выполненные после вызова Goal.