Введение в IDE Lazarus

Эта статья написана на момент актуальности Lazarus 0.9.24. С учётом того, что проект Lazarus стремительно развивается, ряд функций, описанных ниже, может существенно отличаться либо быть упразднённым в последующих версиях продукта. Впрочем, у автора есть все основания полагать, что описанный здесь базовый функционал будет поддерживаться всеми последующими версиями продукта Lazarus.

I. Общие сведения о проекте Lazarus

1.1. Цель проекта

Lazarus – это IDE (интегрированная среда разработки) для создания (графических и консольных) приложений при помощи компилятора FreePascal. FreePascal - это компилятор языков Pascal и Object Pascal, распространяемый под лицензией (L)GPL, и работающий под Windows, Linux, Mac OS X, FreeBSD, и не только.

Рисунок 1. Логотип Lazarus

Lazarus - это недостающий элемент, который позволит вам разрабатывать программы для всех вышеперечисленных платформ в Delphi-подобном окружении. Эта IDE является инструментом RAD, включающим в себя дизайнер форм. Подбор визуальных компонентов максимально приближен к VCL, и носит название FCL – Free Components Library.[1]

1.2. Установка Lazarus

Дистрибутивы Lazarus доступны для свободного скачивания вместе с исходным кодом со страницы http://sourceforge.net/project/showfiles.php?group_id=89339. Для корректной работы необходимо скачать вначале сам язык Free Pascal для нужной платформы, после чего скачать дистрибутив Lazarus. Дистрибутив Free Pascal имеет размер около 30 Мб, установочный пакет Lazarus - приблизительно 50 Мб.

В случае с ОС Windows установка полностью автоматизирована, и не представляет трудностей для пользователя.

1.3. Работа с Lazarus

Lazarus является средой визуального программирования, ориентированной на быстрое создание оконных приложений. Основным плюсом этой среды является принцип “Write once, compile everywhere”, позволяющий компилировать код программ на различных платформах без внесения каких-либо изменений. Это достигнуто за счёт высокого уровня абстракции от ОС: используются собственные функции работы с пользовательским интерфейсом, по-своему реализованные в различных ОС, но полностью совместимые с точки зрения функциональности. Поэтому исходный код программы, написанной в Lazarus под ОС Linux, без внесения каких-либо изменений будет скомпилирован в версии Lazarus для Windows.[2]

Языком программирования в среде Lazarus является Free Pascal – мощный диалект языка Паскаль, распространяемый свободно. Этот диалект мощнее, чем язык, используемый в Delphi 7. К примеру, в языке Free Pascal поддерживается перегрузка операторов.

---

[1] © Official Lazarus Product Information. Меню[Help]/[About] для более полной информации.

[2] Разумеется, исключение составляют случаи, в которых приложение использует особенности той или иной ОС, отличающие её от остальных. К примеру, использование новых визуальных стилей – особенность приложений Windows XP.

Внешний вид среды Lazarus изображён на рисунке 2.

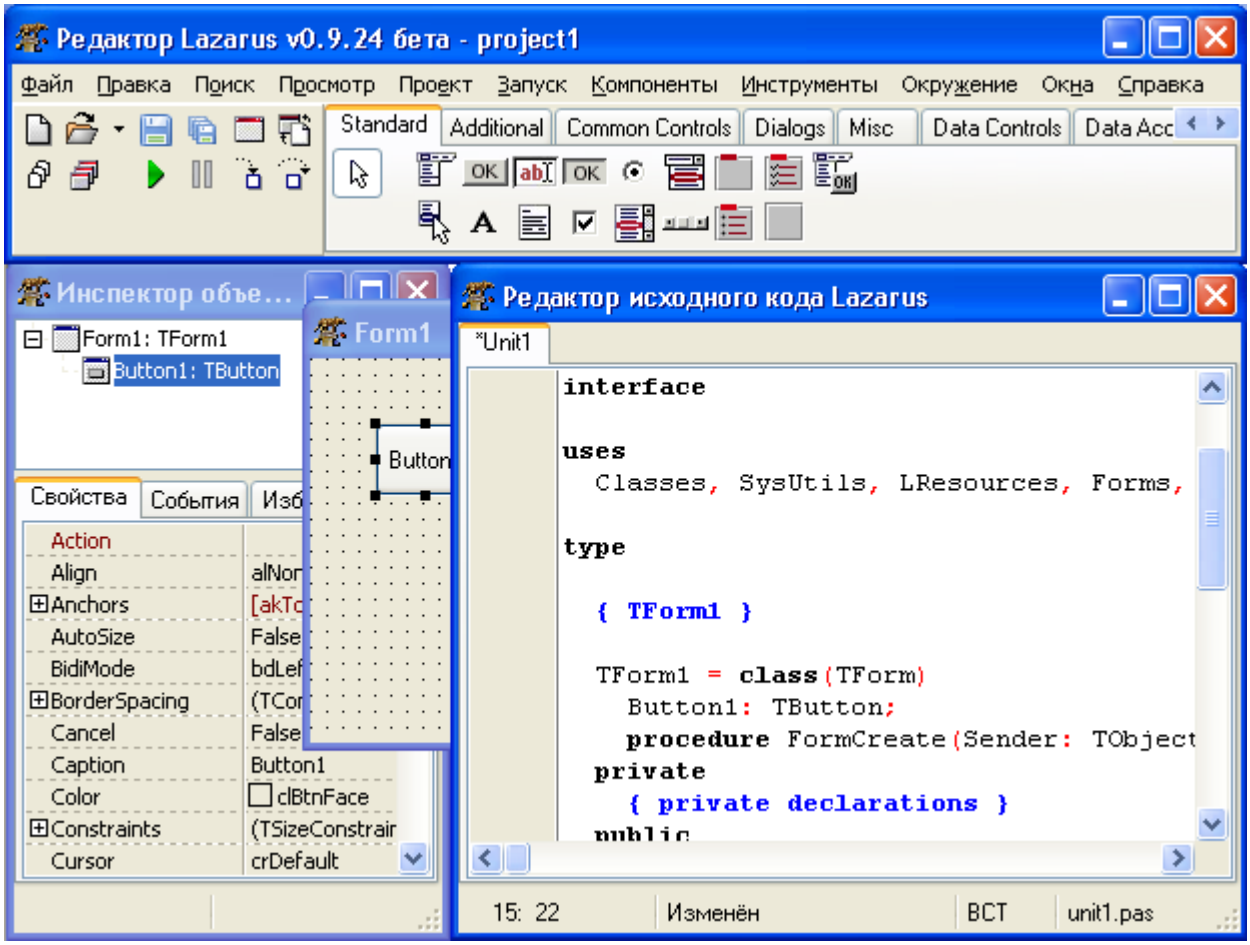

Рисунок 2. Среда Lazarus

Как видно из рисунка 2, внешний вид среды максимально приближен к Borland Delphi 7. Однако среда Lazarus имеет ряд особенностей, отсутствующих в Delphi. К ним можно отнести следующие особенности:

- «Избранное» в «Инспекторе объектов» - вкладка, содержащая наиболее часто используемые свойства элементов управления
- Улучшенный редактор исходного кода, позволяющий, например, сворачивать код процедур, оставляя лишь их заголовки, что существенно упрощает просмотр исходного кода в громоздких модулях
- Набор специальных визуальных компонентов, весьма полезных при решении многих типичных задач. Об этих компонентах будет написано позднее.

2. Первая программа в Lazarus

2.1. Создание проекта

Чтобы создать проект в среде Lazarus, необходимо выбрать пункт «Создать проект» меню «Проект». Выбрав пункт «Приложение» (графическая программа с поддержкой LCL), Вы создадите типичное оконное приложение. Общий вид среды после создания пустого проекта показан на рисунке 3.

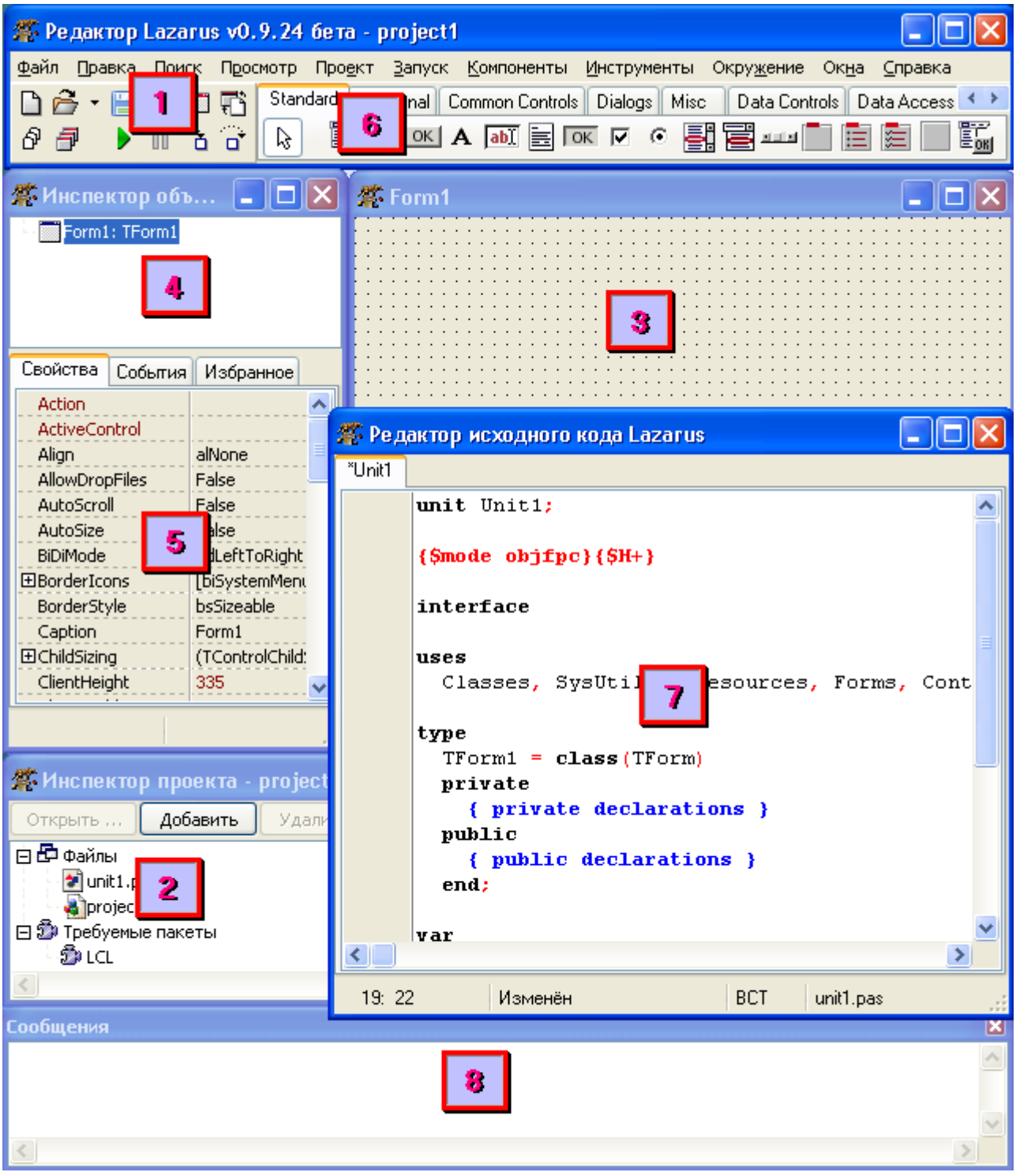

Рисунок 3. Основные окна среды Lazarus

Рассмотрим назначение каждой из отмеченных областей:

1. **Основная панель инструментов и меню**

   Меню содержит все основные команды среды Lazarus, сгруппированные по их предназначению (работа с файлами, отладка, компиляция, справка и т. п.). На панель инструментов вынесены наиболее часто используемые пункты меню для упрощения доступа к ним – это команды сохранения и открытия, запуска, останова, трассировки и пр.

2. **Инспектор проекта**

   Это окно содержит инструменты для управления файловым составом проекта. С точки зрения среды Lazarus, любое создаваемое приложение является проектом. Проект – это файл, содержащий в себе информацию, необходимую для компиляции сложной программы, состоящей из нескольких модулей. Проект Lazarus имеет расширение LPR. В нашем случае проект состоит только из двух файлов – файла проекта и файла формы Form1. В серьёзных решениях число файлов в проекте может быть очень большим. Помимо прочего, инспектор проекта также позволяет показать его зависимости от внешних пакетов. В нашем случае, project1 требует наличия LCL – библиотеки оконных компонентов.

3. **Макет пользовательской формы**

   Это – дизайнер форм, позволяющий визуально создавать графический интерфейс пользователя (GUI) для Вашей программы. Дизайнер форм имеет ряд удобных функций, таких как, например, выравнивание элементов управления относительно друг друга.

4. **Дерево объектов инспектора объектов**

   Эта область инспектора объектов отображает иерархию элементов управления пользовательской формы. Она может использоваться для наглядного представления объектной структуры и для выделения объектов, расположенных на форме так, что выделить их мышью непосредственно не удаётся.

5. **Редактор свойств выбранного элемента управления**

   Эта область инспектора объектов позволяет программисту задавать или просматривать свойства выбранного элемента во время проектирования приложения (design-time). Следует отметить, что редактор свойств позволяет просматривать и задавать свойства, имеющие агрегатный тип (шрифт, рисунок, дочерний объект и т.п.). Также редактор свойств используется для назначения обработчиков событий элементам управления. Наиболее часто используемые свойства и события для элементов управления для удобства вынесены на особую вкладку – «Избранное». Высоту строк списка свойств можно менять в настройках среды Lazarus (меню «Окружение»/«Настройки окружения»/«Инспектор объектов»/«Высота элемента»).

6. **Палитра элементов управления**

   Эта область IDE содержит набор элементов управления, размещаемых на пользовательских формах, распределённых по логическим группам в соответствии с их функциональным назначением.

7. **Редактор кода**

   Это окно позволяет редактировать исходный код Вашего приложения. Редактор кода в Lazarus весьма удобен. К числу полезных особенностей можно отнести наличие возможности скрывать код процедур (и вообще содержимое операторных скобок begin..end), что существенно упрощает поиск нужного кода.

8. **Окно сообщений**

Это окно содержит информацию о результатах компиляции проекта. Здесь Вы можете получить список ошибок в коде Вашего приложения, а также список предупреждений и подсказок компилятора.

2.2. Разработка интерфейса и написание обработчиков событий

Как и в любой другой среде визуального программирования, в Lazarus элементы управления размещаются на пользовательской форме с помощью мыши. К примеру, разместим две кнопки на пользовательской форме:

1. Выберем элемент TButton на палитре элементов, после чего выполним щелчок левой кнопкой мыши в произвольной области пользовательской формы. В этом месте появится командная кнопка (Button). Если вместо щелчка мышью нажать левую кнопку мыши и, не отпуская её, вести мышь в сторону, размер элемента можно будет задать уже при его создании. В любом случае, размер элемента может быть изменён при помощи соответствующих маркеров на его границах.

2. Разместим ещё одну кнопку на пользовательской форме.

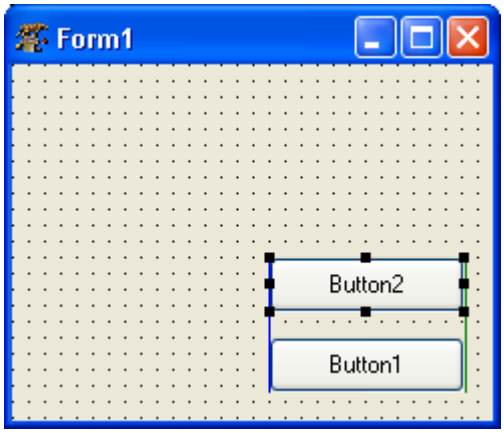

Рисунок 4. Выравнивание элементов

3. Положение уже размещённых элементов можно менять, перетаскивая их мышью. Для иллюстрации поместим созданные кнопки внизу формы, одну над другой, и выровняем их размер. Обратите внимание, что редактор форм «почувствовал» желание выровнять элементы друг относительно друга, и самостоятельно подобрал горизонтальный отступ второй кнопки соответственно отступу первой. На рисунке 4 видны линии выравнивания, помогающие разместить элементы управления в точности один над другим. Этот же механизм позволяет выровнять и размер элементов.

4. Зададим нужный заголовок пользовательской форме. Воспользуемся для этой цели инспектором объектов. Заголовок формы содержится в свойстве Caption.

5. Зададим названия кнопок, размещённых на форме, изменив свойство Caption каждой из них. Установим их имена в программе (свойство Name) соответственно btnYes и btnNo.

6. Разместим на окне элемент управления типа «метка» (TLabel), и установим необходимый текст метки (рисунок 5).

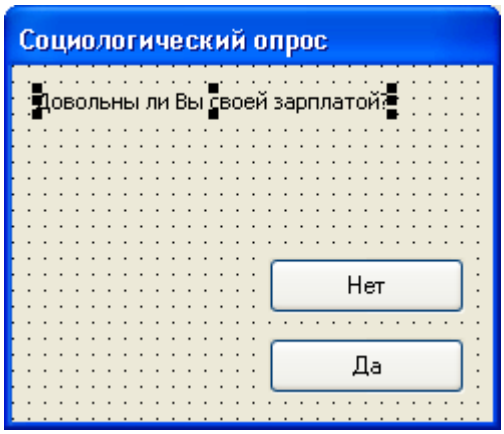

Рисунок 5. Конечный вид формы

7. Разместим на окне элемент управления типа «метка» (TLabel), и установим необходимый текст метки.

8. Удалим кнопки управления окном из пользовательской формы, установив все параметры BorderIcons для пользовательской формы в «False».

9. Добавим реализма и жизненности ситуации, назначив обработчики событий OnEnter, OnMouseEnter, OnMouseExit для кнопки btnNo. Для назначения обработчика события, необходимо выполнить двойной щелчок мышью по пустому полю справа от имени события в инспекторе объектов (вкладка «События»).

Добавим код обработчиков:

```
procedure TForm1.Button2MouseEnter(Sender: TObject);
begin
   Button2.Enabled:=false;
end;
procedure TForm1.Button2MouseLeave(Sender: TObject);
begin
   Button2.Enabled:=true;
end;
```

В нашем случае код процедуры `TForm1.Button2Enter` совпадает с кодом процедуры `TForm1.Button2MouseEnter`, поэтому код этой процедуры опущен.

Добавим последний обработчик события, обрабатывающий щелчок пользователя по кнопке «Да»:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  MessageDlg('Опрос',
             'Мы и не сомневались в вашем ответе!',
             mtInformation,
             [mbOK],
             '') ;
  close
end;
```

2.3. Выполнение программы

Для того, чтобы запустить программу, Вы можете нажать F9, либо выбрать пункт «Запуск» из меню «Запуск». После этого на экране появится созданная Вами пользовательская форма.

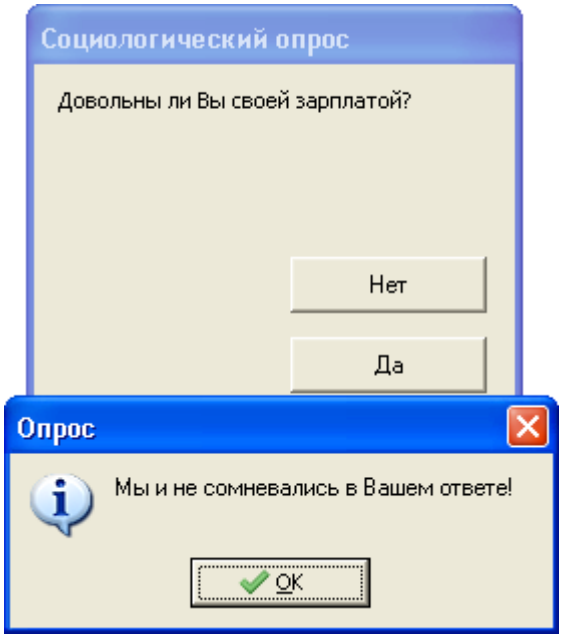

Рисунок 6. Выполняющаяся программа

Таким образом, в Lazarus создаются простые оконные приложения. Заметим, что перенос такого кода на другие платформы осуществляется вполне безболезненно, так как написанный Вами код может быть без модификаций скомпилирован, например, на Lazarus для Linux.

Приведём для справки названия и краткие описания некоторых элементов управления:

- **TCalcEdit**

  Это поле даёт возможность пользователю вводить числовые данные при помощи встроенного калькулятора, вызываемого кнопкой справа от поля ввода. Во многих случаях это может оказаться весьма удобным.

- **TDateEdit, TDirectoryEdit, TFileNameEdit**

  Эти поля также дают возможность комфортного ввода соответствующих данных посредством визуального их выбора пользователем. Интерфейс выбора также вызывается нажатием кнопки справа от поля ввода.

- **TButtonPanel**

  Этот элемент позволяет делать интерфейс Вашей программы более стандартным, удобным и красивым. Он содержит в себе типовые кнопки, полезные для множества разнообразных диалоговых окон (рисунок 7).

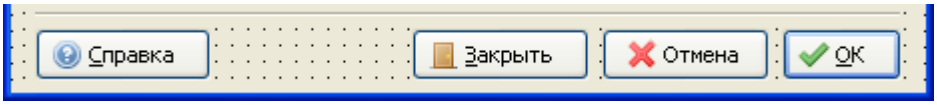

Рисунок 7. Панель TButtonPanel

3. Попробуйте написать сами

1. Создайте приложение, позволяющее получить название месяца и количество дней в нём по его номеру. Номер месяца вводится пользователем в поле TEdit (за хранимый текст отвечает свойство Text компонента TEdit), название месяца и число дней выводятся по нажатию кнопки (TButton) в два других текстовых поля. Для преобразования целого числа в строку использовать функцию IntToStr( ).

2. Создайте приложение, определяющее количество гласных во введённом пользователем слове русского языка. Для ввода и вывода использовать TEdit. Подсказка: перед подсчётом количества гласных слово проще перевести в нижний или верхний регистр, для упрощения сравнения. Проверка соответствия гласности легко реализуется, используя множества и оператор **IN**.

3. Создайте приложение, вычисляющее *геометрическое отражение* двухмерного вектора. Геометрическое отражение вектора $v_1(x_1,y_1)$ от вектора $v_2(x_2,y_2)$, именуемого «зеркалом», определяется как разность *проекции вектора* v_1 *на вектор* v_2 **и** *проекции вектора* v_1 *на вектор* v_3, получаемый из вектора v_2 поворотом на 90 градусов по часовой стрелке. Подсказка: вектор (x,y), повернутый на 90 градусов по часовой стрелке, имеет вид (y,-x). Организовать ввод данных через поля Tedit.

4. Создайте приложение, считающее среднее геометрическое введённых в TMemo чисел. Считать, что в каждой строке текстового окна введено только одно число. Случаи пустых или символьных строк не рассматривать. Для доступа к строкам поля использовать свойство Lines: Memo1.Lines[i] вернёт i-ю строку поля Memo1. Для преобразования строки в число использовать функцию StrToFloat( ).

5. Создайте приложение, позволяющее транспонировать аккорды. Исходные аккорды находятся в поле TMemo, по одному на строку. Допустимые аккорды: C,C#,D,D#,E,F,F#,G,G#,A,A#,B. Аккорды расположены в том порядке, в каком проходят транспонирование. Транспонирование заключается в циклическом сдвиге тональности: при транспонировании на 4 лада вперёд, C переходит в E, A переходит в C#, и т. д. Результирующий набор аккордов вывести в другое текстовое поле.

6. Создайте приложение, использующее компонент «TTimer» для отображения «бегущей строки». Для этого разместите компонент TTimer на форме, установите его свойство Interval равным 100, после чего в обработчике события OnTimer добавьте код для изменения свойства Left (X-координата) компонента TLabel на постоянное значение, и проверку «вылета» текста за границы формы путём сравнения координаты Left компонента TLabel с шириной формы (Form1.Width). Если компонент покинул пределы формы, переместить его в противоположный край (Label1.Left := -Label1.Width).
7. Создайте приложение, осуществляющее сортировку строк элемента TMemo любым известным Вам способом. Результат сортировки вывести в другой элемент TMemo. Подсказка: сравнение строк возможно при помощи операторов “>” и “<”.

Алексей Розанов, март 2008 г.