

Поволжский государственный университет  
телекоммуникаций и информатики

Акчурин Э.А.

Программирование на языке FreePascal  
Часть 2. Интегрированные среды разработки

Учебное пособие для студентов направления  
«Информатика и вычислительная техника»

Самара 2008

Факультет информационных систем и технологий  
Кафедра «Информатика и вычислительная техника»

Автор - д.т.н., профессор Акчурин Э.А.



Другие материалы по дисциплине Вы найдете на сайте

[www.ivt.psuti.ru](http://www.ivt.psuti.ru)

1. Список литературы .....	5
2. Введение.....	5
3. ИСР Free Pascal .....	5
3.1. Главное меню	6
3.2. Пункт File (Файл)	7
3.3. Пункт Edit (Правка)	10
3.4. Пункт Search (Поиск)	11
3.5. Пункт Run (Запуск)	12
3.6. Пункт Compile (Компиляция)	13
3.7. Пункт Debug (Отладка)	14
3.8. Пункт Tools (Инструменты)	15
3.9. Пункт Options (Опции)	16
3.10. Пункт Windows (Окна)	26
3.11. Пункт Help (Справка)	27
4. ИСР Lazarus.....	29
4.1. Установка ИСР Lazarus	29
4.2. Вход в ИСР Lazarus	31
4.3. Главное окно Lazarus	34
4.3.1. Содержание окна	34
4.3.2. Панель инструментов	35
4.3.3. Панель Компоненты	35
4.4. Окно Конструктора формы	43
4.5. Окно инспектора объектов	44
4.6. Окно редактора кода	46
4.7. Структура программ Lazarus	48
4.7.1. Консольное приложение	49

4.7.2. Приложение	50
4.8. Компиляция и выполнение проекта	52
4.9. Главное меню	53
4.9.1. Пункт Файл	53
4.9.2. Пункт Правка	55
4.9.3. Пункт Поиск	56
4.9.4. Пункт Просмотр	57
4.9.5. Пункт Проект	58
4.9.6. Пункт Запуск	58
4.9.7. Пункт Компоненты	59
4.9.8. Пункт Инструменты	59
4.9.9. Пункт Окружение	59
4.9.10. Пункт Окна	60
4.9.11. Пункт Справка	60
4.10. Модули в составе ИСП Lazarus	62
4.11. Графика	62
4.11.1. Класс TCanvas - холст	62
4.11.2. Вывод текста	63
4.11.3. Простые графические примитивы	64
4.11.4. Фигуры	67
4.11.5. Заполнение замкнутых областей	71
4.12. Рисование графиков функций	73
4.13. Компоненты	76
4.13.1. Текстовые компоненты	77
4.13.2. Кнопки	79
4.13.3. Счетчики	80
4.13.4. Переключатели	81
4.13.5. Группы	82
4.13.6. Списки	82
4.13.7. Управляющие компоненты	84
4.13.8. Таблицы	84
4.13.9. Диалоги	85
4.13.10. Системные компоненты	85
4.13.11. Компоненты графики	86
4.13.12. Компоненты создания меню	89
5. Отладка программ.....	90

## 1. Список литературы

1. Ван Кеннейт М.. Справочное руководство для FPC. 2005.
2. Van Canneyt M. Free Pascal: Reference Guide. 2007.
3. Van Canneyt M. Free Pascal: Programmer's manual. 2007.
4. Van Canneyt M., Klampfl F. Free Pascal: Users' manual. 2007.
5. Van Canneyt M. Free Pascal code documenter: Reference manual. 2007.
6. Van Canneyt M. Free Component Library (FCL): Reference guide. 2007.
7. Van Canneyt M. Run-Time Library (RTL). Reference guide. 2007.
8. Van Canneyt M. Free Component Library (FCL): Reference guide. 2007.
9. <http://www.FPC.org/download/i386/win32-ftp.FPC.org.html>
10. <http://www.FPC.org/download.html>
11. Рудюк С.А., Lazarus - кросс-платформенный Delphi. Программирование для свободных людей. Компания НеРусСофт.
12. Карпов Б. Delphi: Специальный справочник. - СПб.: Питер, 2001. -688 с.
13. Акчурин Э.А. Программирование на языке высокого уровня. Учебное пособие. Самара, ПГАТИ, 2003.
14. Акчурин Э.А. , Стефанова И.А. Программирование в среде Delphi. Методические указания к лабораторным работам. Самара, ПГАТИ, 2003.
15. Акчурин Э.А. Задания и методические указания к курсовой работе. ПГАТИ, 2001.

## 2. Введение

В настоящее время бесплатный компилятор FPC (Free Pascal Compiler) используется в нескольких интегрированных средах разработки (ИСР). Наиболее распространены две ИСР:

- FreePascal (FP) обеспечивает пользователю комфортный интерфейс. Содержит редактор с синтаксическими выделениями, отладчик, браузер символов и др. ИСР для всех поддерживаемых ОС использует символьный интерфейс, подобный ИСР Turbo Pascal. Не поддерживает графику.
- Lazarus. Эта ИСР подобна Delphi, использует графический интерфейс. Поддерживает графику.

## 3. ИСР Free Pascal

При старте ИСР отображается окно среды, которое содержит:

- Строку заголовка с именем ИСР – Free Pascal.
- Главное меню.
- Рабочее поле.
- Окно сообщений Messages.

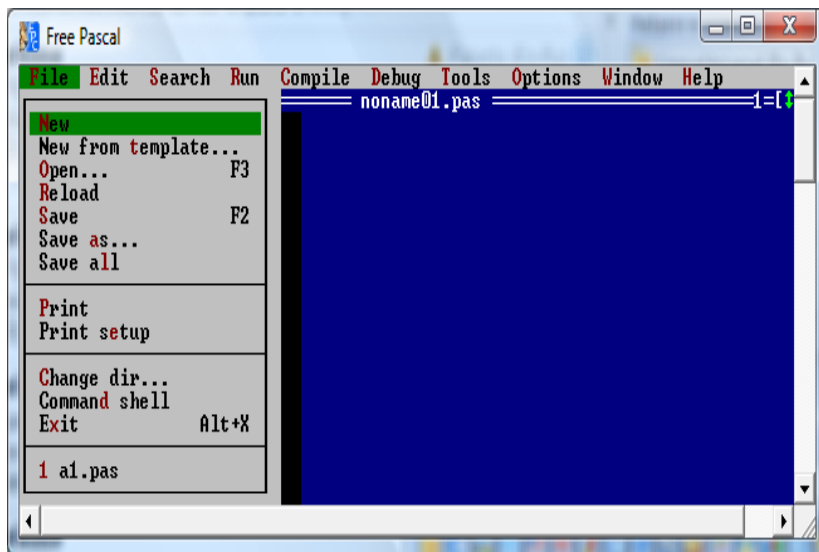
- Строка быстрого выбора часто используемых команд.



### 3.1. Главное меню

В главном меню доступны все команды ИСР.

### 3.2. Пункт File (Файл)

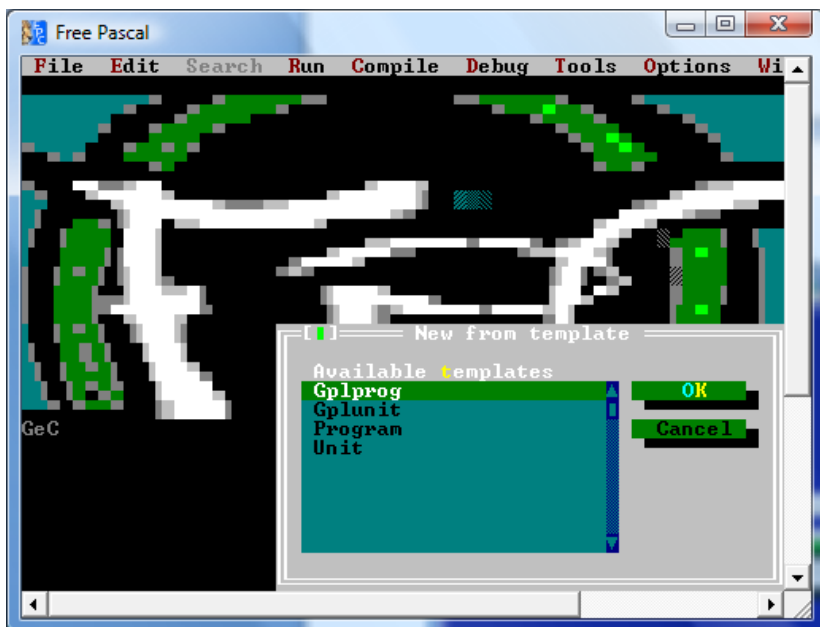


Использует диалоговые команды работы с файлами:

Команда	Действие
New	Новый. Создается пустое окно редактора кода.
New from template...	Новый из шаблона. Создается окно редактора с шаблоном кода для программы или модуля.
Open...	Открыть.
Reload	Перезагрузить.
Save	Сохранить под старым именем.
Save as...	Сохранить как.
Save all	Сохранить все..
Print	Печать.
Print setup	Установка принтера.
Change dir	Изменить каталог.
Command shell	Запускается оболочка с командной строкой.
Exit	Выход

Команда New создает пустое окно программы с именем noname01.

Команда New from template формирует окно выбора шаблона. В нем нужно выбрать тип проекта (Program - программа, Unit - модуль).

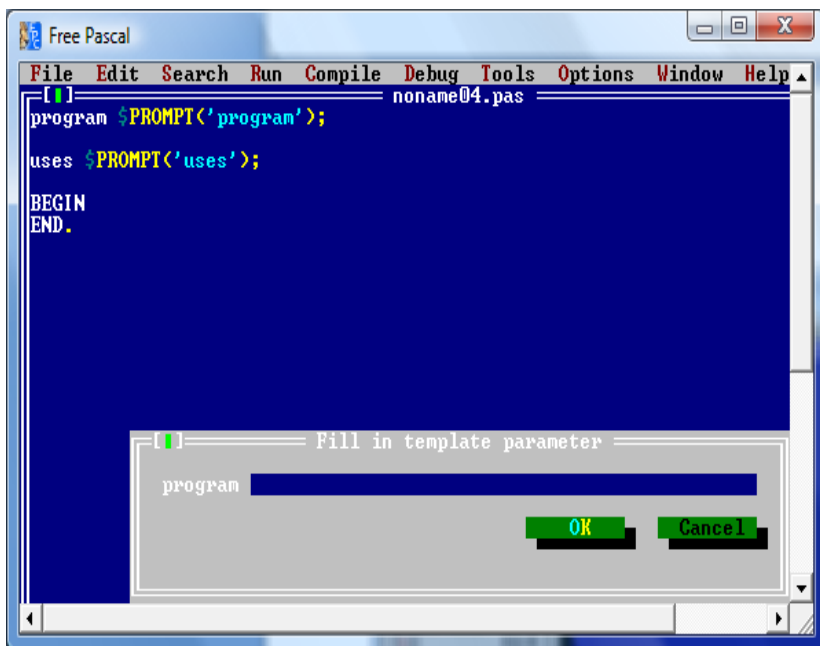


После выбора типа проекта формируется окно проекта с приглашением уточнить:

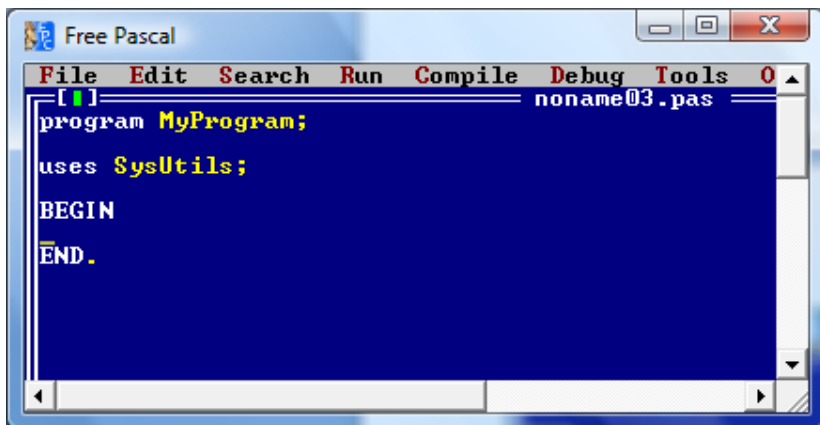
- имя программы Prompt<'program'>,
- список используемых модулей Prompt<'uses'>.

Запрашиваемые данные нужно последовательно ввести в диалоговых полях ввода.

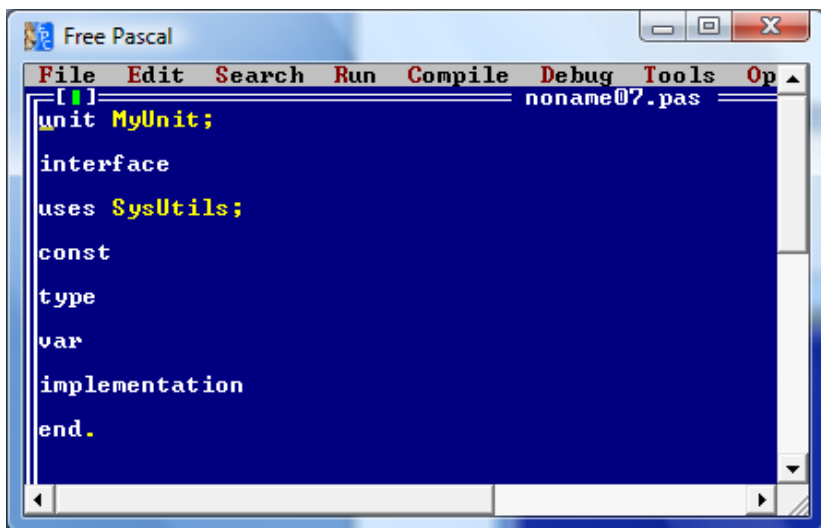




В итоге формируется окно программы с обязательными строками кода. Это ускоряет программирование и уменьшает количество ошибок.

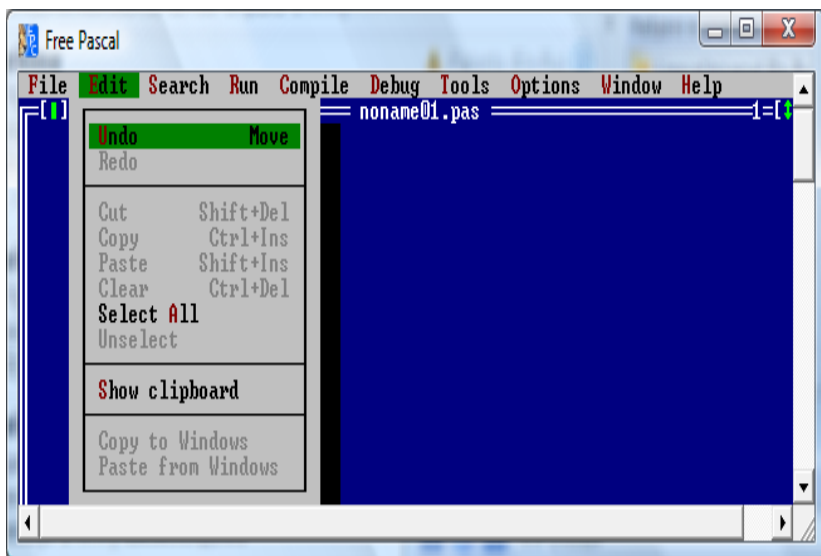


Аналогичное действие для модуля приводит к созданию следующего окна кода модуля.



Завершающая часть – сохранение кода программы.

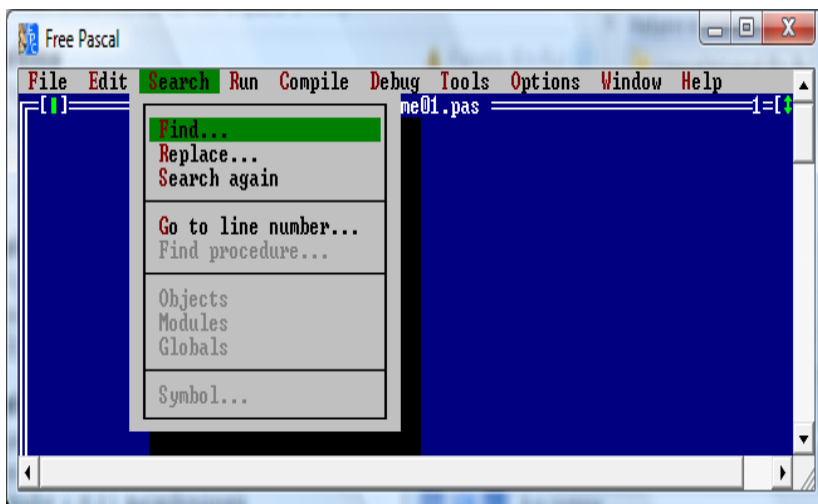
### 3.3. Пункт Edit (Правка)



Использует стандартные диалоговые команды редактирования:

Команда	Действие
Undo	Отменить последнюю команду.
Redo	Повторить отмененную команду.
Cut	Вырезать выделенные строки в буфер.
Copy	Копировать выделенные строки в буфер
Paste	Вставить из буфера в позицию курсора.
Clear	Удалить выделенное.
Select all	Выделить все.
Unselect	Отменить выделение
Show clipboard	Показать буфер
Copy to Windows	Копировать выделенные строки в Windows.
Paste from Windows	Вставить из Windows.

### 3.4. Пункт Search (Поиск)

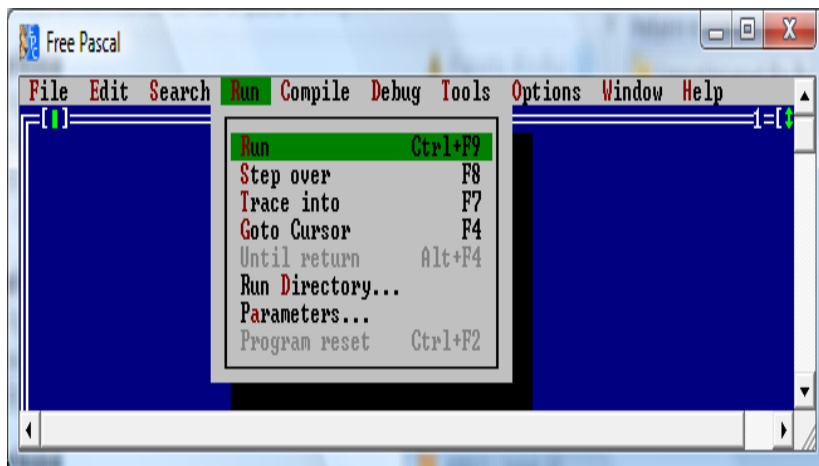


Использует стандартные диалоговые команды поиска:

Команда	Действие
Find...	Найти. Окно диалога поиска.
Replace...	Заменить. Окно диалога поиска и замены.
Search again	Повторить последний поиск
Go to line number...	Перейти к строке номер...
Find procedure...	Найти процедуру (пока не реализовано)
Objects	Объекты.
Modules	Модули.

Globals	Глобальные символы.
Symbol...	Все символы.

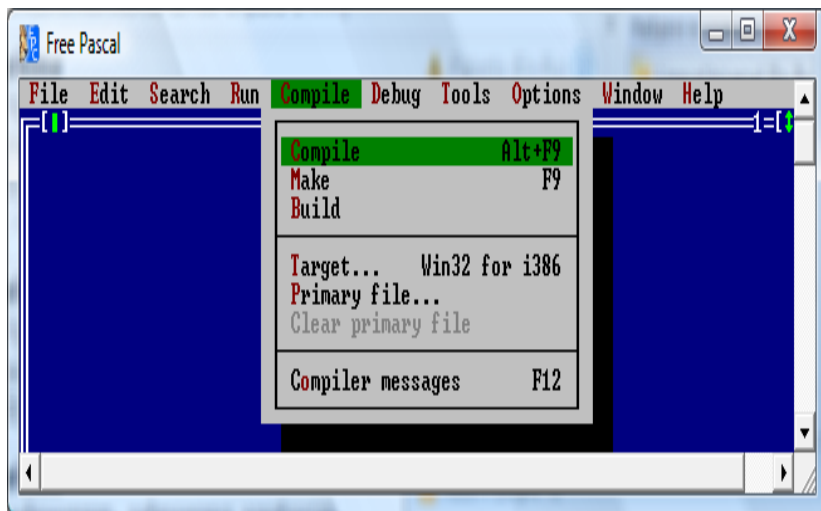
### 3.5. Пункт Run (Запуск)



Использует диалоговые команды исполнения:

Команда	Действие
Run	Запуск. Если исходник изменялся, то он сохраняется, компилируется и запускается на исполнение.
Step over	Шаг в обход. Пошаговый проход, подпрограмма за один шаг.
Trace into	Шаг с входом. Пошаговый проход, подпрограмма по шагам.
Goto Cursor	Запуск до курсора.
Until return	Прогон до конца текущей процедуры.
Run Directory...	Изменить каталог.
Parameters...	Ввод значений параметров.
Program reset	Останов.

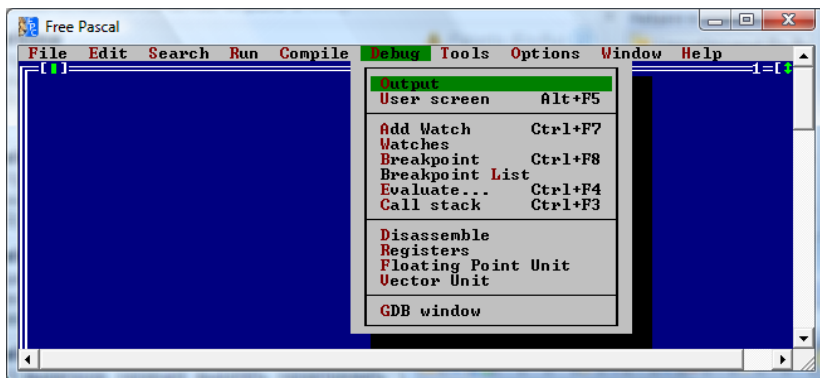
### 3.6. Пункт Compile (Компиляция)



Использует диалоговые команды компиляции:

Команда	Действие
Compile	Компиляция кода текущего окна.
Make	Компиляция текущего окна и всех модулей и программ его использующих, которые были изменены после последней компиляции.
Build	Компиляция текущего окна и всех модулей и программ его использующих.
Target... Win32 for i386	Выбор платформы из списка. По умолчанию Win32 для i386
Primary file...	Выбрать первичный файл
Clear primery file	Удалить первичный файл
Compiler messages	Отобразить окно с сообщениями компилятора

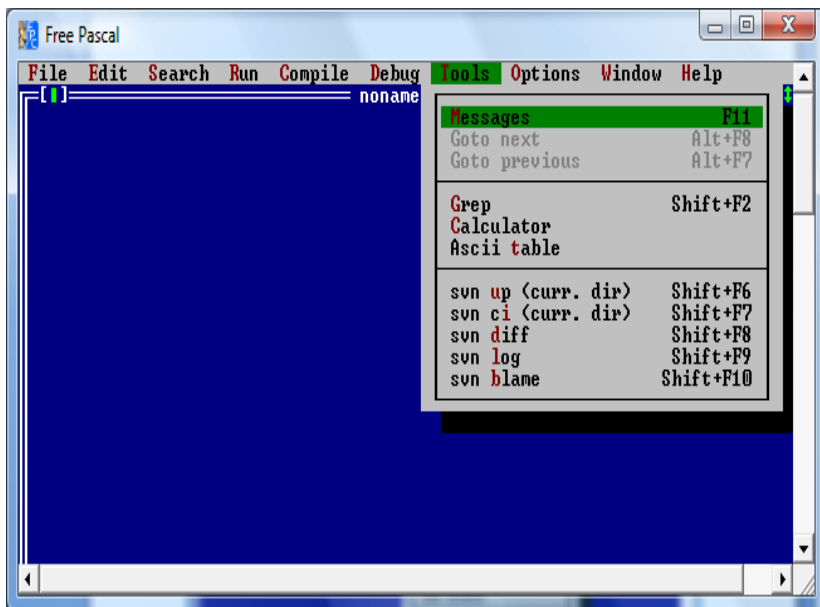
### 3.7. Пункт Debug (Отладка)



Использует диалоговые команды отладки:

Команда	Действие
Output	Вывод.
User screen	Экран пользователя, последний до прогона.
Add Watch	Добавить наблюдаемое.
Watches	Список наблюдения.
Breakpoint	Добавить точку останова.
Breakpoint List	Список точек останова.
Evaluate	Вычислить.
Call stack	Вызов стека.
Disassemble	Вывод окна дизассемблера.
Registers	Вывод окна с содержимым регистров процессора.
Float Point Unit	Модуль с плавающей точкой.
Vector Unit	Векторный модуль.
GDB window	Показать окно GDB отладчика.

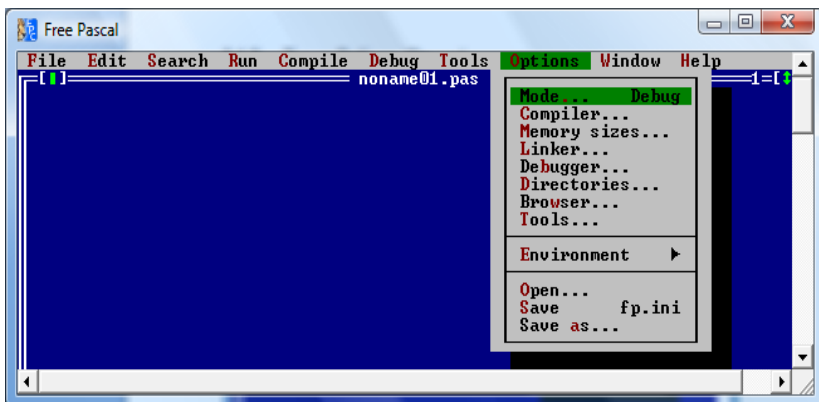
### 3.8. Пункт Tools (Инструменты)



Использует диалоговые команды обращения к инструментальным средствам:

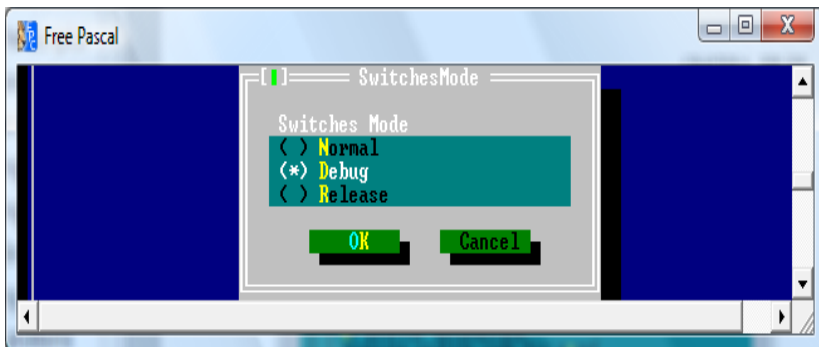
Команда	Действие
Messages	Показать окно сообщений, в него будут выводиться сообщения от одного из инструментов.
Goto next	Перейти к следующему сообщению.
Goto previous	Перейти к предыдущему сообщению.
Grep	Выводит приглашение с регулярным выражением и его опциями, которые будут переданы инструменту Grep, который осуществляет поиск заданной строки в файле.
Calculator	Отображает калькулятор.
ASCII table	Выводит таблицу символов ASCII.
Svn up (curr dir)	
Svn ci (curr dir)	
Svn diff	
Svn log	
Svn blame	

### 3.9. Пункт Options (Опции)



Использует диалоговые команды установки режимов.

Команда Mode используется для выбора режима ИСП. Выводит окно диалога

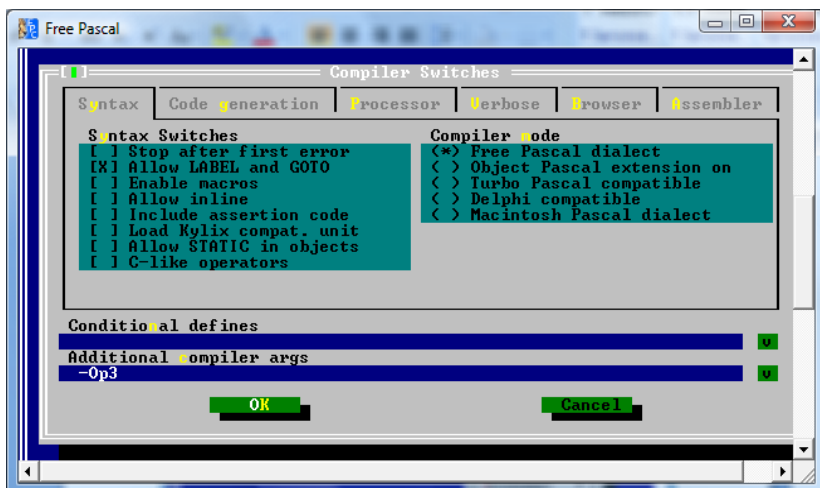


для выбора режима из списка возможных:

- Normal – нормальная (быстрая) компиляция.
- Debugger – с отладкой (по умолчанию).
- Release – компиляция программы для реализации, с удалением отладочной информации.

Команда Compiler... используется для выбора параметров компилятора. Отображается диалоговое окно с закладками. Закладка Syntax (Синтаксис)

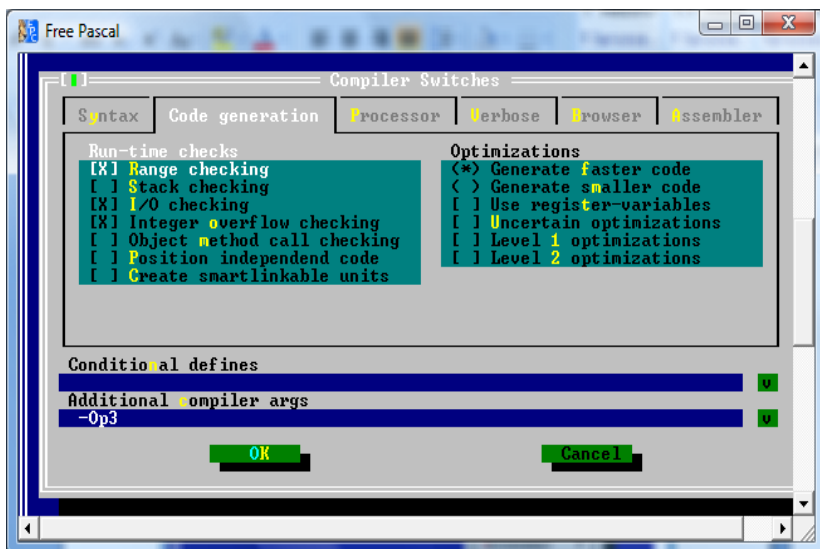




включает поля:

- Syntax switches – переключатели опций синтаксиса.
- Compiler mode - режима компилятора.

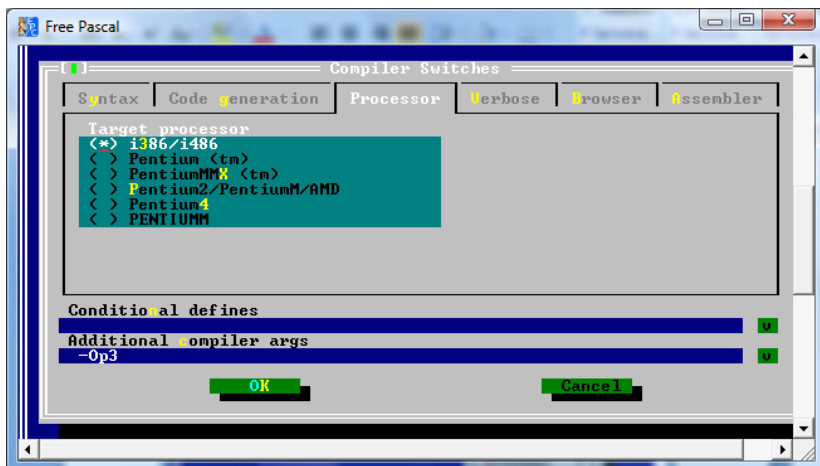
Закладка Code generation (Генерация кода)



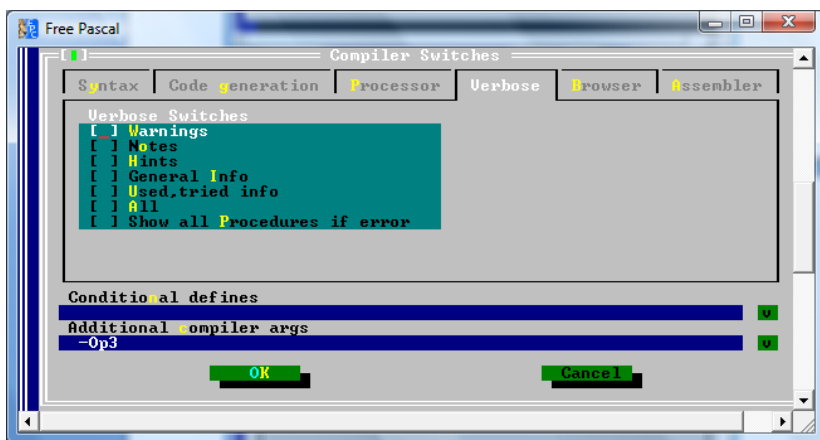
включает поля:

- Run-time checks - проверки в режиме реального времени.
- Optimisations - типы оптимизации.

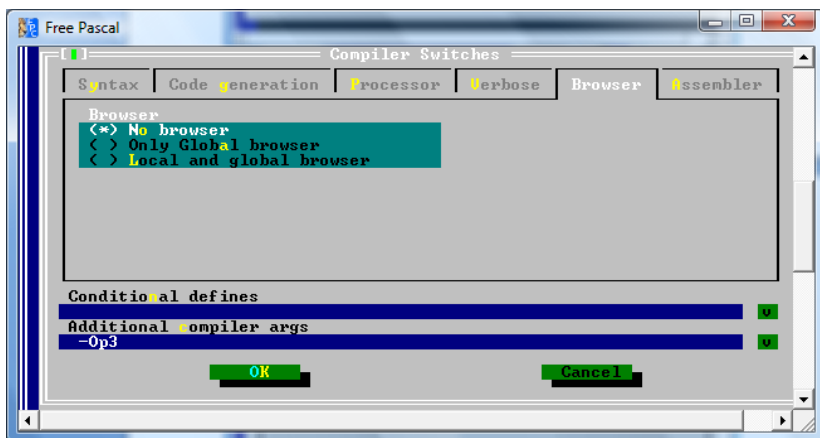
Закладка Processor (Процессор) позволяет выбрать процессор, по умолчанию i386.



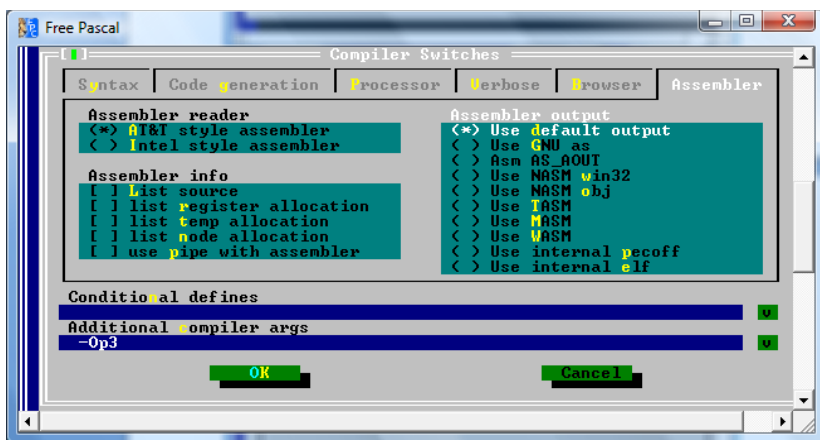
Закладка Verbose (Подробности) позволяет выбрать поясняющие действия.



Закладка Browser (Браузер) позволяет использовать браузер для просмотра локальных или глобальных переменных.



Закладка Assembler (Ассемблер) – выбор типа ассемблера.

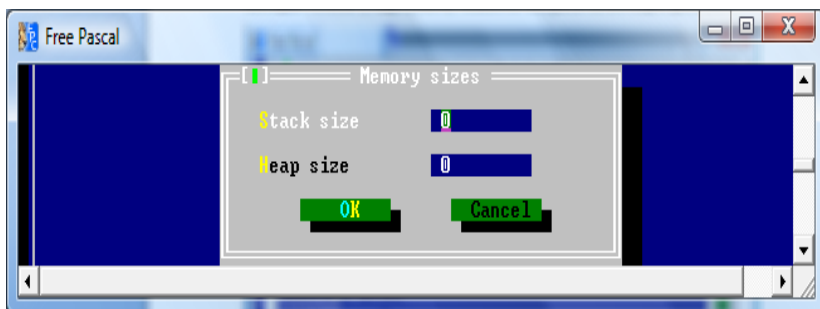


Включает поля:

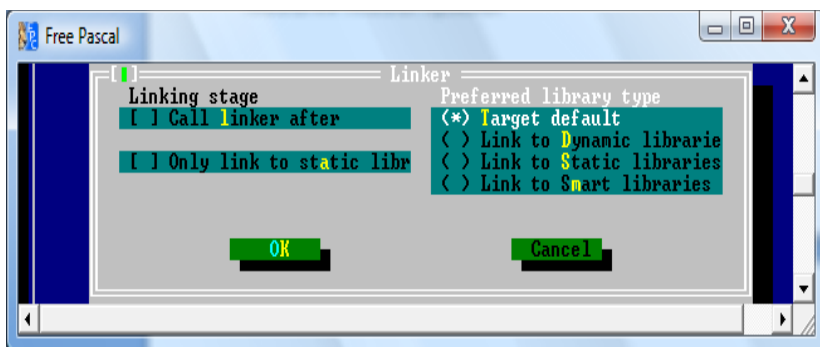
- Assembler reader – режим чтения ассемблера.
- Assembler info – информация о ассемблере.
- Assembler output - режим вывода ассемблера.

Команда Memory size используется для выбора размеров памяти. Выводит окно диалога, в полях которого можно задать:

- Stack size - размер стека.
- Heap size – размер кучи (динамической памяти).



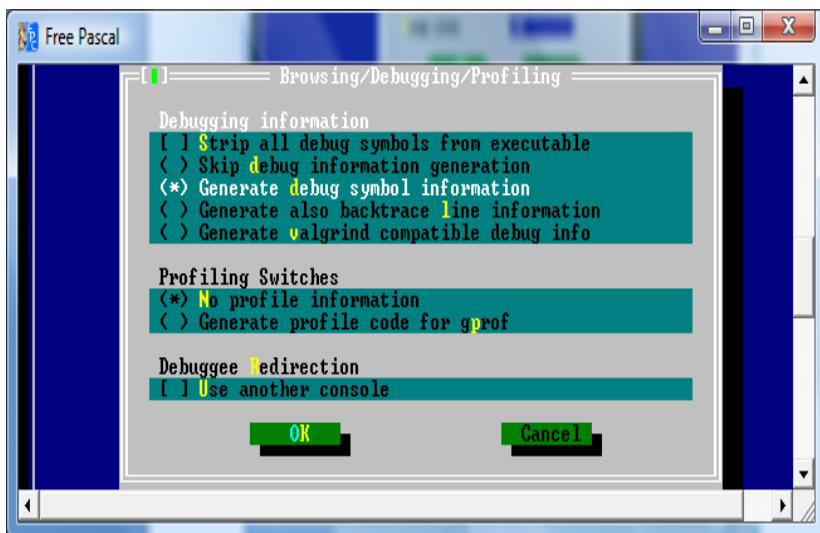
Команда Linker используется для компоновщика. Выводит окно диалога



с полями:

- Linking stage – этапы компоновки: вызов компоновщика, связь со статическим библиотеками.
- Preferred linking type – тип компоновки: платформа по умолчанию + связь с библиотеками статическими, динамическими или гибкими.

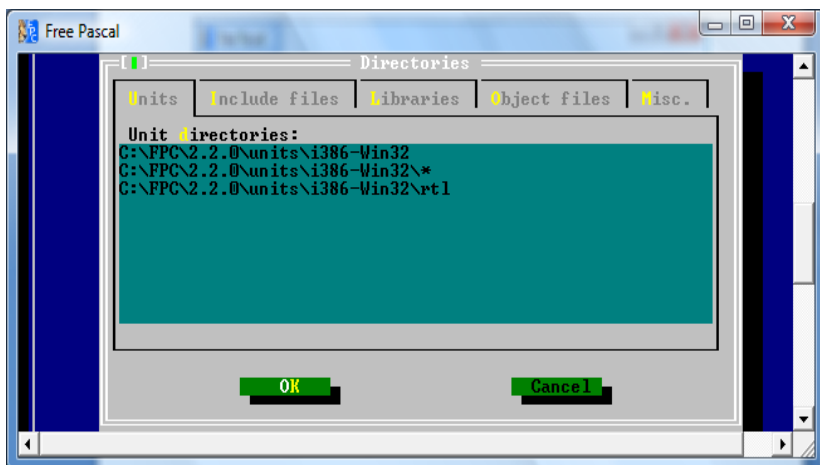
Команда Debugger используется для отладчика. Выводит окно диалога



с полями:

- Debugging information – что делать с отладочной информацией.
- Profiling Switches – создавать или нет информацию профилирования.
- Debuggee Redirection – как делать повторную отладку.

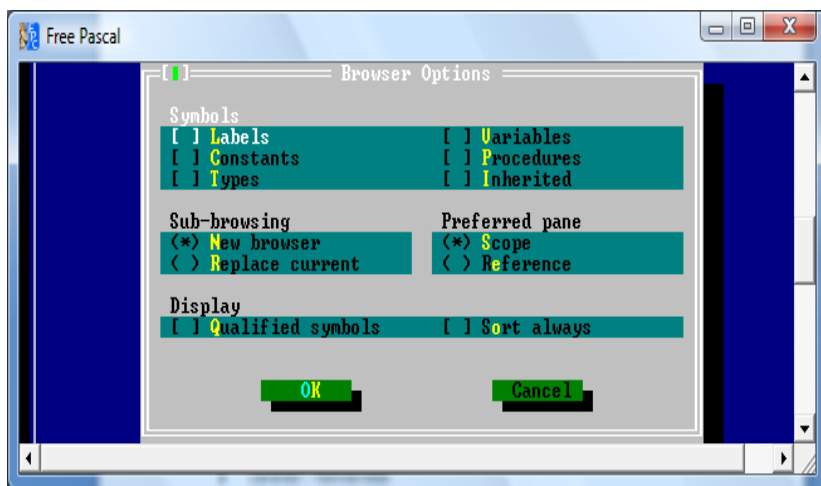
Команда Directories используется для выбора размера памяти. Выводит окно диалога



с закладками, в которых указаны каталоги, где расположены:

- Units - модули.
- Included files – включаемые файлы.
- Libraries – библиотеки.
- Object files – объектные файлы.
- Misc - разное.

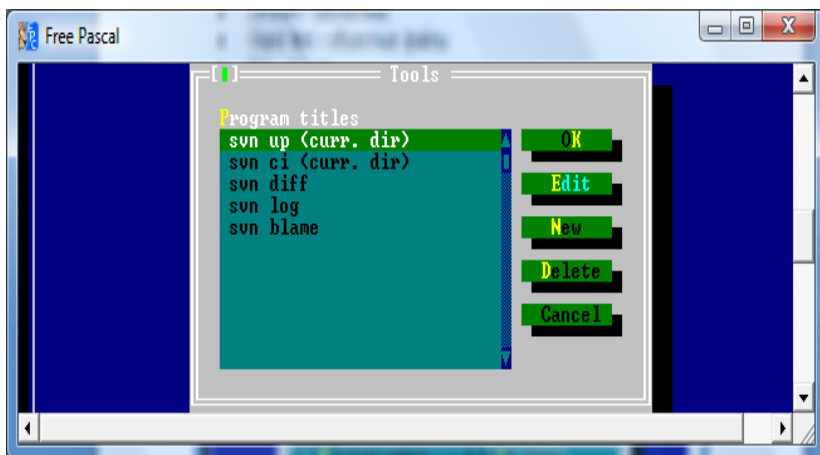
Команда Browser используется для выбора размера памяти. Выводит окно диалога



с полями:

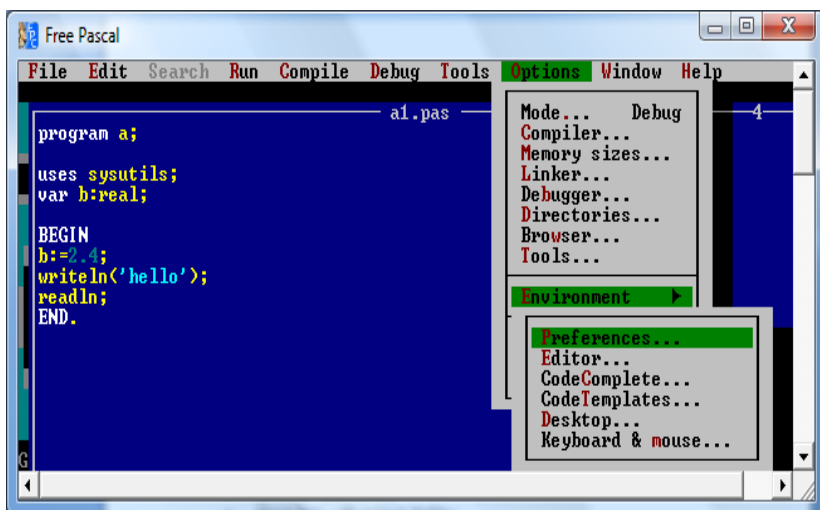
- Symbols – список выбора символов для просмотра.
- Sub-browsing – выбор дополнительного браузера.
- Preferred pane – выбор средства отображения (просмотр, ссылка).
- Display – как отображать символы.

Команда Tools используется для выбора размера памяти. Выводит окно диалога



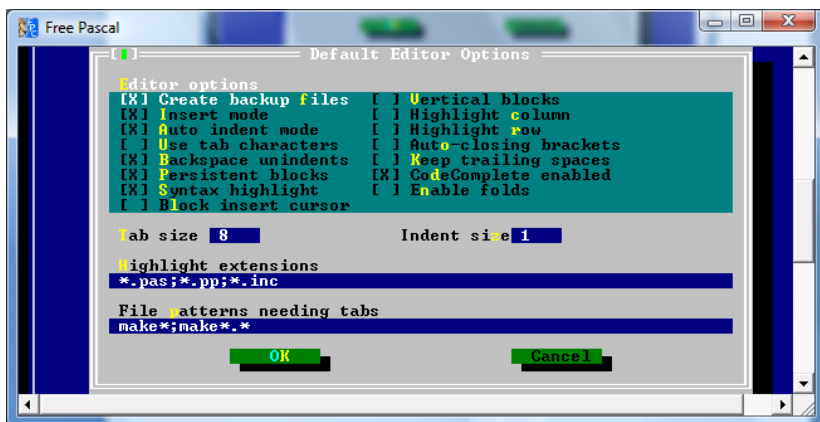
Со списком внешних инструментов. Инструменты можно редактировать, удалять и добавлять.

Команда Environment используется для средств окружения. При исполнении выводит список атрибутов средств окружения для изменения.

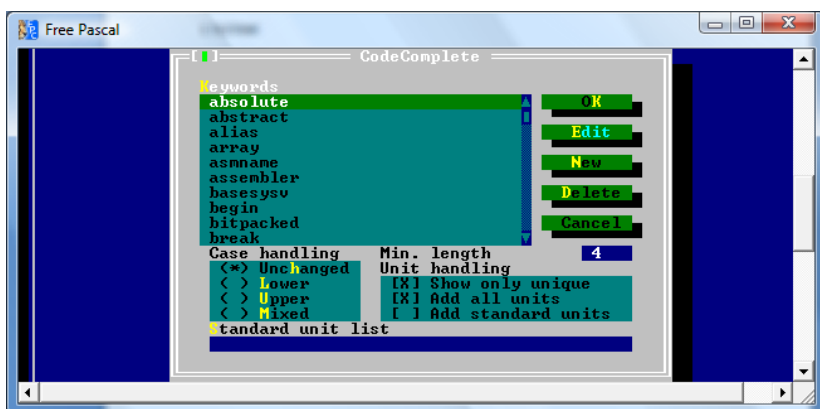


При выборе Preferences можно менять предпочтения.

При выборе Editor отображается окно диалога для выбора параметров редактора кода.

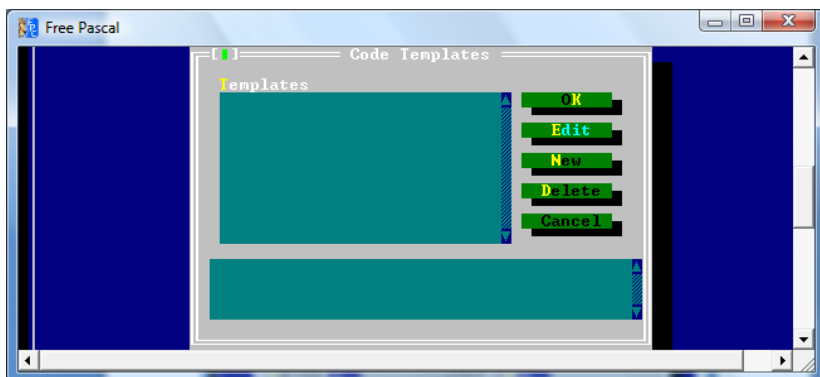


При выборе CodeComplete отображается окно диалога для выбора отображения ключевых слов языка.

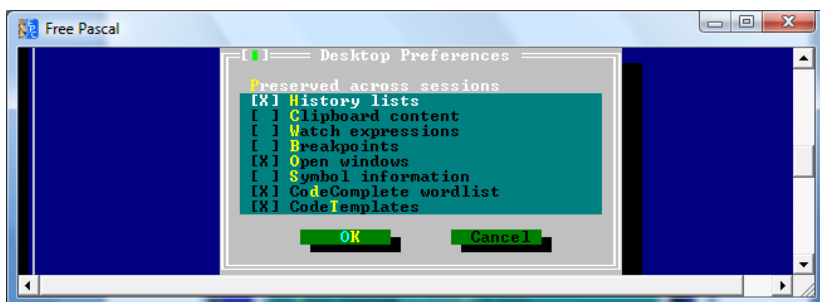


При выборе Code Template отображается окно диалога для редактирования шаблонов кода. Шаблон кода – это фрагмент кода с уникальным именем. Если нужно, то можно создавать новые шаблоны.

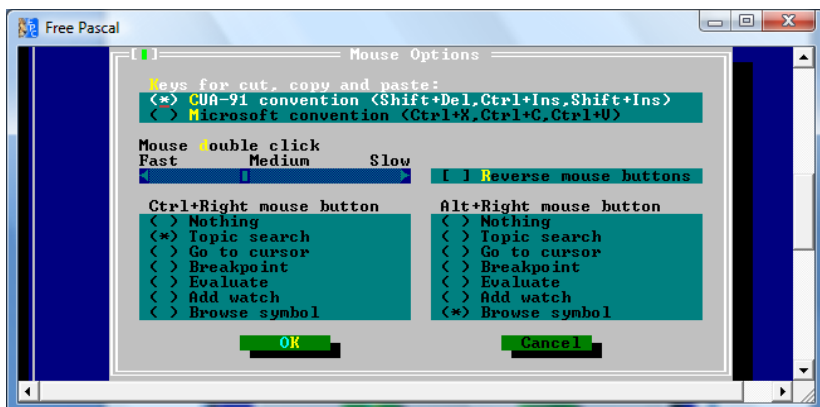




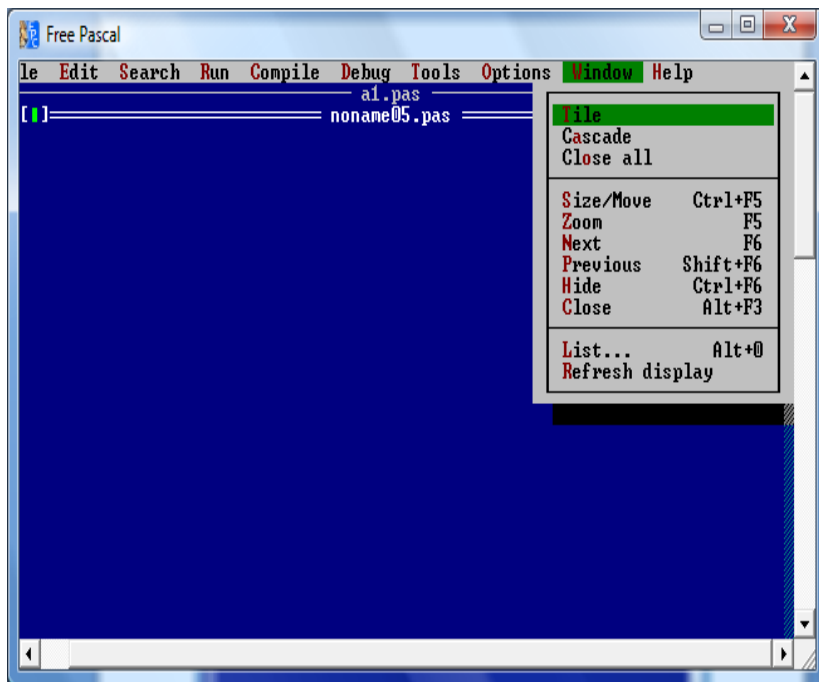
При выборе Desktop отображается окно диалога для выбора окон, отображаемых на рабочем столе.



При выборе Keyboard & mouse отображается окно диалога с выбором правила работы клавиатуры и манипулятора мышь.



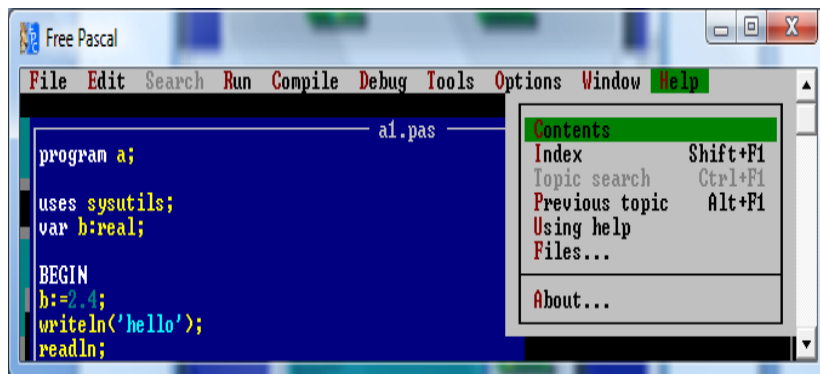
### 3.10. Пункт Windows (Окна)



Использует диалоговые команды работы с окнами:

Команда	Действие
Tile	Окна друг за другом.
Cascade	Окна каскадно.
Close all	Закреть все.
Size/Move	Размер/Передвинуть.
Zoom	Увеличить.
Next	Следующее окно.
Previous	Предыдущее окно.
Hide	Скрыть.
Close	Закреть.
List...	Список окон, в том числе скрытых.
Refresh display	Обновить дисплей.

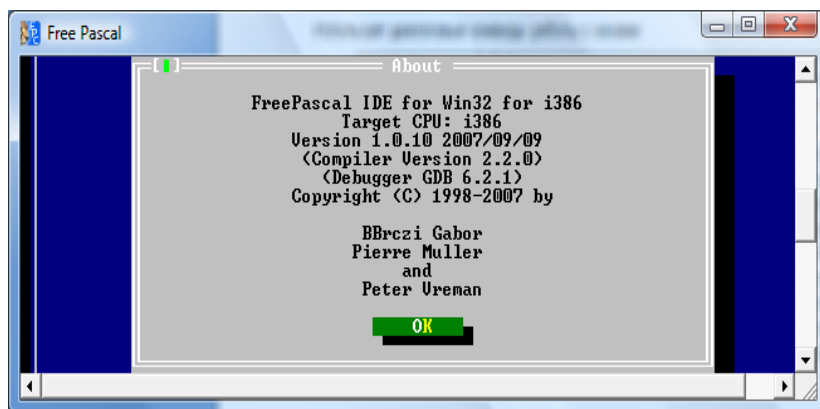
### 3.11. Пункт Help (Справка)



Использует диалоговые команды работы с справками:

Команда	Действие
Contents	Содержание.
Index	По индексу
Topic search	Поиск темы
Previos topic	Предыдущая тема
Using help	Использование справки
Files	Файлы справки
About	О программе

Команда Help=>About отображает окно. В нем сообщаются основные данные о ИСП Free Pascal.

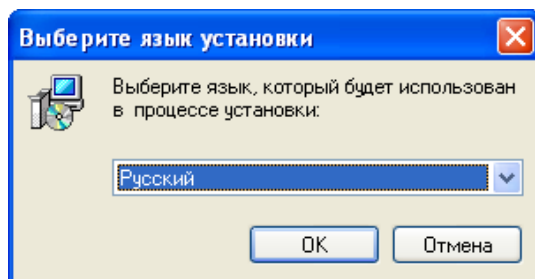




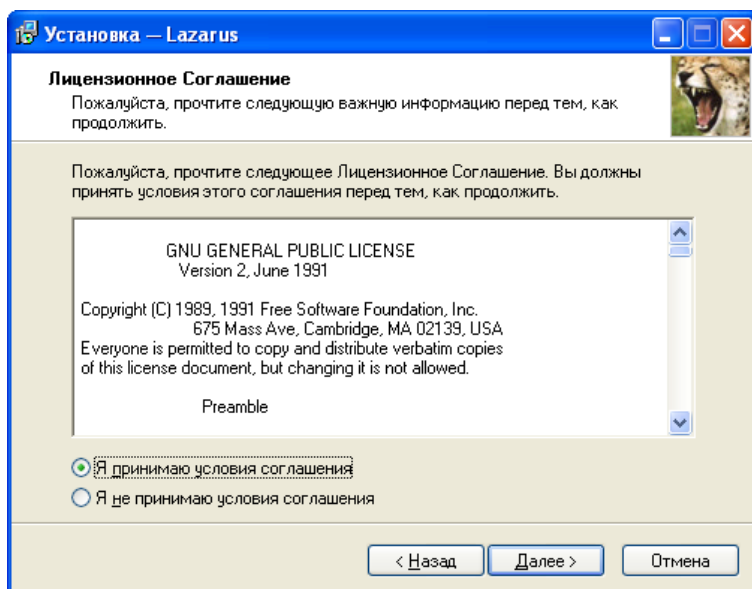
## 4. ИСП Lazarus

### 4.1. Установка ИСП Lazarus

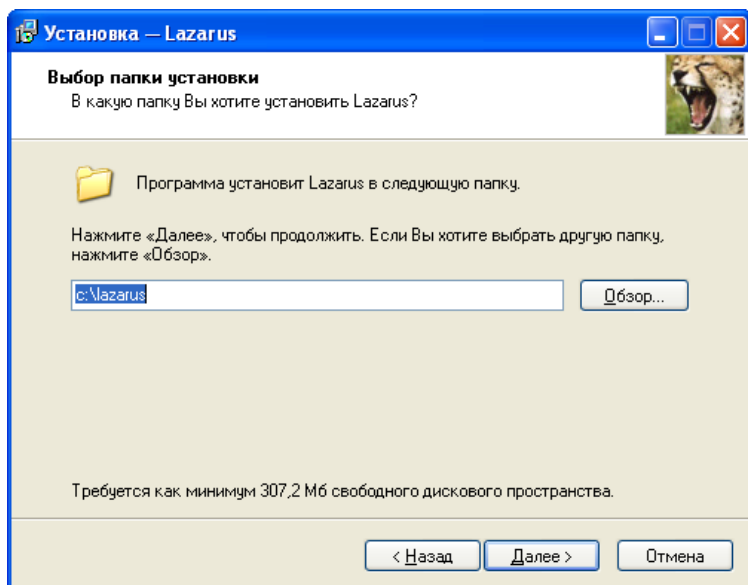
Установим ИСП Lazarus на наш компьютер. Для этого заходим на <http://www.FPC.ru/> и закачаем последнюю стабильную версию Lazarus (размер инсталляции около 40 МБайт) - Lazarus-0.9.24. Теперь начнём установку Lazarus.



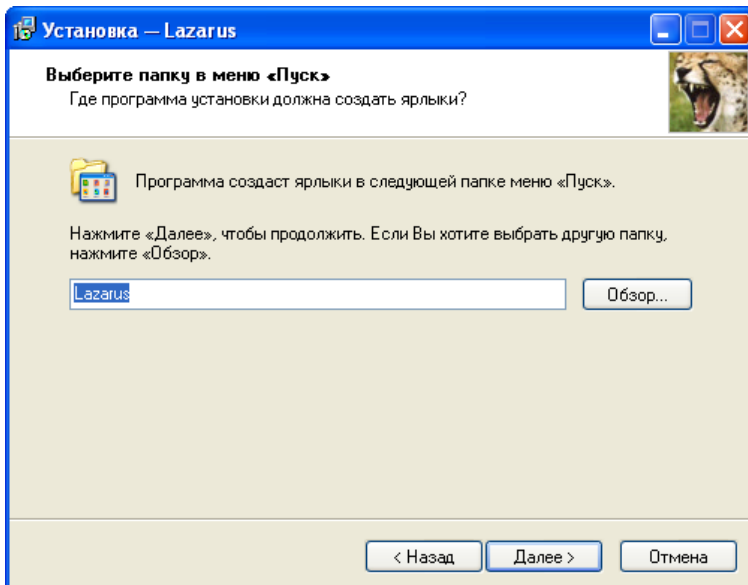
Выберем язык для установки.



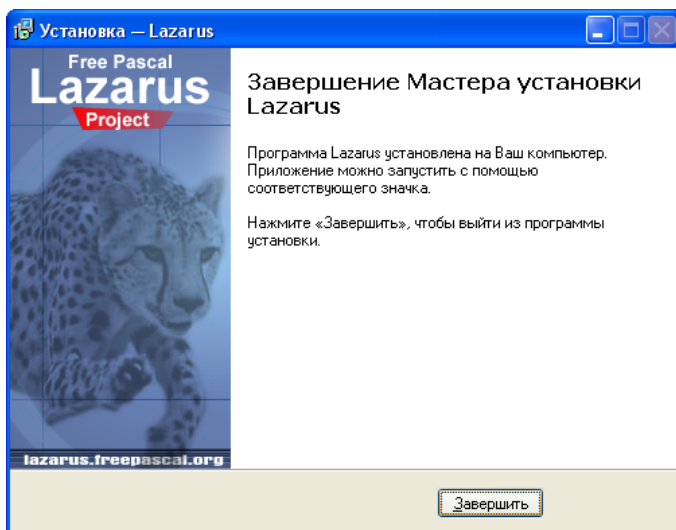
Ознакомимся с текстом лицензии и согласимся с ним.



Для удобства (если возможно) оставим такой же самый путь.



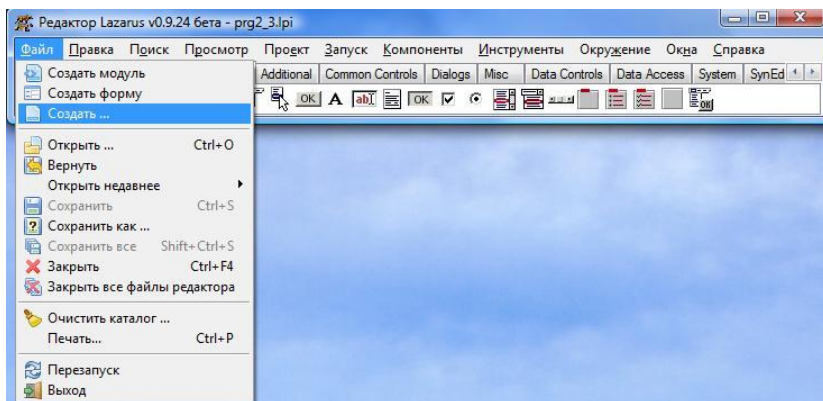
Назовём группу в меню и пойдём далее. Наконец, окно завершения установки.



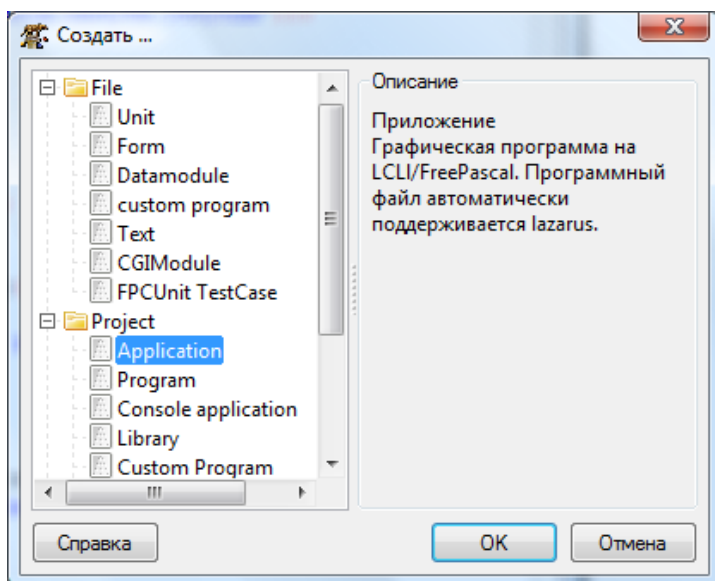
Как видите, среда разработки Lazarus сильно напоминает Delphi. Более того, многие компоненты похожи, но ряда компонент нет вообще, но появились и некоторые новые.

## 4.2. Вход в ИСР Lazarus

Запуск можно выполнить традиционно из меню Windows. После запуска отображается главное окно ИСР. В его главном меню нужно выбрать пункт Файл, в котором использовать команду Файл => Создать.

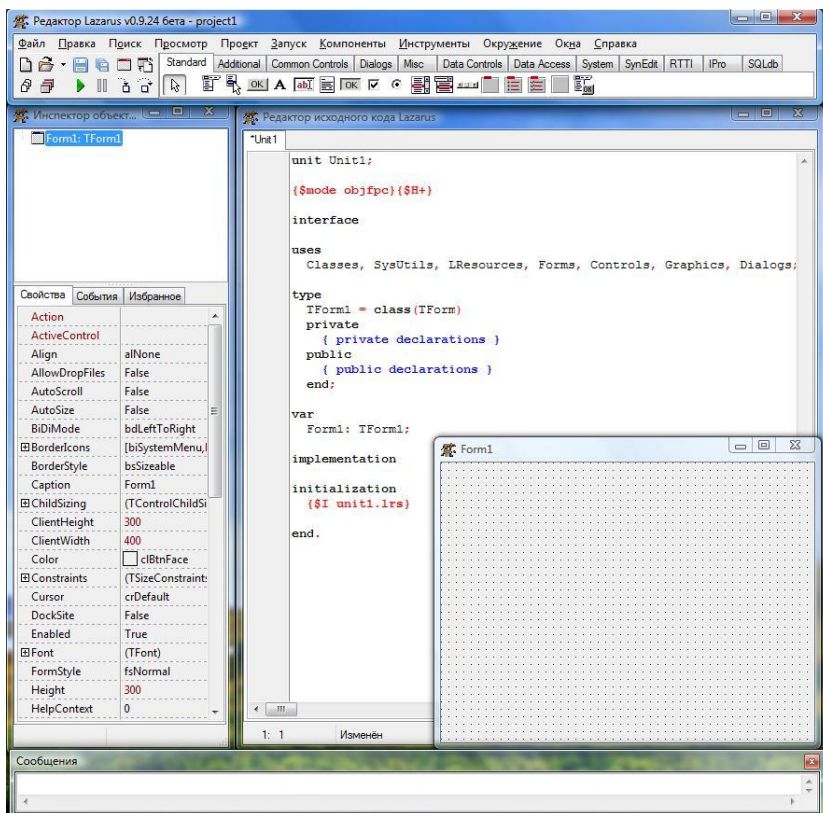


Появляется окно выбора, содержащее два поля. В левом поле выбирается тип проекта (например, Приложение - Application). В правом отображается описание типа проекта.



После этого отображаются остальные окна проекта. Для приложения это всего 5 окон, имеющее примерно такой вид.





Экран среды включает окна:

- Главное (окно проекта) – с именем по умолчанию Project1.
- Конструктор формы – с именем по умолчанию Form1.
- Редактор кода – модуль формы с именем по умолчанию Unit1.pas. Располагается там же, где Конструктор формы. По соглашению размещено под ним. Порядок расположения окон Конструктора формы и Редактора кода можно переключать клавишей F12.
- Инспектор объекта. В нем располагаются: Дерево объектов вверху и параметры выделенного объекта внизу на трех закладках (Свойства, События, Избранное).
- Сообщения. Располагается в нижней части.

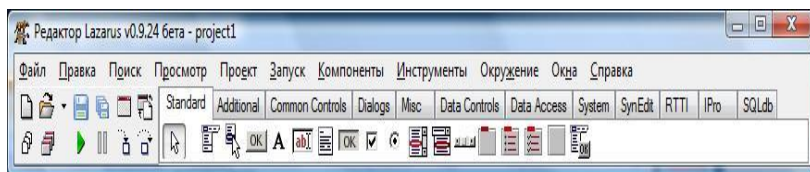
Все окна, кроме главного могут быть удалены или размещены произвольно. Выше описано наиболее целесообразное размещение окна, принятое в ИСР

по соглашению. Если Вы меняли расположение окон, то в следующем сеансе новое расположение будет повторено.

## 4.3. Главное окно Lazarus

### 4.3.1. Содержание окна

Управляет проектом создаваемой программы. При минимизации этого окна автоматически исчезают все дочерние окна.








Содержит:

- Заголовок. Традиционно он размещен в верхней строке окна, содержит имя программы и имя проекта слева и кнопки управления справа. По соглашению проект имеет имя Project<номер по порядку>. При создании проекта рекомендуется назначать проекту имя со смыслом.
- Главное меню. Размещено под заголовком и содержит перечень пунктов. При выборе пункта возникнет выпадающее меню пункта с перечнем доступных в нем команд.
- Панели инструментов, размещенные ниже главного меню слева. Содержат пиктограммы (значки) часто употребляемых команд. Панели позволяют быстро исполнять команды. Их набор можно настроить под себя.
- Панель Компонентов - панель инструментов для выбора стандартных компонент при конструировании формы. Она размещена справа от панелей инструментов. Чтобы отображать визуально большое число компонент, они разбиты по именованным категориям, значки компонент размещены на закладках категорий.

Все окна можно мышью перетаскивать в любое место экрана.

### 4.3.2. Панель инструментов

Команда	Назначение
	Создать модуль
	Открыть
	Создать форму
	Переключатель Форма/Модуль
	Сохранить
	Сохранить все
	Показать модули
	Показать формы
	Запуск
	Пауза
	Шаг со входом в подпрограммы
	Шаг в обход. Подпрограмма за один шаг.

### 4.3.3. Панель Компоненты















Компоненты представляют собой элементы, из которых конструируется видимое изображение, создаваемое работающей программой. Существует значительное количество компонентов, которые не создают видимого изображения, но тем не менее играют важную роль в тех или иных случаях. Правильнее думать о компонентах, как о заранее приготовленных для вас фрагментах программы, которые можно вставлять, если в этом есть необходимость, в разрабатываемую программу. В этом разделе приводится начальный обзор компо-




нентов, который даст вам самое общее представление о богатстве возможностей Lazarus.

Панель Компонент содержит набор страниц категорий компонент с закладками и линейкой прокрутки. По умолчанию включены все страницы. В таблице ниже полужирным шрифтом выделены наиболее часто употребляемые страницы.

Страница	Содержание
<b>Standard</b>	Стандартные интерфейсные элементы
<b>Additional</b>	Дополнительные компоненты
<b>Common Controls</b>	Обычные элементы управления
<b>Dialogs</b>	Стандартные диалоговые окна.
Data Controls	Компоненты для управления данными.
Misc	Разное.
Data Access	Компоненты доступа к базе данных.
System	Системные компоненты.
Syn Edit	Редакторы кода с визуальной поддержкой синтаксиса.
RTTI	Элементы для доступа к данным.
IPro	Элементы для системы IPro.
SQLdb	Элементы для SQL запросов к базам данным.

Страница STANDARD. На странице сосредоточены стандартные интерфейсные элементы, без которых не обходится практически ни одна программа.












Компонент	Назначение
 TMainMenu	Главное меню программы.
 TPopupMenu	Локальное меню. Вызывается правой кнопки мыши.
 TButton	Командная кнопка.
 TLabel	Метка. Для размещения однострочных надписей.
 TEdit	Однострочный текстовый редактор.
 TMemo	Многострочный текстовый редактор.
 TToggleBox	Зависимый переключатель.
 TCheckBox	Независимый переключатель.
 TRadioButton	Радиокнопка.
 TListBox	Список выбора.
 TComboBox	Комбинированный список выбора.
 TScrollBar	Линейка скроллинга (прокрутки).
 TGroupBox	Группа элементов.
 TRadioGroup	Группа радиокнопок.

	TCheckGroup	Группа независимых переключателей.
	TPanel	Панель для объединения нескольких компонентов.
	TActionList	Список действий.













Страница ADDITIONAL. В страницу помещены дополнительные компоненты, чтобы разнообразить вид диалоговых окон.



Компонент	Назначение
	TBitBtn Командная кнопка с надписью и пиктограммой.
	TSpeedButton Скоростная кнопка.
	TStaticText Статический текст.
	TImage Рисунок.
	TShape Фигура.
	TBevel Кромка.
	TPaintBox Инструмент рисования.
	TNotebook Произвольная таблица.
	TLabelledEdit Комбинация однострочного редактора и метки.
	TSplitter Разделитель.
	TTrayIcon Иконка.
	TCheckListBox Список множественного выбора.
	TScrollBar Блок с линейками прокрутки.
	TApplicationProperties Свойства приложения.
	TStringGrid Таблица строк.
	TDrawGrid Таблица рисунков.
	TPairSplitter Парный расщепитель.
	TColorBox Выбор цвета.
	TColorListBox Список цветов.
	TChart Диаграмма.

Страница Common Controls. Содержит интерфейсные элементы для ОС.

Компонент	Назначение
 TTrackBar	Ползунковый регулятор.
 TProgressBar	Индикатор процесса.
 TTreeView	Дерево выбора.
 TListView	Панель пиктограмм.
 TStatusBar	Панель статуса.
 TToolBar	Инструментальная панель.
 TUpDown	Цифровой регулятор.
 TPageControl	Набор панелей с закладками.
 TTabControl	Набор закладок.
 THeaderControl	Управляющий заголовок.
 TImageList	Набор рисунков.

Страница DIALOGS. Компоненты страницы Dialogs реализуют стандартные для Windows диалоговые окна.






Компонент	Назначение
 TOpenDialog	Открыть.
 TSaveDialog	Сохранить.
 TSelectDirectoryDialog	Выбрать каталог.
 TColorDialog	Цвет.
 TFontDialog	Шрифт.
 TFindDialog	Поиск.
 TReplaceDialog	Замена.
 TOpenPictureDialog	Открыть рисунок.
 TSavePictureDialog	Сохранить рисунок.
 TCalendarDialog	Календарь.
 TCalculatorDialog	Калькулятор.
 TPageSetupDialog	Установка параметров страницы.








	TPrintDialog	Печать.
	TPrinterSetupDialog	Настройка принтера.

Страница MISK. Содержит компоненты разного назначения.






Компонент	Назначение
	TColorButton Цветная кнопка.
	TSpinEdit Редактор целых чисел.
	TFloatSpinEdit Редактор дробных чисел.
	TArrow Стрелки.
	TCalendar Календарь.
	TEditButton Кнопка с редактором текста.
	TFileNameEdit Редактор имени файла.
	TDirectoryEdit Редактор каталога.
	TDateEdit Редактор данных.
	TCalcEdit Редактор с калькулятором.
	TFileListBox Список файлов.
	TXMLPropStorage TXMLProp накопитель.
	TIniPropStorage TIniProp накопитель.
	TBarChart Диаграмма полосковая.
	TButtonPanel Панель кнопок.
	TIDEDialogLayoutStorage TIDEDialog вывода.

Страница Data Controls. Содержит элементы для работы с данными.








Компонент		Назначение
	TDBNavigator	Навигатор TDB.
	TDBText	Текст TDB.
	TDBEdit	Редактор TDB.
	TDBMemo	Многострочный редактор TDB.
	TDBImage	Рисунок для TDB.

	TDBListBox	Список TDB.
	TDBComboBox	Комбинированный список TDB.
	TDBCheckBox	Переключатель TDB.
	TDBRadioGroup	Группа радиокнопок TDB.
	TDBCalendar	Календарь TDB.
	TDBGroupBox	Контейнер TDB.
	TDBGrid	Таблица TDB.




Страница Data Access. Содержит элементы для доступа к данным.

Компонент	Назначение
 TDataSource	Источник данных.
 TMemDataset	Память для данных.
 TSdfDataset	TSdf данные.
 TFixFormatDataset	Данные в фиксированном формате.
 TDbf	






Страница SYSTEM. На этой странице представлены компоненты, которые имеют различное функциональное назначение, в том числе компоненты, поддерживающие стандартные технологии межпрограммного обмена данными OLE (Object Linking and Embedding -связывание и внедрение объектов).

Компонент	Назначение
 TTimer	Таймер.
 TIdleTimer	Таймер простоя.
 TLazComponentQueue	Очередь компонент Lazarus.
 THTMLHelpDataBase	База данных справки в формате HTML.
 THTMLBrowserHelpViewer	Браузер просмотра справки в формате HTML.
 TProcess	Процесс.
 TSimpleIPCClient	Простой IPC клиент.

























	TSimpleIPCServer	Простой IPC сервер.
	TXMLConfig	Конфигуратор XML.
	TEventLog	Журнал событий.



Страница [SynEdit](#). Содержит редакторы с визуальной поддержкой синтаксиса.

Компонент	Назначение
 TSynEdit	Редакторы с визуальной поддержкой синтаксиса.
 TSynAutoComplete	Редактор с автозавершением.
 TSynExporterHTML	Редактор ExporterHTML.
 TSynMacroRecorder	Редактор MacroRecorder.
 TSynMemo	Многострочный редактор.
 TSynPasSyn	Редактор Паскаля.
 TSynFPCSyn	Редактор FPC.
 TSynCppSyn	Редактор C++.
 TSynJavaSyn	Редактор Java.
 TSynPerlSyn	Редактор Perl.
 TSynHTMLSyn	Редактор HTML.
 TSynXMLSyn	Редактор XML.
 TSynLFMSyn	Редактор LFM.
 TSynJUNIXShellScriptSyn	Редактор J UNIX Shell Script.
 TSynCssSyn	Редактор CSS.
 TSynPHPSyn	Редактор PHP.
 TSynTexSyn	Редактор Tex.
 TSynSQLSyn	Редактор SQL.
 TSynPhytonSyn	Редактор Phyton.
 TSynAnySyn	Редактор Any.
 TSynMultiSyn	Редактор Multi.










Страница [RTTI](#). Содержит интерфейсные элементы для доступа к данным.

Компонент	Назначение
 TTIEdit	Редактор TTI.
 TTIComboBox	Комбинированный список TTI.
 TTIButton	Кнопка TTI.
 TTICheckBox	Переключатель TTI.
 TTILabel	Метка TTI.
 TTIGroupBox	Контейнер TTI.
 TTIRadioGroup	Группа радиокнопок TTI.
 TTICheckGroup	Группа переключателей TTI.
 TTICheckListBox	Список переключателей TTI.
 TTIListBox	Список TTI.
 TTIMemo	Многострочный редактор TTI.
 TTICalendar	Календарь TTI.
 TTIImage	Рисунок TTI.
 TTIFloatSpinEdit	Спин-редактор чисел с плавающей точкой TTI.
 TTISpinEdit	Спин-редактор целых чисел TTI.
 TTITrackBar	Ползунок TTI.
 TTIProgressBar	Линейка прогресса TTI.
 TTIMaskEdit	Редактор с маской TTI.
 TTIColorButton	Цветная кнопка TTI.
 TTIMultiPropertyLink	Связь со свойствами TTI.
 TTIPropertyGrid	Таблица свойств TTI.
 TTIGrid	Таблица TTI.

Страница IPro. Содержит интерфейсные элементы для системы IPro.

Компонент	Назначение
 TipFileDataProvider	Провайдер данных.
 TipHtmlPanel	Панель HTML.

Страница SQLdb. Содержит элементы для SQL запросов к базам данным.

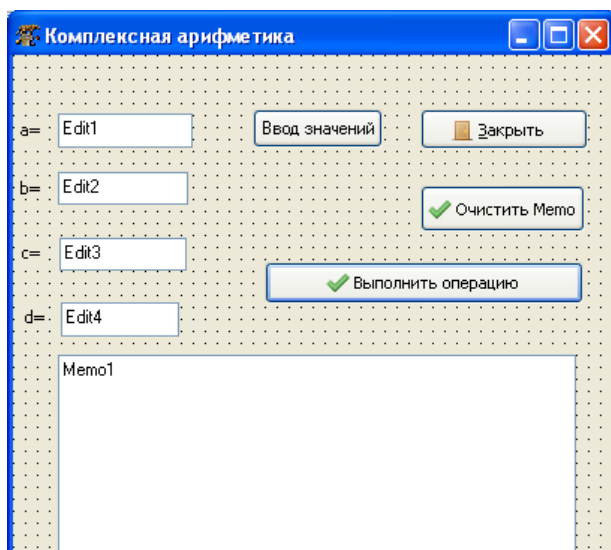
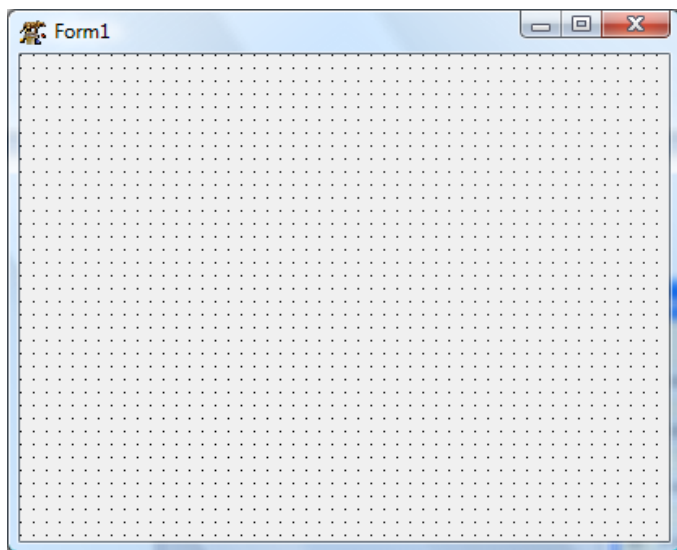
Компонент	Назначение
 TSQLQuery	SQL запрос.
 TSQLTransaction	SQL транзакция.
 TSQLConnection	SQL соединение.
 TOracleConnection	SQL соединение с Oracle.
 TODBCConnection	SQL соединение с ODBC.
 TMySQL40Connection	SQL соединение с MySQL40.
 TMySQL41Connection	SQL соединение с MySQL41.
 TMySQL50Connection	SQL соединение с MySQL50.
 TIBConnection	SQL соединение с IB.

#### 4.4. Окно Конструктора формы

Форма – заготовка для разрабатываемого приложения. Здесь должны размещаться компоненты. Содержит:

- Заголовок с именем по соглашению Form<номер по порядку>. При создании проекта форме лучше давать осмысленное имя, отражающее функцию приложения.
- Поле формы с координатной разметкой. В него заносятся компоненты, выбираемые из панели компонент.

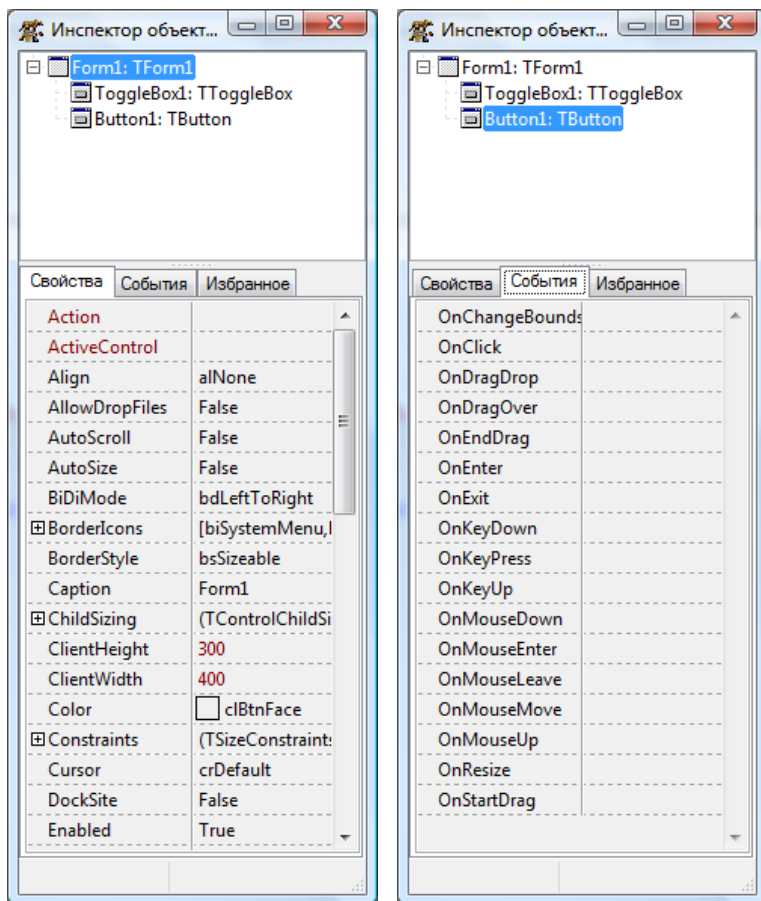
Вид при старте ИСР (сверху) и после заполнения формы компонентами (снизу).



## 4.5. Окно инспектора объектов

Обычно размещается слева от окна формы. Рекомендуется не заслонять окно другими окнами, так как оно часто требуется. В окне отображаются сведения о выделенном в форме компоненте. Содержит:

- заголовок,
- список объектов формы для выбора в виде дерева (вверху),
- закладку «Свойства» выделенного объекта (внизу слева),
- закладку «События» выделенного объекта (внизу в центре),
- закладку «Избранное» для выделенного объекта (внизу справа).



Свойства представлены с группированием по категориям, с возможностью раскрытия групп. Значок плюс/минус - признак группы. Значок плюс обозначает, что группа не раскрыта, а минус, что она раскрыта.

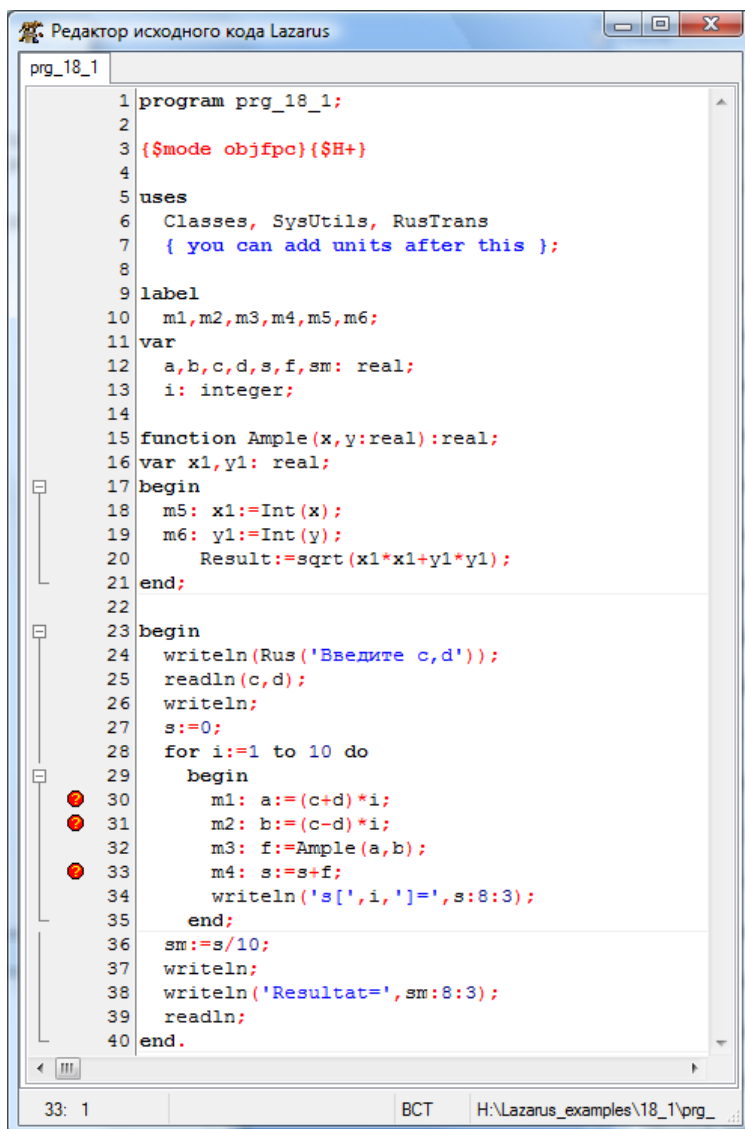
## 4.6. Окно редактора кода

Редактор кода предназначен для создания кода модуля формы программы. Обычно позиционируется там же, где окно Конструктора формы, чтобы не занимать на экране лишнего места. Переключение окон Форма/Модуль клавишей F12. Можно эти окна разместить рядом.

Текст кода выводится с синтаксическими выделениями. По соглашению:

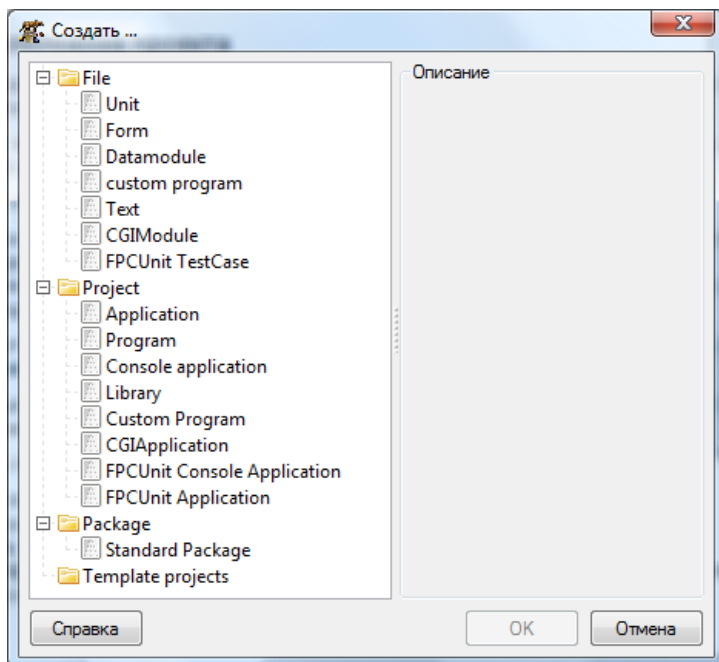
- Зарезервированные слова языка полужирным шрифтом.
- Комментарии курсивом и синим цветом.
- Директивы компилятора красным цветом.
- Строки кода можно номеровать.
- Слева от строки кода можно размещать знак останова (символ ?) для отладки.
- Возможно сворачивание кода по блокам. Для этого нужно щелкнуть по значку [-] в начале блока. Эта операция полезна для визуализации иерархии инструкций.

Этот редактор обладает большими возможностями по редактированию текстов, а так же возможностями подсветки синтаксиса, причём не только для Pascal и SQL семантики, но и других.



## 4.7. Структура программ Lazarus

Структура программы зависит от ее типа. Тип выбирается в меню **Файл**. Там можно выбрать модуль, форму или что-то другое командой **Файл => Создать**. Отображается окно выбора желаемой программы:



Создавать можно файлы:

File	Описание
Unit	Модуль FPC.
Form	Модуль с формой Lazarus LCL.
Datamodule	Модуль с данными.
Castom program	Программа пользователя.
Text	Текстовый файл.
CGIModule	Модуль данных для приложения с CGI интерфейсом.
FPCUnit TestCase	Программа тестирования модуля FPC.

Создавать можно проекты (рекомендуется):

Project	Описание
Application	Графическая программа.



	С формой.
Program	Программа на FPC. Есть поддержка нитей.
Console Application	Консольное приложение. Есть средства работы с командной строкой.
Library	Библиотека на FPC. Для Windows - *.dll.
Castom program	Программа пользователя. Для создания консольного приложения.
CGIApplication	Приложение с CGI интерфейсом. Встроенный тип CGI интерфейса.
FPCUnit Console Application	Консольный модуль. Вложенные средства тестирования.
FPCUnit Application	Модуль на FPC. Вложенные средства тестирования.

Наиболее часто применяются:

- Castom Program. Консольное приложение. Имитация работы под операционной системой DOS в режиме командной строки. Нет графики.
- Application. Это приложение, графическая программа под Windows.

Программа в ИСП Lazarus включает несколько файлов, объединенных в проект. Чтобы файлы разных проектов не смешивались, рекомендуется для каждого проекта создавать свою папку. Количество и типы файлов проекта зависят от вида программы и используемых режимов работы при конструировании программы.

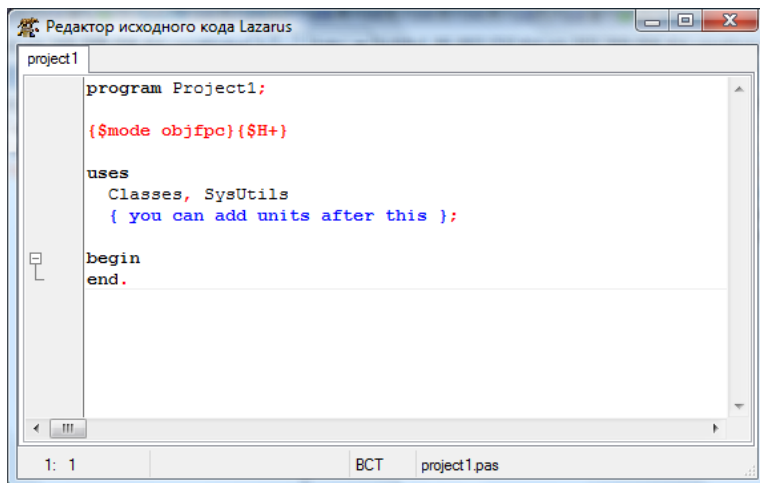
#### 4.7.1. Консольное приложение

Проект содержит файлы:

- Исходник на языке FPC (с расширением .pas).
- Информация о проекте (с расширением .lpi). lpi = Lazarus Project Information, Это конфигурации проекта. Создается ИСП автоматически.
- Компилированный (с расширением .compiled). Содержит конфигурации, нужные для формирования объектного файла. Создается ИСП автоматически.
- Объектный код проекта (с расширением .o). o = Object. Код нужен для сборки проекта. Создается ИСП автоматически.
- Исполняемый файл (с расширением .exe). exe = Execution. Создается ИСП автоматически.

Автоматически создаются и резервные копии файлов, помещаемые во вложенную папку проекта - \backup.

При создании консольного приложения ИСР автоматически создает шаблон программы. Код программы нужно заносить между блочными скобками begin..end.



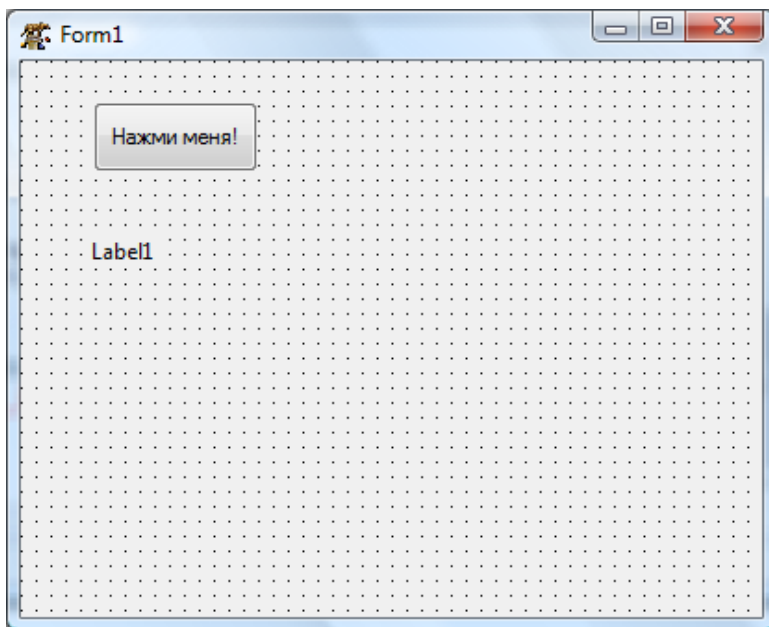
## 4.7.2. Приложение

Проект содержит файлы:

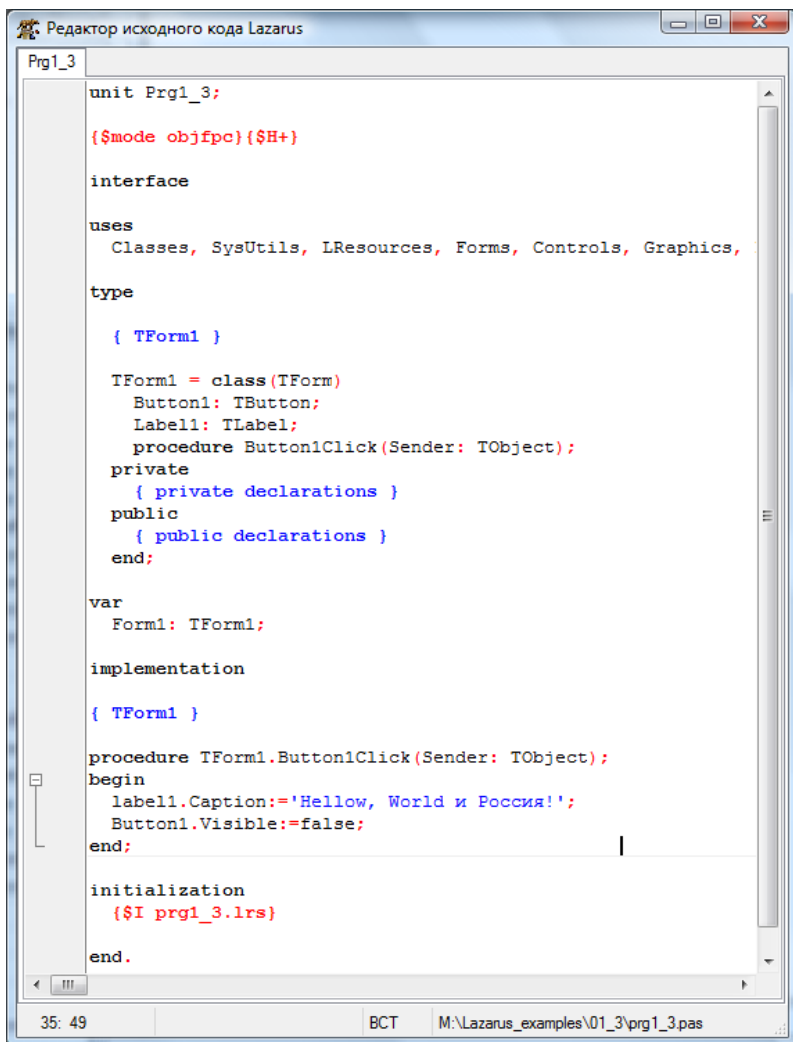
- Код проекта (с расширением .lpr). lpr = Lazarus Project на языке FPC. Создается ИСР автоматически.
- Информация о проекте (с расширением .lpi). lpi= Lazarus Project Information, Это конфигурации проекта. Создается ИСР автоматически.
- Описание формы (с расширением .lfm). lfm = Lazarus form. Создается ИСР автоматически.
- Модуль (с расширением .pas). Код модуля на языке FPC.
- Компилированный (с расширением .compiled). Содержит конфигурации проекта, нужные для формирования объектного файла проекта. Создается компилятором автоматически.
- Ассемблерный (с расширением .ppu). ppu = p p unit. Создается ИСР автоматически при компиляции..
- Объектный код модуля (с расширением .o). o = Object. Код нужен для сборки проекта. Создается ИСР автоматически.
- Объектный код проекта (с расширением .lpr). lpr = Lazarus Project. Создается ИСР автоматически.

- Ресурсы (с расширением .lrs). lrs = Lazarus Resource. Курсоры, иконки и др. Создается ИСР автоматически..
- Исполняемый файл (с расширением .exe). exe = Execution. Создается ИСР автоматически

Автоматически создаются и резервные копии файлов, помещаемые во вложенную папку проекта - \backup. Для приложения ИСР автоматически создает файл проекта (.lpr) и файл шаблона кода модуля формы. Они размещаются на разных страницах. Код проекта по умолчанию не отображается, так как его редактировать не рекомендуется.



В шаблон код модуля формы программист должен добавить функциональность.



## 4.8. Компиляция и выполнение проекта

При компиляции файла проекта создается готовый к выполнению файл, которым может быть приложение с расширением \*.exe или динамически связываемая библиотека (DLL - Dynamic Linked Library) с расширением \*.dll.

Приложение является автономным и не требует при своей работе дополнительных файлов. Если в приложении используются внешние файлы, то они должны быть доступны.

Запуск компиляции осуществляется командой Запуск => Быстрая компиляция.

При компиляции выполняются действия:

- Компилируются файлы всех модулей проекта, содержание которых изменилось после предыдущей компиляции. Для каждого модуля создаются два файла: ассемблерный с расширением (.pri) и объектный с расширением (.o).
- Если в модуль были внесены изменения, то перекомпилируются и все модули, ссылающиеся на него во фразе uses.
- После компиляции всех модулей проекта компилируется файл проекта.

Вместо компиляции командой Запуск => Сборка может быть выполнена сборка проекта. При этом компилируются все файлы проекта независимо от вносимых в них изменений.

Запуск проекта на исполнение осуществляется командой Запуск => Запуск, кнопкой из панели инструментов или клавишей F9. Если в файлы проекта вносились изменения, то предварительно выполняется компиляция и сборка проекта.

Особенности запуска приложений из ИСР:

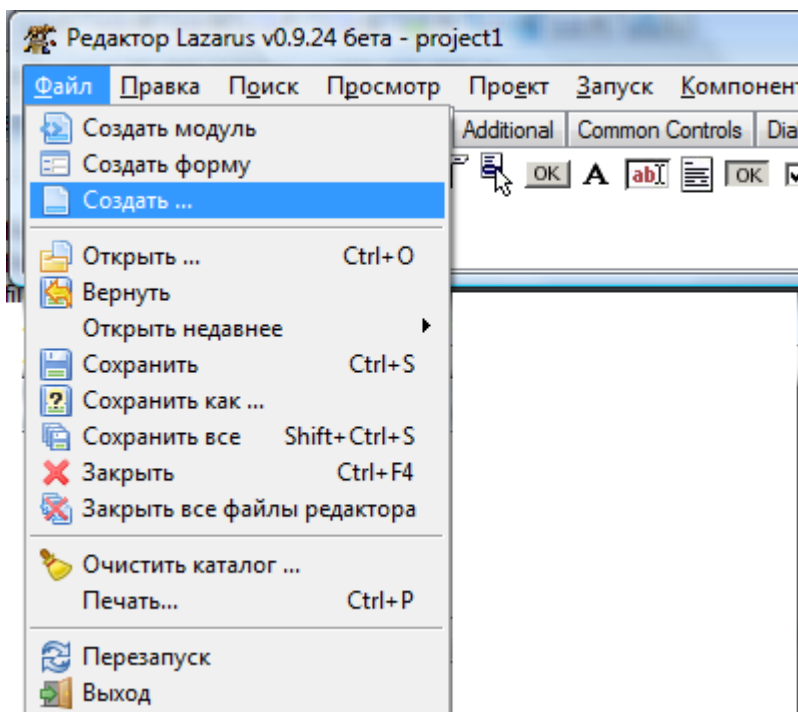
- Нельзя запустить вторую копию.
- Продолжить разработку проекта можно после завершения работы приложения.
- При зависании приложения приостановить его работу нужно командой Запуск => Останов (или клавишами Ctrl F2).
- Встроенный отладчик обнаруживает ошибки. Чтобы исключить реакцию отладчика на ошибки следует запускать исполняемый файл не из ИСР, а автономно.

## 4.9. Главное меню

В главном меню доступны все команды ИСР. Наиболее часто употребляемые команды дублируются в панели инструментов.

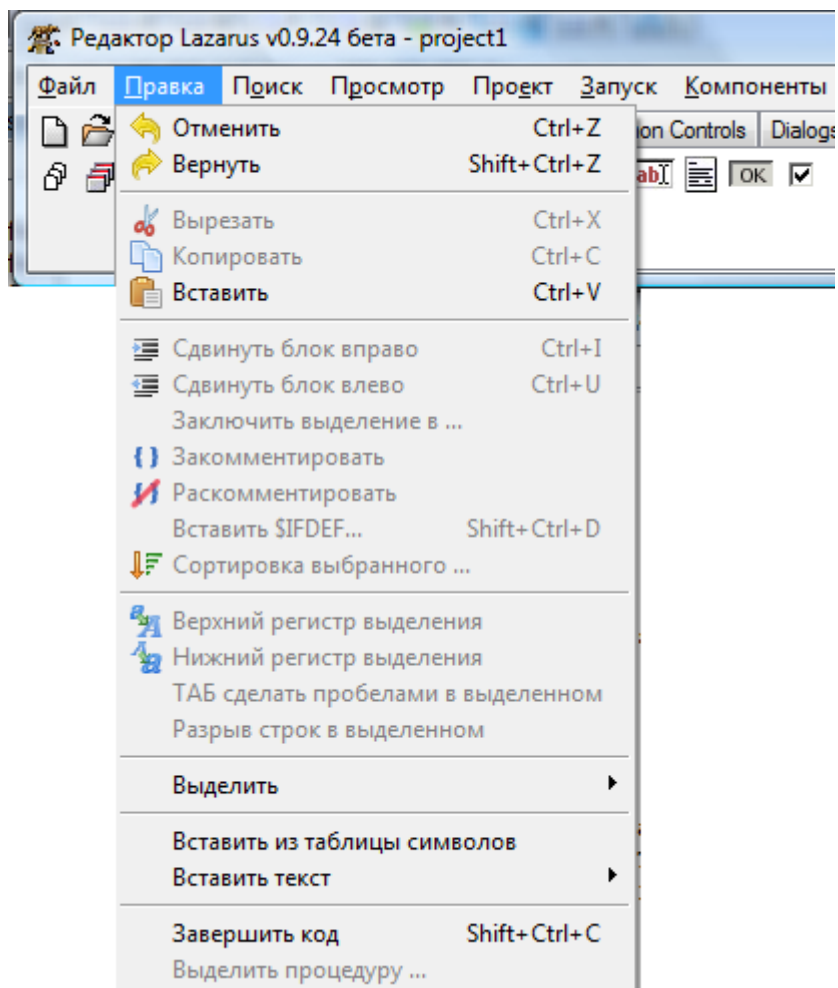
### 4.9.1. Пункт Файл

Использует обычные диалоговые команды работы с файлами.

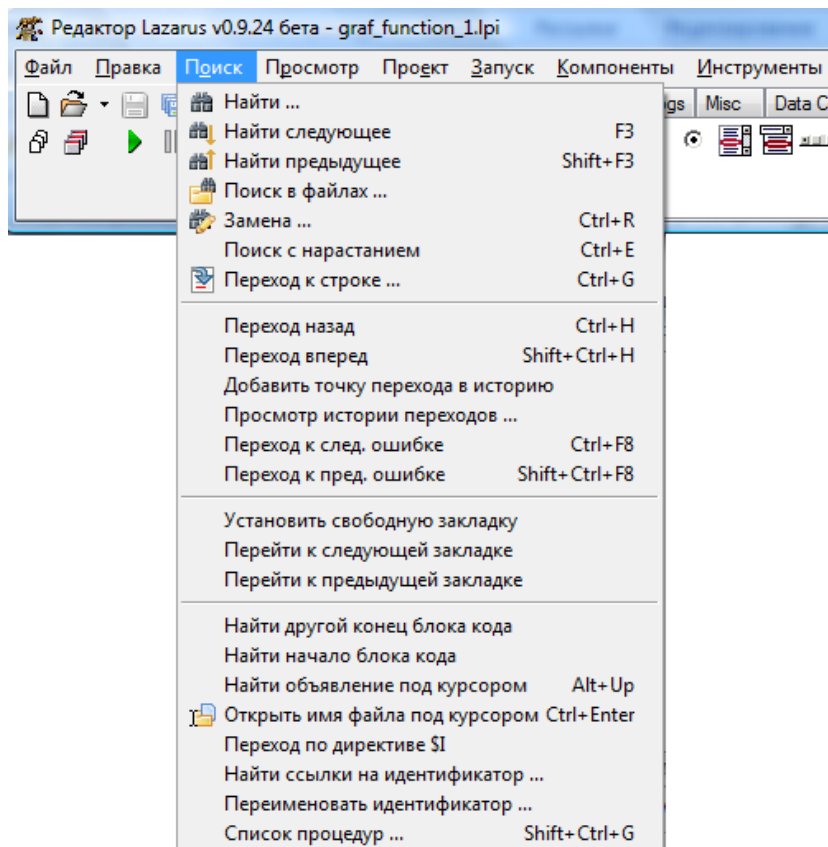


## 4.9.2. Пункт Правка

Использует диалоговые команды редактирования. Одни предназначены для Редактора кода, другие - для Конструктора формы.

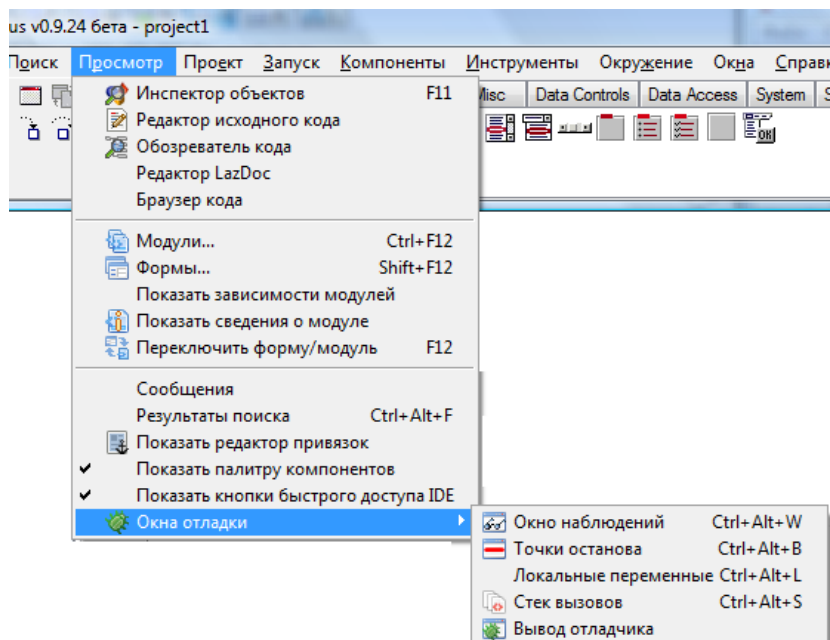


### 4.9.3. Пункт Поиск

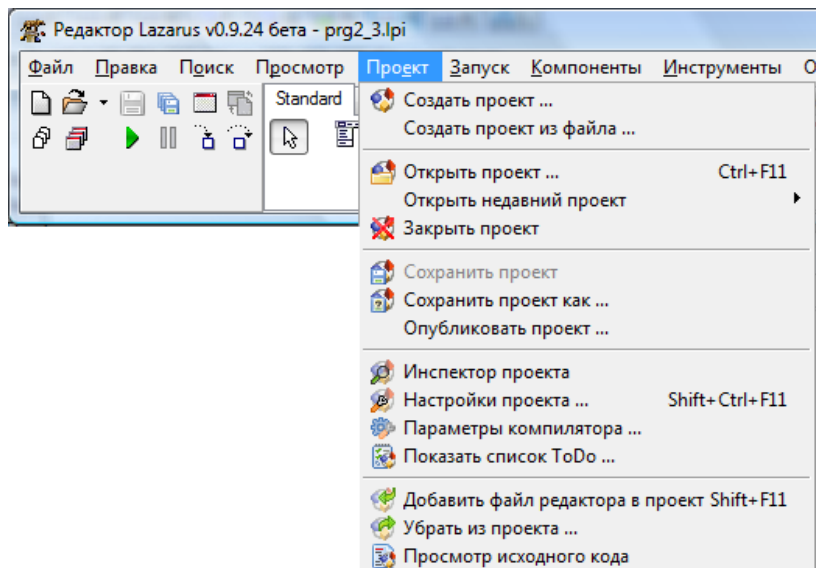




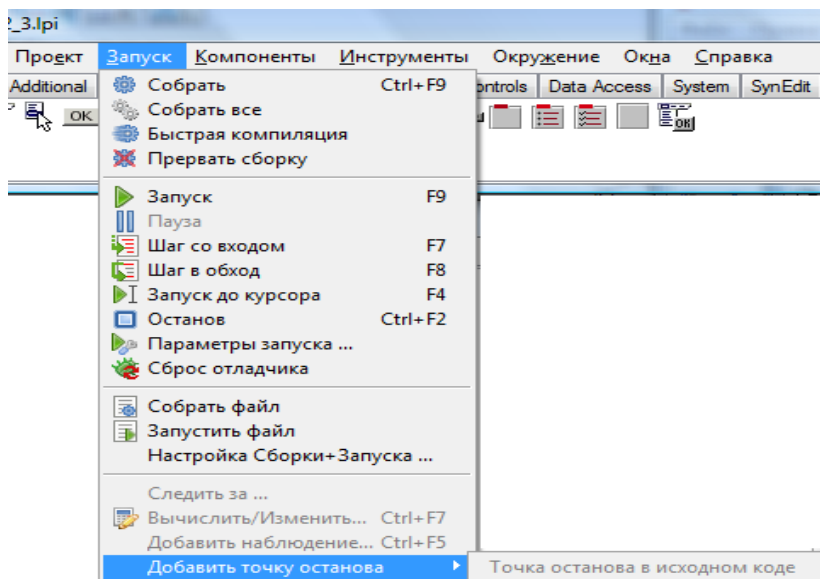
## 4.9.4. Пункт Просмотр



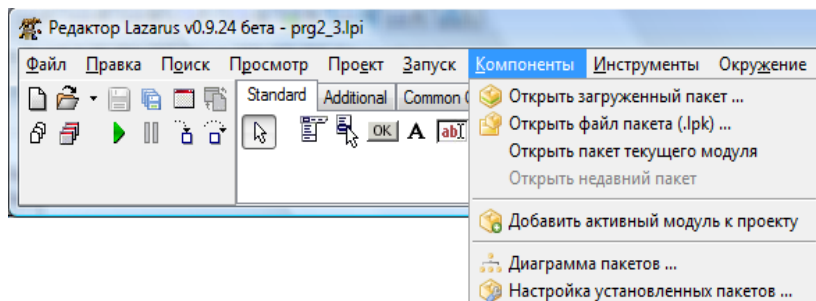
#### 4.9.5. Пункт Проект



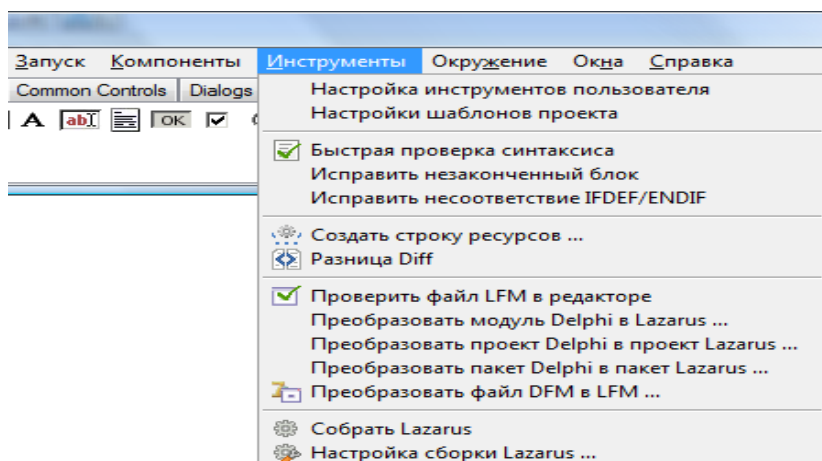
#### 4.9.6. Пункт Запуск



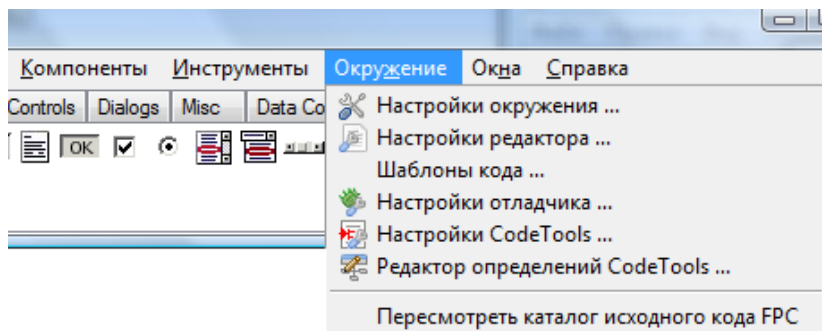
### 4.9.7. Пункт Компоненты



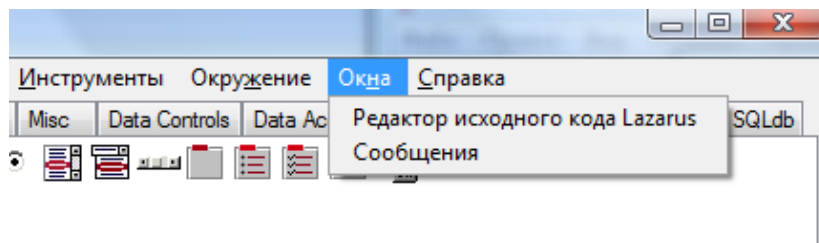
### 4.9.8. Пункт Инструменты



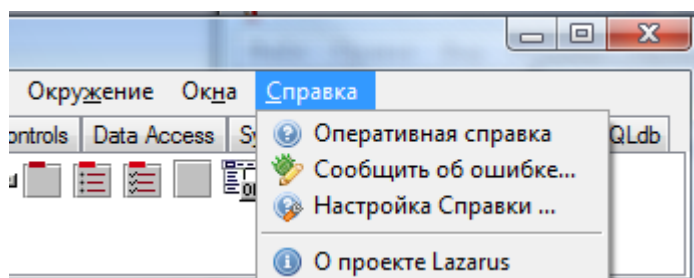
### 4.9.9. Пункт Окружение



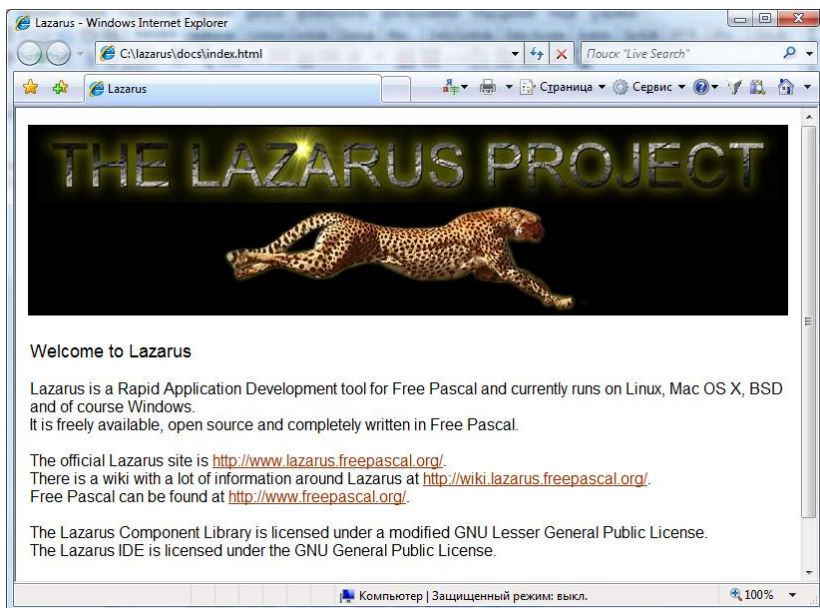
#### 4.9.10. Пункт Окна



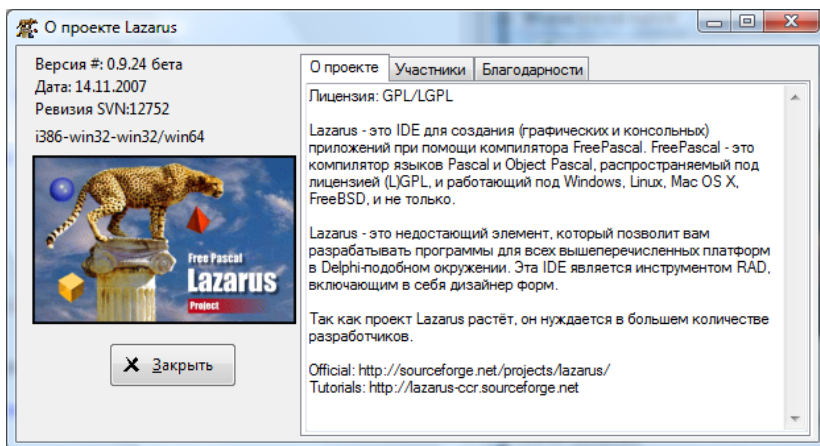
#### 4.9.11. Пункт Справка



Команда Справка => Оперативная справка отображает окно. В нем сообщаются основные данные о проекте Lazarus.



Команда Справка => О проекте Lazarus выводит более подробные данные о проекте Lazarus..



## 4.10. Модули в составе ИСП Lazarus

В ИСП Lazarus встроен FPC в полной версии. ИСП использует все его модули. Кроме того в ИСП определены дополнительные модули для работы с компонентами и графикой. Основные модули:

Модуль	Описание
Forms	Формы.
LResources	Ресурсы Lazarus.
Controls	Управление.
Dialogs	Диалоги.
Graphics	Графика.

## 4.11. Графика

В ИСП Lazarus графика поддерживается специальными классами и компонентами. Модуль Graphics включает классы-потомки для работы с графикой с графическим интерфейсом. Объекты этих классов участвуют в операциях ввода/вывода графики. Основные операции рисования определены в классе TCanvas (холст).

### 4.11.1. Класс TCanvas - холст

Объекты, которым доступны графические возможности, имеют свойство TCanvas (одновременно это класс, имеющий и свои свойства). Холст это прямоугольная область, которая состоит из отдельных точек (пикселей), каждая из которых может иметь свой цвет. Пиксель имеет координаты:

- по горизонтали X,
- по вертикали Y.

Отсчет от левого верхнего угла (0, 0). Графический объект - совокупность графических примитивов:

- точки,
- линии,
- фигуры.

Для их вычерчивания используются методы объекта Canvas. Их формат:

Object.Canvas.Method;

Если в графическом объекте прорисовывается много графических примитивов, то целесообразно включить их в оператор With. Пример:

```
with Object.Canvas do  
begin
```

```

Method1;
{.....}
Method N;
end;

```

Холст имеет свойства, часть из которых одновременно являются классами.

Класс	Описание
TPen	Перо. Рисует линии.
TBrush	Кисть. Закрашивает замкнутые области.
TFont	Шрифт.
TStyle	Стиль.
TColor	Цвет.

Основной объект со свойством Canvas – форма. Есть и другие компоненты, имеющие свойство Canvas. Например, Label.

Внимание. Холст формы и холст метки на форме разные вещи.

Для навигации по холсту определены классы:

- TPoint - точка, задаваемая двумя координатами (x, y).
- TRect - прямоугольная область, задаваемая парой точек (APoint1 - левый верхний и APoint2 - правый нижний углы области), либо четырьмя границами области (левая - ALeft, верхняя - ATop, правая - ARight, нижняя - ABottom).

Экземпляр точки APoint создается функцией:

```
APoint:=Rect(x, y);
```

Экземпляр области ARect создается функцией, которая имеет две реализации.

При использовании точек:

```
ARect:=Rect(APoint1, APoint2);
```

При использовании границ:

```
ARect:=Rect(ALeft, ATop, ARight,ABottom);
```

### 4.11.2. Вывод текста

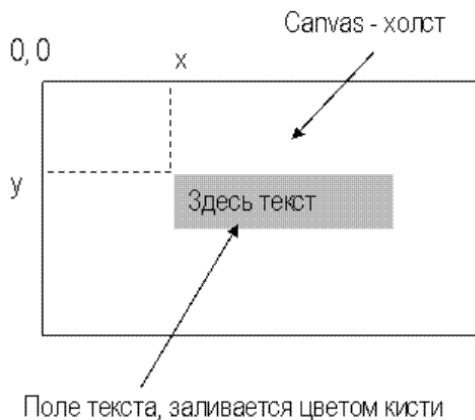
Для вывода текста используются процедуры:

Вызов	Действие
TextOut (x, y, 'строка текста')	Вывод строки S в позицию (x,y) формы. Выводится вся строка.
TextRect (ARect, x, y, 'строка текста')	Вывод строки S в область ARect. Не помещающийся текст обрезается.

Процедура

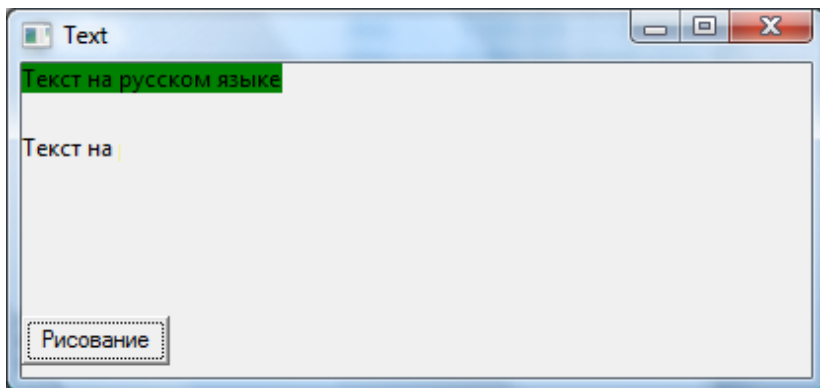
```
Object.Canvas.TextOut(x,y, 'Text');
```

Object - имя объекта, куда выводится строка текста (форма или компонент).



Здесь  $x$ ,  $y$  - координаты левого верхнего угла области вывода текста. Текст выводится шрифтом, определенным свойством Font объекта.

В процедуре TextRect ограничиваются размеры области для вывода текста.



### 4.11.3. Простые графические примитивы

Информация о цвете каждой точки холста содержится в свойстве Canvas.Pixels, представляющем двумерный массив данных типа TColor. Этот тип позволяет установить любой цвет точки, как комбинацию интенсивностей цветов RGB (R - Red, G - Green, B - Blue). Для точки холста с координатами  $x$ ,  $y$  можно задать цвет, используя выражение:



```
Pixels[x,y]:=<цвет>;
```

Это приводит к рисованию на холсте точки.

#### 4.11.3.1. Линия

Отрезок прямой линии отображаются с помощью процедур:

```
MoveTo(x,y);    // Указатель в начальную точку.  
LineTo(x,y);    // Линия до точки, начиная от текущей.
```

Линии через множество точек отображаются с помощью процедур:

```
PolyLine(Points:TPoint;          // Ломанная линия.  
PolyBezier(Points:TPoint,false,false); // Кривая Безье, фрагменты из 3 точек.
```

Points - массив точек типа TPoint, каждая из которых имеет координаты (x, y).

Пример. Рисование линий.

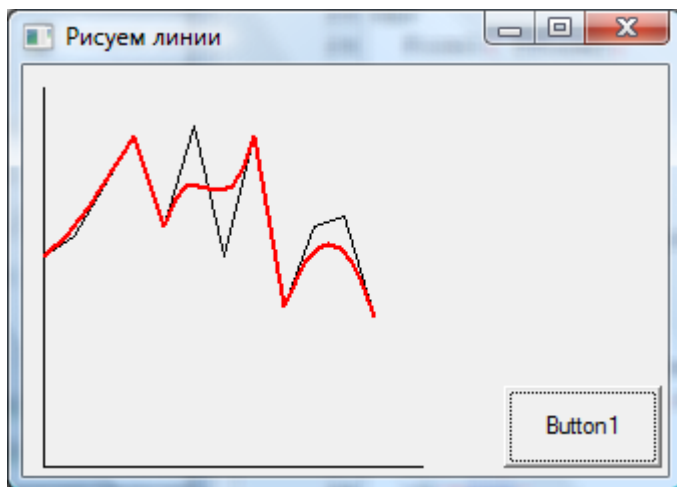
```
unit DrawLine_  
{ $mode objfpc } { $H+ }  
interface  
uses  
    Classes, SysUtils, LResources, Forms, Controls, Graphics, Dialogs, StdCtrls;  
type  
    TForm1 = class(TForm)  
        Button1: TButton;  
        procedure Button1Click(Sender: TObject);  
    end;  
var  
    Form1: TForm1;  
Implementation  
  
procedure TForm1.Button1Click(Sender: TObject);  
const  
    count=10;  
var  
    i,dx,dy,x0,y0:integer;  
    graphic:array[1..count] of TPoint;  
begin  
    x0:=10;  
    y0:=200;  
    dx:=15;  
    dy:=5;  
    for i:=1 to count do
```

```

begin
    graphic[i].x:=x0+(i-1)*dx;
    graphic[i].y:=y0-random(40)*dy;
end;
with Form1.Canvas do
begin
    MoveTo(x0, y0);           // Ось y
    LineTo(x0,10);
    Line(x0, y0, x0+190, y0); // Ось x
    PolyLine(graphic);        // Ломанная линия
    Pen.Color:=clRed;
    Pen.Width:=2;
    PolyBezier(graphic,false,false); // Кривая Безье
end;
end;

initialization
{$I drawline_.lrs}
end.

```



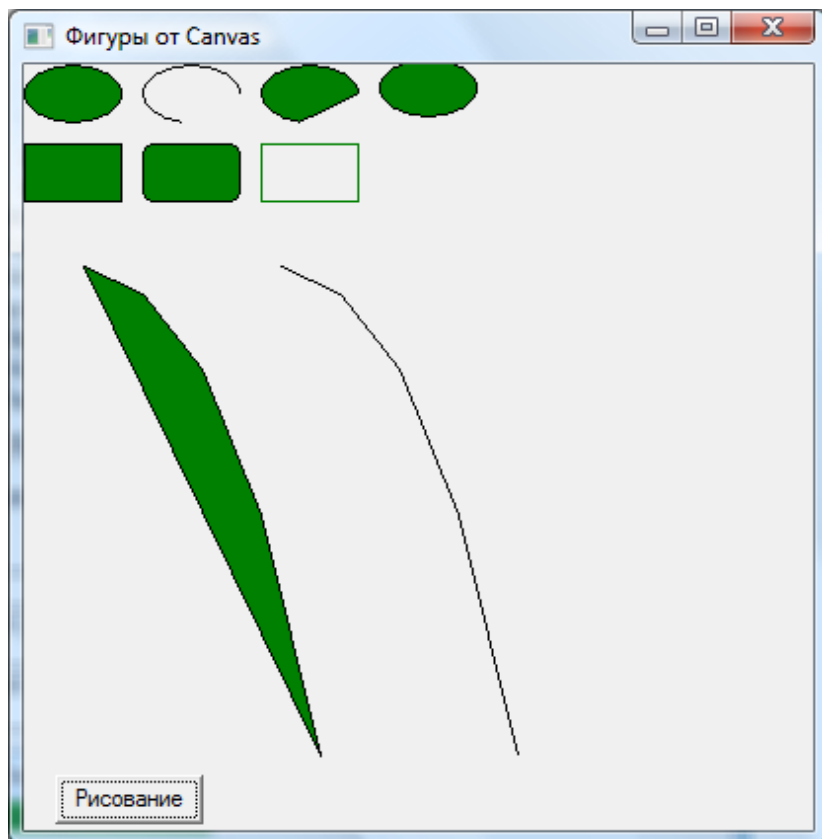
Ось X рисуется процедурой Line, ось Y процедурами MoveTo и LineTo. Графики нарисованы разными стилями. Черный цвет – ломанная линия, красный жирный – кривая Безье.

#### 4.11.4. Фигуры

Фигура вписывается в прямоугольную область с координатами углов:

- $x_1, y_1$  - левый верхний,
- $x_2, y_2$  - правый нижний.

Фигура рисуется линиями, цвет и стиль которых определяется свойством Pen. Цвет и стиль заполнения фигуры определяется свойством Brush.



Для рисования фигур используются процедуры:

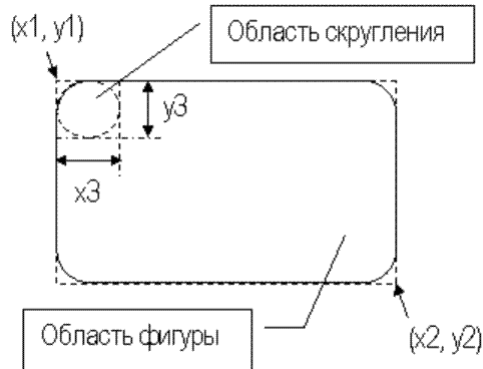
Вызов	Действие
Arc(X, Y, StAngle, Rad)	Дуга окружности. <ul style="list-style-type: none"><li>• Rad – радиусом,</li></ul>

	<ul style="list-style-type: none"> <li>• (X,Y) – центр,</li> <li>• StAngle – начальный угол.</li> </ul>
Bar3D(X1, Y1, X2, Y2, Depth,Top)	Параллелепепед. <ul style="list-style-type: none"> <li>• (X1,Y1) и (X2,Y2) углы передней грани,</li> <li>• Depth – глубина,</li> <li>• Top – признак объемности.</li> </ul>
Ellipse(X, Y, StAngle, EndAngle, XRad, YRad)	Часть эллипса. <ul style="list-style-type: none"> <li>• (X,Y) – центр,</li> <li>• StAngle, EndAngle – начальный и конечный углы,</li> <li>• XRad, YRad – радиусы по осям X, Y.</li> </ul>
PieSlice(X,Y,StAngle,EndAngle,Rad)	Закрашенный сектор окружности. (X,Y) – центр. Rad – радиусом, StAngle начальный угол радиуса, EndAngle – конечный угол радиуса.
Sector(X,Y,StAngle,EndAngle,XRad,YRad)	Закрашенный сектор эллипса. с центром в (X,Y). StAngle, EndAngle – начальный и конечный углы радиуса. XRad,YRad – радиусы по осям X, Y.

#### 4.11.4.1. Многоугольники

Определены подпрограммы для прямоугольников:

- Rectangle(ARect) – закрашенный прямоугольник. Границы области задаются объектом ARect, который должен быть предварительно создан.
- Rectangle(x1, y1, x2, y2) – закрашенный прямоугольник. Границы области задаются координатами левого верхнего (x1, y1) и правого нижнего (x2, y2) углов.
- FrameRecT(ARect) – незакрашенный прямоугольник. Границы области задаются объектом ARect, который должен быть предварительно создан.
- RoundRec(x1, y1, x2, y2, x3, y3) – закрашенный прямоугольник со скругленными углами

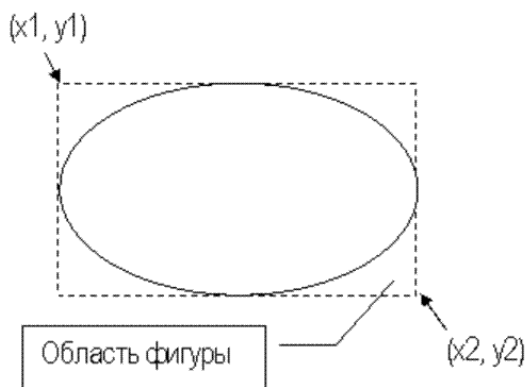


- `RoundRect(ARect, gx, gy)` – закрашенный прямоугольник со скругленными углами. Границы области задаются объектом `ARect`, который должен быть предварительно создан. Радиусы закруглений по осям  $x$ ,  $y$  –  $gx$ ,  $gy$ .
- `Polygon(Pol : array [1..N] of TPoint)` – закрашенный многоугольник - Здесь `Pol` - массив точек типа `TPoint`, каждая из которых имеет координаты  $(x, y)$ .

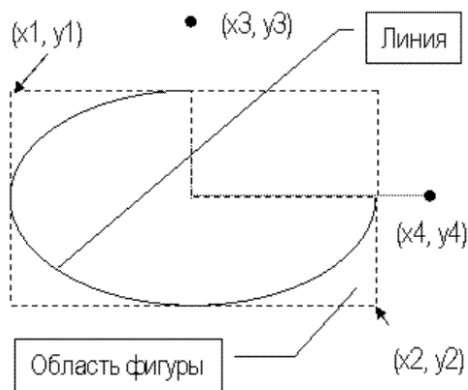
#### 4.11.4.2. Криволинейные фигуры

Определены:

- `Ellipse(ARect)` – закрашенная окружность или эллипс. Если область фигуры квадрат, то рисуется окружность. Границы области задаются объектом `ARect`, который должен быть предварительно создан.
- `Ellipse(x1, y1, x2, y2)` - закрашенная окружность или эллипс. Если область фигуры квадрат, то рисуется окружность. Границы области задаются координатами углов: левого верхнего  $(x1, y1)$  и правого нижнего  $(x2, y2)$ .

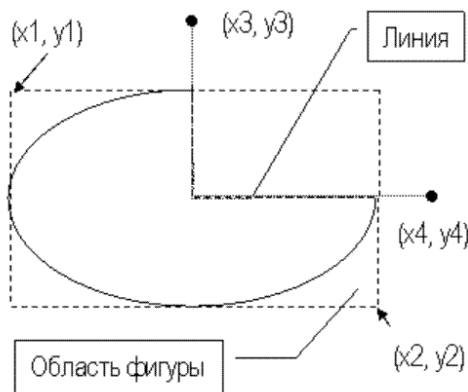


- Arc ( $x_1, y_1, x_2, y_2, \text{StAngle}, \text{EndAngle}$ ) - дуга эллипса. Границы области задаются координатами левого верхнего ( $x_1, y_1$ ) и правого нижнего ( $x_2, y_2$ ) углов. StAngle – начальный угол, EndAngle – конечный угол (единица измерения - 0.001 градуса).
- Arc ( $x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4$ ) - дуга эллипса. Граничные точки дуги - точки пересечения эллипса с прямыми линиями из центра эллипса к точкам ( $x_3, y_3$ ) и ( $x_4, y_4$ ).



- Chord ( $x_1, y_1, x_2, y_2, \text{StAngle}, \text{EndAngle}$ ) - дуга+хорда эллипса. Процедурой Arc рисуется дуга, концы дуги соединяются хордой. Полученная замкнутая фигура закрашивается.
- EllipseC ( $x_1, y_1, r_x, r_y$ ) – закрашенный центрированный эллипс. Здесь ( $x_1, y_1$ ) - центр эллипса, ( $r_x, r_y$ ) - радиусы.

- **RadialPie** ( $x1, y1, x2, y2, StAngle, EndAngel$ ) – закрашенный сектор эллипса. Границы области задаются координатами левого верхнего ( $x1, y1$ ) и правого нижнего ( $x2, y2$ ) углов.  $StAngle$  – начальный угол,  $EndAngel$  – конечный угол (единица измерения - 0.001 градуса).
- **Pie** ( $x1, y1, x2, y2, x3, y3, x4, y4$ ) – закрашенный сектор (эллипса). Отличается от **RadialPie** способом задания начальной и конечной точек дуги.



#### 4.11.5. Заполнение замкнутых областей

Для заполнения замкнутых областей предназначена процедура:

**FloodFill**( $x, y, FillColor, FillStyle$ );

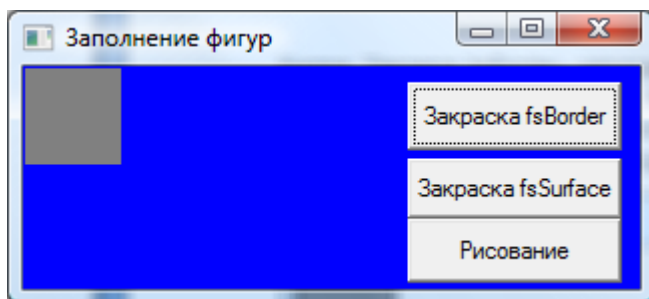
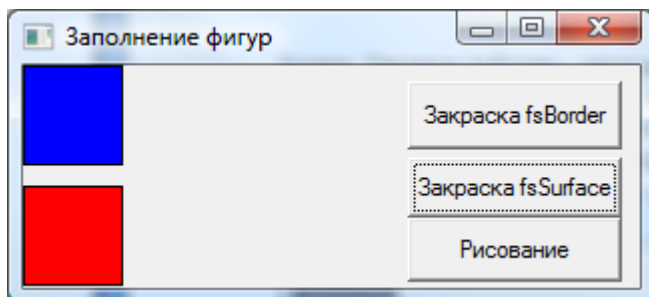
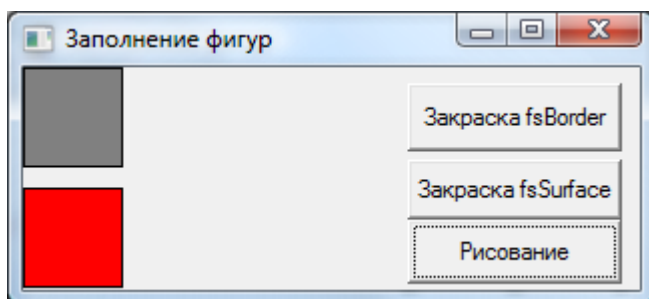
- ( $x, y$ ) – точка, относительно которой происходит заполнение цветом и стилем кисти.
- **FillColor** – цвет, определяющий границы заполнения.
- **FillStyle** – стиль заполнения (**fsSurface**, **fsBorder**). При **FillStyle=fsSurface** заполняется область холста с цветом **FillColor**. При **FillStyle=fsBorder** заполняется область холста с цветами кроме **FillColor**.),

В примере в форме кнопкой «Рисование» рисуются два прямоугольника, закрашенные серым и красным цветами.

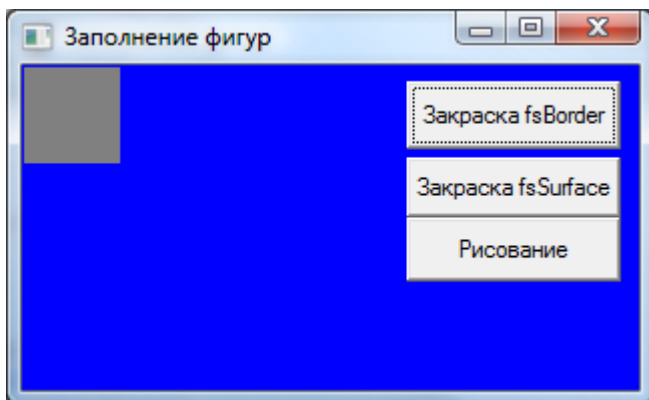
Затем задается и **FillColor=clGray**. два раза применяется процедура закраски синим цветом.

Кнопка «Закраска sSurface» запускает свой обработчик. Он устанавливает синий цвет кисти и запускает процедуру **FloodFill** с серым цветом и **FillStyle=fsSurface** относительно точки внутри первого прямоугольника. Цвет первого прямоугольника меняется на синий.

Кнопка «Закраска bsBorder» запускает свой обработчик. Он устанавливает синий цвет кисти и запускает процедуру FloodFill с серым цветом и Fill-Style=fsBorder относительно точки внутри первого прямоугольника. Цвет всех областей, кроме первого прямоугольника, меняется на синий.







## 4.12. Рисование графиков функций

Рисование графиков может осуществляться:

- С использованием процедуры `LineTo(x,y)` рисования отрезков линий.
- С использованием рисования точек.
- С использованием компонента `TChart`.

При рисовании линиями порядок действий:

- Для точек графика в цикле по координате  $x$  вычисляются значения функции  $y(x)$ .
- Процедурой `MoveTo(x,y)` указатель размещается в начальной точке.
- В цикле процедурами `LineTo(x,y)` рисуются отрезки линий, набор которых аппроксимирует график функции.

При рисовании точек порядок действий:

- Для точек графика в цикле по координате  $x$  вычисляются значения функции  $y(x)$ .
- В цикле точки графика рисуются с использованием выражения `Pixels[x,y]:=<цвет>`.

В обоих случаях при необходимости выполняются сопутствующие операции:

- Анализируется диапазон изменения функции для автомасштабирования.
- Отображаются координатные оси, сетка и поясняющие надписи.
- Осуществляется перерисовка при изменении размеров окна формы.

При крупном шаге изменения аргумента рисование линиями обычно дает график более наглядный.

Рассмотрим проект приложения под Windows с именем graf\_function\_1\_, обеспечивающий рисование точками и линиями графика функции синуса.

```
unit graf_function_1_;
{$mode objfpc}{$H+}
interface
uses
  Classes, SysUtils, LResources, Forms, Controls, Graphics, Dialogs, StdCtrls;
type
  TForm1 = class(TForm)
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
  end;
var
  Form1: TForm1;
implementation

procedure TForm1.Button1Click(Sender: TObject);
var i,amp,x01,y01,x02,y02,imax,dt1,dt2,func:integer;
begin
  imax:=100;           // число точек в периоде
  dt1=2;               // цена деления X
  dt2=2;               // шаг во времени
  amp:=70;             // амплитуда
  x01:=20;             // начала координат
  x02:=20;
  y01:=20+amp;
  y02:=y01+2*amp+40;
  // Рисуем график с Pixels
  Canvas.TextOut(0,0,'График функции с Pixels');
  Canvas.MoveTo(x01,y01); //Рисуем ось X
  Canvas.LineTo(x01+imax*dt1,y01);
  Canvas.MoveTo(x01,y01+amp); //Рисуем ось Y
  Canvas.LineTo(x01,y01-amp);
  for i:=0 to imax do    //Рисуем график
  begin
    func:=-round(amp*sin(2*pi/imax*i*dt2));
    Canvas.Pixels[x01+i*dt1,y01+func]:=clBlack;
  end;
```

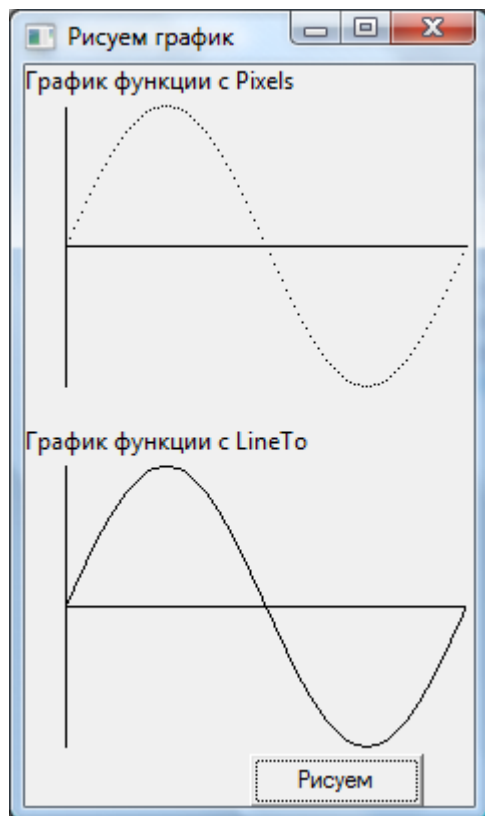
```

// Рисуем график с LineTo
Canvas.TextOut(0,y02-amp-20,'График функции с LineTo');
Canvas.MoveTo(x02,y02);           //Рисуем ось X
Canvas.LineTo(x02+imax*dt1,y02);
Canvas.MoveTo(x0,y0+amp);         //Рисуем ось Y
Canvas.LineTo(x02,y02-amp);
for i:=0 to imax do               //Рисуем график
begin
    func:=-round(amp*sin(2*pi/imax*i*dt2));
    Canvas.LineTo[x02+i*dt1,y02+func]:=clBlack;
end;

end;
initialization
    {$I graf_function_1.lrs}
end.

```

Графики зависят от цены деления по оси X  $dt1$  и шага во времени  $dt2$ .



При большом шаге в первом графике просматривается точечная структура графика. При большом шаге лучше использовать рисование графика функции линиями.

#### 4.13. Компоненты

ИСП содержит огромное количество компонент разного назначения, перечень которых дан выше в обзоре ИСП. Ниже приведены более подробные сведения по наиболее часто используемым компонентам.

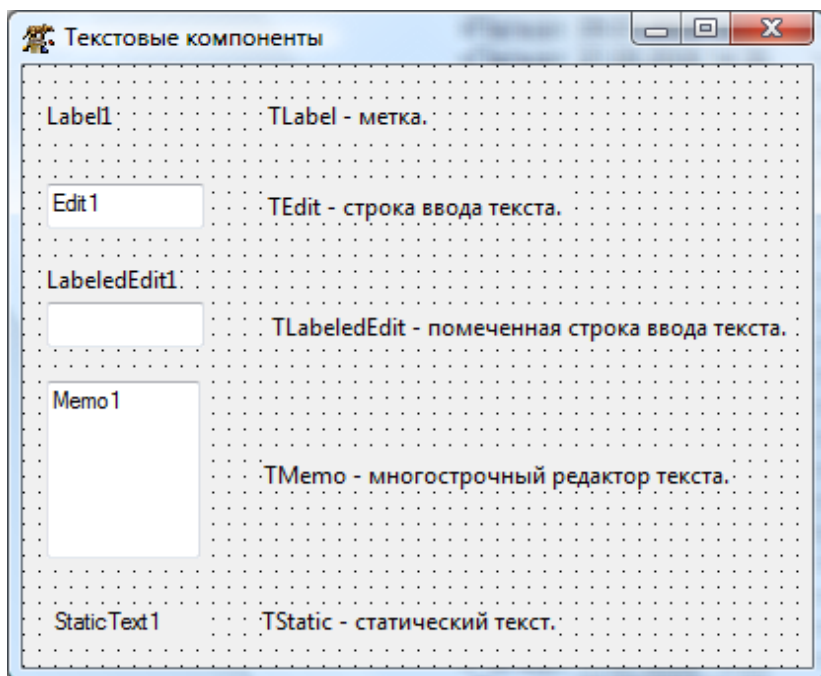
Для внесения компонента в форму нужно в Палитре Компонент выбрать страницу, где он находится, выбрать мышью компонент и указать в форме место размещения. Отрисованный компонент будет выделен. В инспекторе объектов отобразятся его свойства.

В основе всего многообразия классов и компонентов, используемых в ИСР, лежат всего лишь шесть базовых классов:

- TObject. Он является родоначальником всей иерархии использующихся в ИСР классов. Он реализует функции, которые обязательно будет выполнять любой объект, который может быть создан в ИСР. В первую очередь - это создание экземпляра объекта и его уничтожение. Процесс создания объекта включает выделение области адресного пространства, установку указателя на экземпляр объекта, задание начальных значений свойств и выполнение установочных действий, связанных с назначением объекта.
- TPersistent. Он обеспечивает своих потомков возможностью взаимодействовать с другими объектами и процессами на уровне данных. Его методы позволяют передавать данные в потоки, а также обеспечивают взаимодействие объекта с Инспектором объектов.
- TComponent. Это предок всех компонентов. Он используется в качестве основы для создания невизуальных компонентов и реализует основные механизмы, которые обеспечивают функционирование любого компонента.
- TLCLComponent. Это предок всех компонентов библиотеки LCL.
- TControl. Его основное назначение — обеспечить функционирование визуальных компонентов, которые умеет работать с GUI (Graphic User Interface — графический интерфейс пользователя ОС) и отображать себя на экране.
- TWinControl. Он обеспечивает создание оконных (получающих фокус) элементов управления.
- TGraphicControl. Он используется для создания неоконных (не получающих фокус) элементов управления.

#### **4.13.1. Текстовые компоненты**

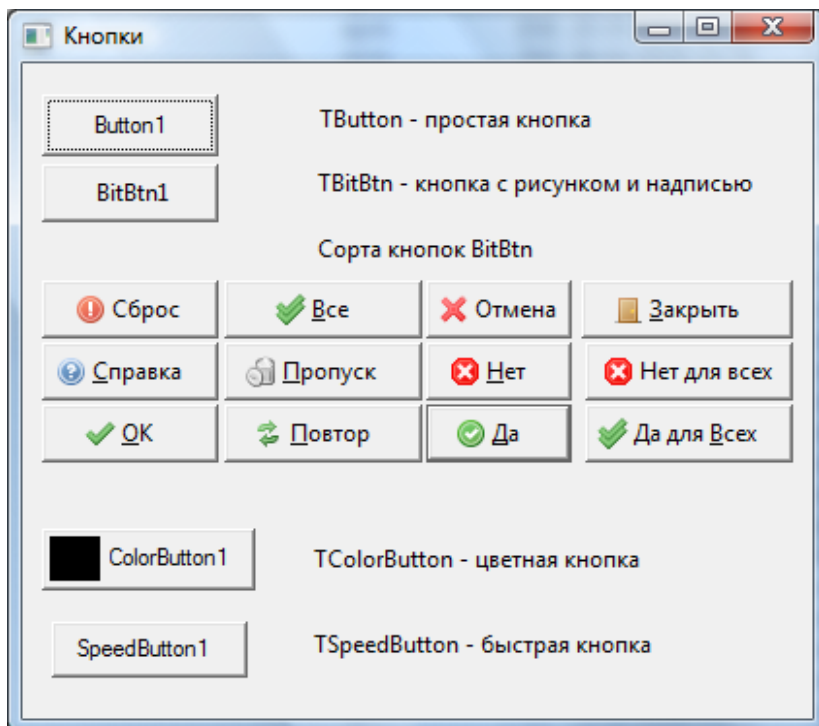
В ИСР лпределены несколько текстовых компонент:



Компонент	Страница	Описание
TLabel	Standard	Метка. Текстовая информация разного назначения: названия, заголовки, пояснения и др.
TEdit	Standard	Строка ввода. Ввод и редактирования одной строки текста.
TLabeledEdit	Additional	Помеченная строка ввода. Комбинация метки и строки ввода.
TMemo	Standard	Многострочный редактор. Ввод и редактирование многострочного текста.
TStaticText	Additional	Статический текст. Текст в рельефной рамке.
TSpinEdit	Misc	Редактор целых чисел со счетчиком. Включает реверсивный счетчик UpDown.
TFloatSpinEdit	Misk	Редактор вещественных чисел со счетчиком. Включает реверсивный счетчик UpDown.

### 4.13.2. Кнопки

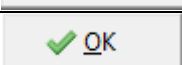

Назначение кнопки – формирование события при нажатии на нее. В ИСР определены классы кнопок:



Компонент	Страница	Описание
TButton	Standard	Метка. Для вывода текстов разного назначения: названия, заголовки, пояснения и др.
TBitBtn	Additional	Кнопка с рисункм и надписью, которые иллюстрируют ее функциональное назначение.
TSpeedButton	Additional	Быстрая кнопка. Не может получать фокус ввода и потому быстрая.
TColorButton	Misc	Цветная кнопка.

Для кнопок программируется обработчик события (по умолчанию `onClick` – нажатие). При возникновении события свойство кнопки `ModalResult` принимает значение `True` и обработчик срабатывает.

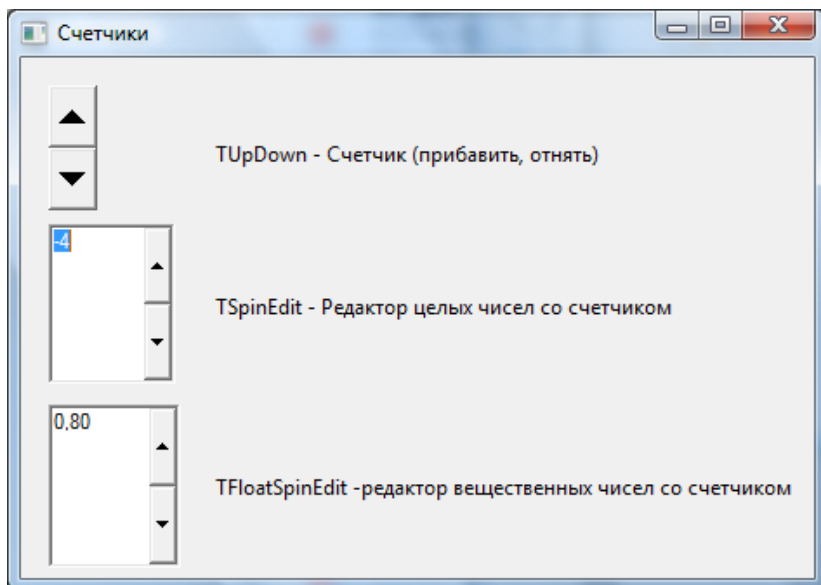
Для кнопки BitBtn определены стандартные сорта разного назначения, для которых есть свои обработчики. Чтобы несколько кнопок BitBtn работали независимо, каждому сорту назначается свое значение ModalResult. Определены сорта кнопок BitBtn:

Сорт	Вид	Описание
bkCustom		Заготовка для пользователя. ModalResult = mrNone.
bkAbort		Завершить действие. ModalResult = mrAbort.
bkAll		Закончить. ModalResult = mrAll.
bkCancel		Отменить. ModalResult = mrCancel.
bkClose		Закреть форму. ModalResult = mrNone. Используется системный обработчик.
bkHelp		Вывод справки. ModalResult = mrNone. Используется системный обработчик.
bkIgnore		Игнорировать. ModalResult = mrIgnore.
bkNo		Отказ. ModalResult = mrNo.
bkNoToAll		Отказ. ModalResult = mrNoToAll.
bkOK		Согласие. ModalResult = mrOk.
bkRetry		Повторить. ModalResult = mrRetry.
bkYes		Да. ModalResult = mrYes.
BkYesToAll		Да. ModalResult = mrYesToAll.

### 4.13.3. Счетчики

Счетчики предназначены для изменения данных на заданную величину. В ИСР определены классы счетчиков:



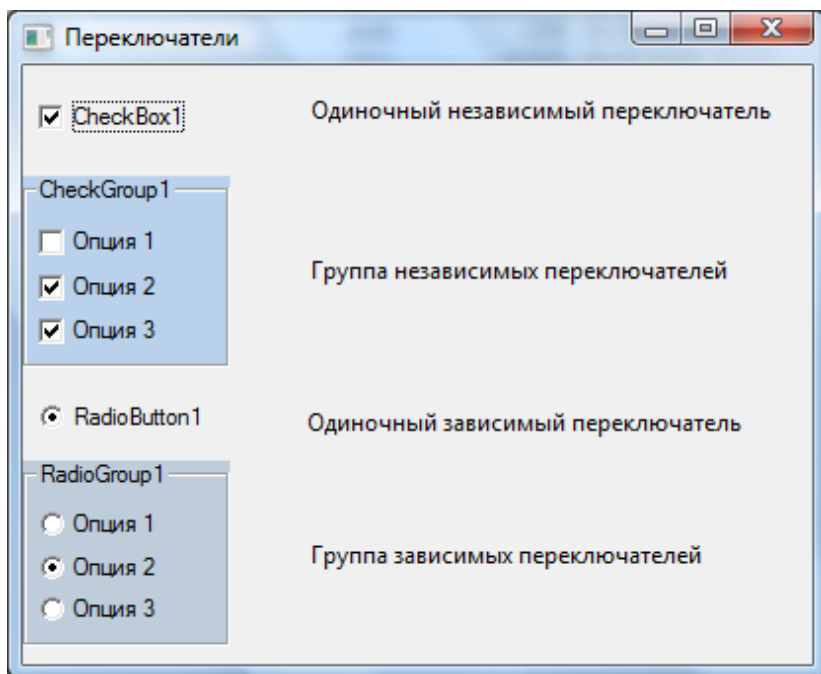


Компонент	Страница	Описание
TUpDown	Common Controls	Счетчик. Прибавляет или отнимает заданное значение.
TSpinEdit	Misc	Редактор целых чисел со счетчиком.
TFloatSpinEdit	Misc	Редактор вещественных чисел со счетчиком..

#### 4.13.4. Переключатели

Переключатели (флажки) предназначены для выбора значений из определенного множества. Состояние переключателя определяется значением его свойства Checked. Переключатель включен, при Checked = True и выключен при Checked = False. В ИСП определены классы переключателей:

- TCheckBox - один независимый переключатель. Символ – квадрат, признак включения - галочка внутри.
- TCheckGroup - группа независимых переключателей. Включить можно любое количество переключателей из группы.
- TRadioButton - один зависимый переключатель. переключатель. Символ – круг, признак включения - закрашенный овал внутри.
- TRadioGroup - группа зависимых переключателей (или радиогруппа). Включить можно только один переключатель из группы. При включении одного другие автоматически выключаются.



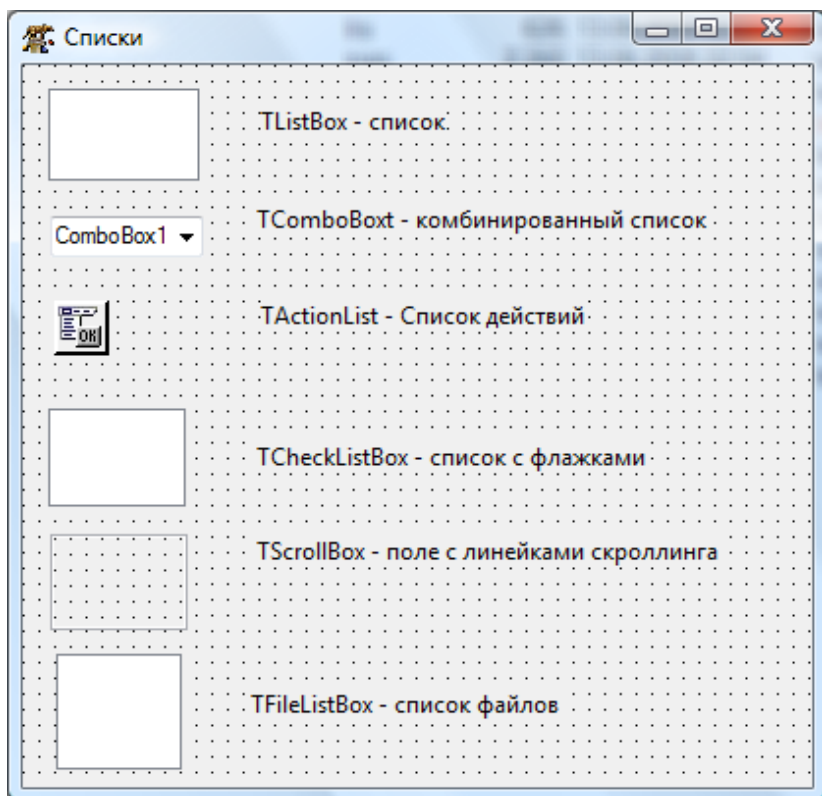
#### 4.13.5. Группы

Группы предназначены для визуального выделения связанных друг с другом компонент. В ИСР определены классы групп:

- TGroupBox – группа. Представляет собой прямоугольную рамку с заголовком.
- TPanel – панель. Представляет собой прямоугольное поле с двойной фаской.

#### 4.13.6. Списки

Списки предназначены для накопления однотипных данных. В ИСР определены классы списков:



Компонент	Страница	Описание
TListBox	Standard	Простой список.
TComboBox	Standard	Комбинированный список. Это комбинация простого списка и строки ввода Edit.
TActionList	Standard	Список действий, которые должны выполняться при определенных событиях. Компонент скрытый.
TCheckListBox	Additional	Список с флажками.
TScrollBox	Additional	Поле с линейками скроллинга.
TFileListBox	Misc	Список файлов из выбранной папки

### 4.13.7. Управляющие компоненты

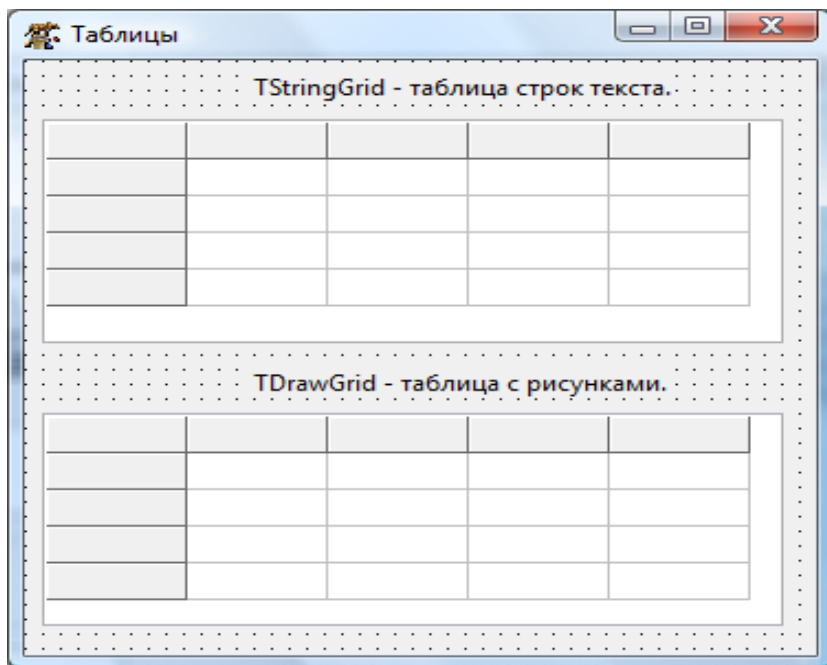
Управляющие компоненты предназначены для создания пользовательского интерфейса. В ИСП определены классы управляющих компонент:

- TTrackBar – ползунок. Позволяет мышью выбирать значение в пределах от минимального до максимального.
- TProgressBar – линейка прогресса. Поле, которое закрашивается при выполнении процесса.
- TStatusBar – статусная строка. Поле для текста внизу формы.
- TToolBar – поле вверху формы для панели инструментов
- TButtonPanel – панель с набором кнопок.

### 4.13.8. Таблицы

Табличные компоненты предназначены для создания таблиц в стиле Excel. В ИСП определены классы таблиц:

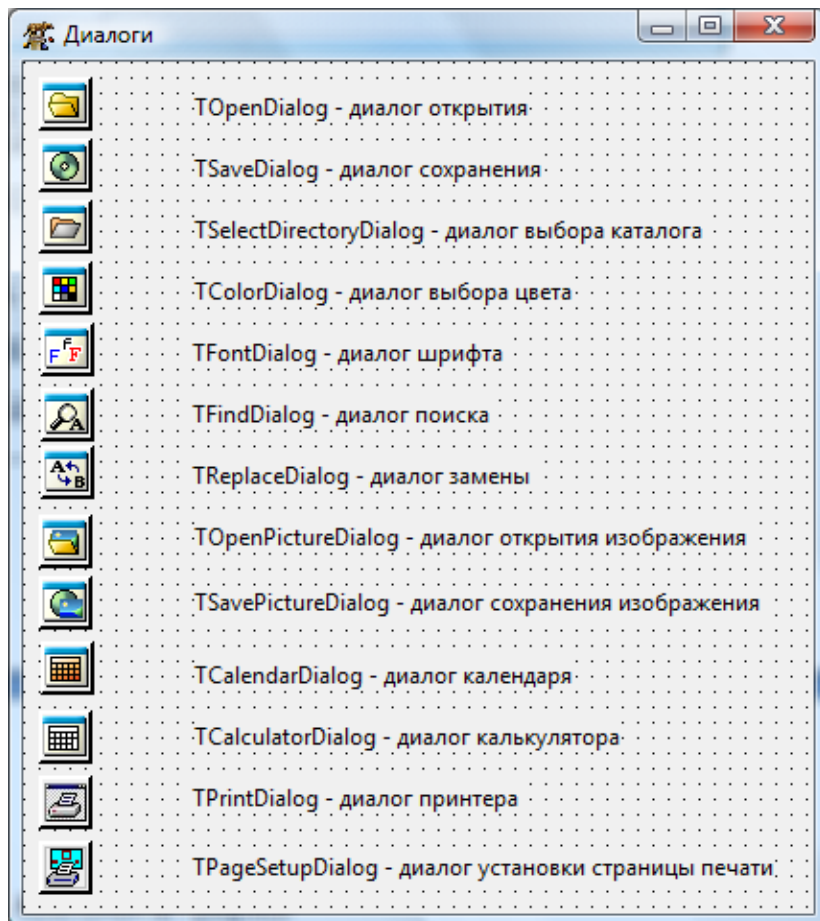
- TStringGrid – таблица строк текста.
- TDrawGrid – таблица с рисунками.



Обе таблицы организованы в виде двумерного массива ячеек. Верхняя строка резервируется для заголовков столбцов, а левый столбец – для заголовков строк.

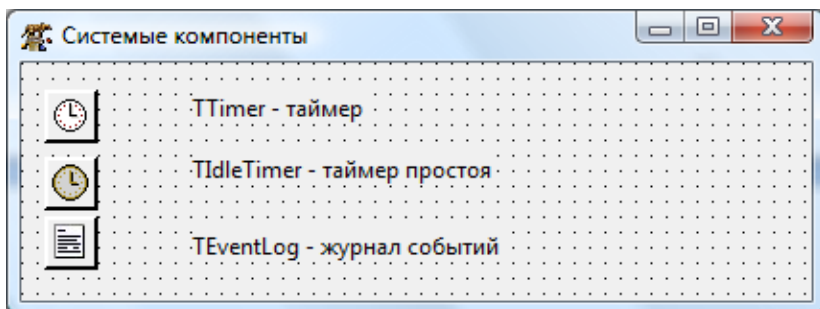
#### 4.13.9. Диалоги

В ИСР определены стандартные классы диалогов, не отличающиеся от системных:



#### 4.13.10. Системные компоненты

В ИСР определены следующие системные компоненты:



### 4.13.11. Компоненты графики

Для поддержки графики в ИСР определено много компонент. Компоненты для отображения картинок:

- Image,
- Shape,
- PaintBox.

#### 4.13.11.1. Компонент TImage

Компонент TImage используется для размещения на форме картинки, загружаемой из файла. Находится в закладке Additional палитры компонент.

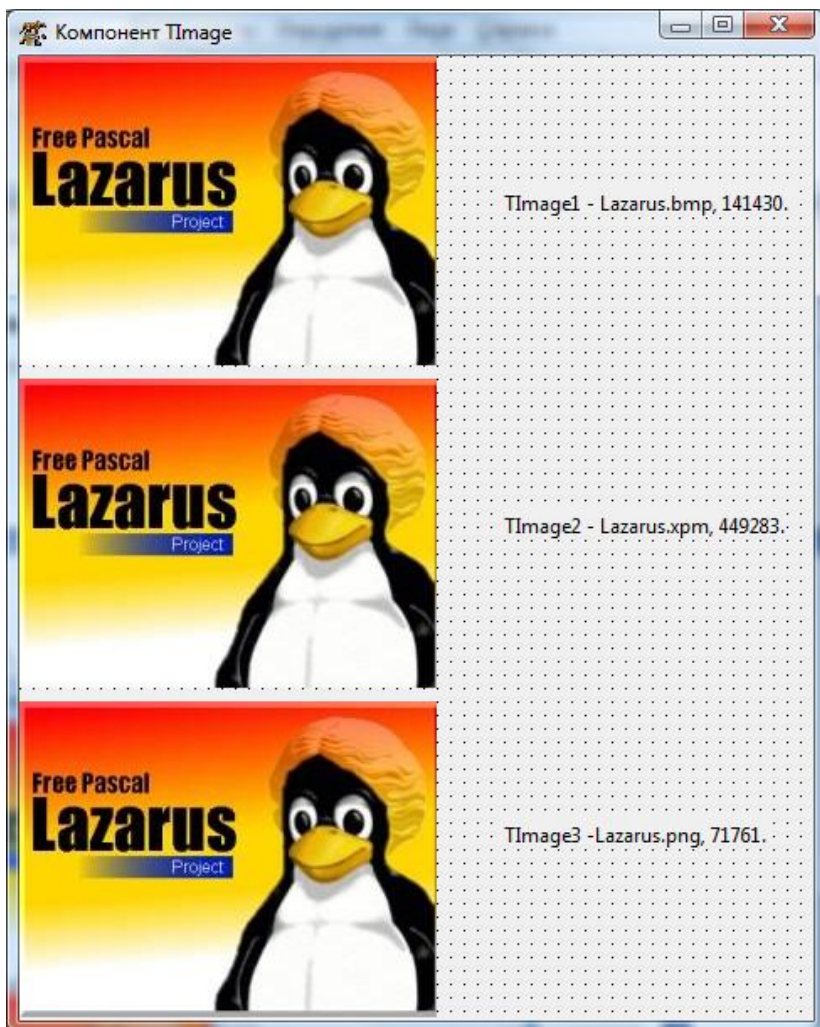
Возможные типы изображений:

- битовая карта BitMap (\*.bmp),
- пиксельная карта PixMap (\*.pmp),
- переносимая битовая карта Portable BitMap (\*.pbm),
- серая битовая карта Gray BitMap (\*.pgm),
- пиктограмма (\*.ico),
- переносимая сетевая графика PNG - Portable Network Graphic (\*.png),
- расширенная карта XPM(\*.xpm).

В примере форма содержит 3 компонента TImage, в которых загружена одна и та же картинка Lazarus в разных форматах с разными размерами файла:

- TImage1 – Lazarus.bmp, размер файла 141430 байт.
- TImage2 – Lazarus.xpm, размер файла 449283 байт.
- TImage3 – Lazarus.png, размер файла 71761 байт.

Для загрузки картинки можно выбрать компонент TImage и в инспекторе объектов в свойстве Picture вызвать диалог загрузки изображения. В нем нужно выбрать картинку, которая будет загружена в компонент.

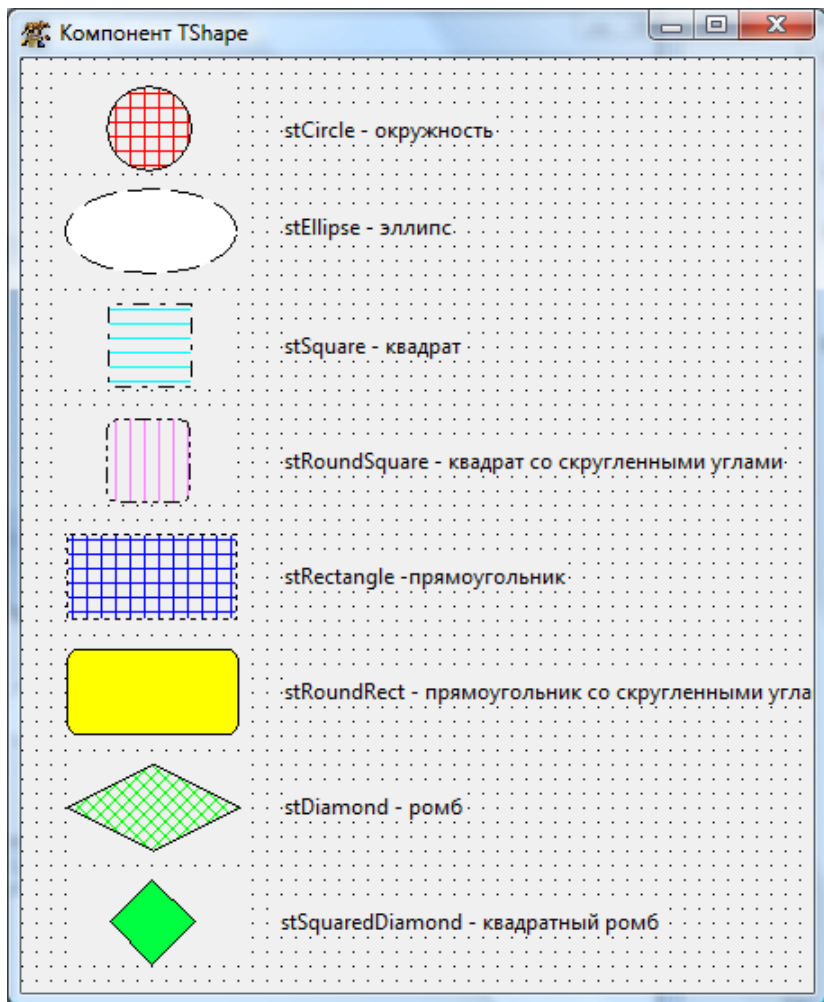


#### 4.13.11.2. Компонент TShape

Компонент Shape облегчает рисование фигур. Экземплярами класса TShape являются фигуры: круги, эллипсы, прямоугольники и др. Находится в закладке Additional палитры компонент.

Позволяет создавать на форме простейшие графические примитивы, выбираемые из списка. Для каждой фигуры можно устанавливать:

- Размеры.
- Цвет линии. Выбирается в диалоговом окне выбора цвета.
- Стиль линии. Возможны стили - сплошная, пунктирная, штрихпунктирная, точки, двойные точки.
- Стиль заполнения. Возможны стили – сплошной, горизонтальная штриховка, вертикальная штриховка, диагональная штриховка.
- Цвет заполнения. Выбирается в диалоговом окне выбора цвета.





### 4.13.11.3. Компонент PaintBox

Это область для рисования, выделяемая на форме. При ее использовании координаты, передаваемые от указателя мыши, отсчитываются не для формы, а для компонента PaintBox. Для компонента нужен обработчик события.

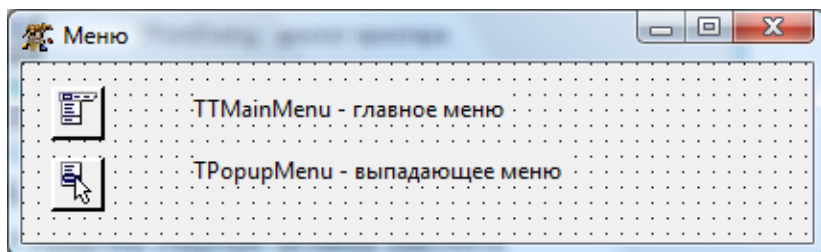
### 4.13.12. Компоненты создания меню

Меню предназначено для выбора действия из списка. Пункты меню содержат перечень допустимых операций. С каждым пунктом меню может быть связана некоторая процедура, которая выполняется в случаях, когда пункт активизирован. Метод запуска:

- щелчок мышью по пункту,
- нажатие клавиши <Enter>.

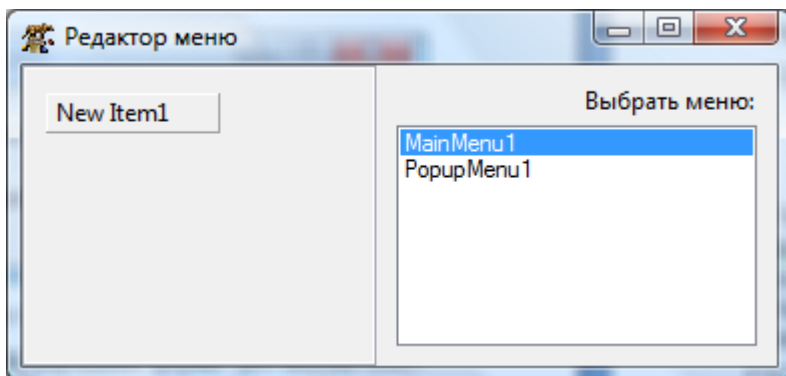
Меню имеет иерархическую структуру. На верхнем уровне иерархии меню нулевого уровня. Далее меню первого уровня и т.д. Меню нижних уровней отображается, как выпадающее, при активизации пункта меню текущего уровня. В ИСР для создания меню используется компоненты

- Главное меню MainMenu. Это меню окна приложения.
- Контекстное меню PopupMenu. Оно ассоциируется с оконным элементом управления и может быть вызвано щелчком правой кнопки мыши по нему.



Находятся в закладке Standard палитры компонент.

Главное и выпадающее меню создаются на этапе конструирования формы с помощью редактора меню. Для вызова редактора меню нужно дважды щелкнуть левой кнопкой мыши по компоненту MainMenu или PopupMenu. Будет вызван редактор меню, в котором выделен вид меню. Затем следует нажать кнопку NewItem1. Это передает фокус в процедуру проектирования меню, которая автоматически создается при первом обращении.



## 5. Отладка программ

Успешное завершение процесса компиляции не означает, что в программе нет ошибок. Убедиться, что программа работает правильно, можно только в процессе проверки ее работоспособности, который называется тестирование. Обычно программа редко сразу начинает работать так, как надо, или работает правильно только на некотором ограниченном наборе исходных данных. Это свидетельствует о том, что в программе есть алгоритмические ошибки. Процесс поиска и устранения ошибок называется отладкой/

В обеих ИСР предусмотрены средства отладки.

- Компилятор с регулируемыми опциями.
- Отладчик для поиска и устранения ошибок в программе. Отладчик позволяет выполнять трассировку программы, наблюдать значения переменных, контролировать выводимые переменные.