§ 5. Создание приложений

Пример 5.1. Рекомендации по созданию оконных приложений.

1. В приложении рекомендуется разместить главное меню и инструментальную панель быстрых кнопок, дублирующих основные разделы меню.
2. Желательно, чтобы объекты приложения обладали контекстными меню, появляющимися при нажатии правой клавишей мыши на объекте.
3. Для объектов рекомендуется прописать подсказки, всплывающие при наведении указателя мыши на объект.
4. Рекомендуется реализовать строку состояния, используемую для выдачи различной информации.
5. При нажатии клавиши F1 должен загружаться файл справки.
6. В программе желательно реализовать возможность настройки и сохранения настроек, чтобы при следующем сеансе работы их не пришлось устанавливать заново.
7. Если результат работы приложения зависит от каких-либо параметров, обязательно укажите значения по умолчанию. Они позволят ускорить взаимодействие пользователя с программой, а также являются примером того, в каком формате данные следует вводить.

Мощным воздействием на психику человека является цвет, поэтому с ним нужно обращаться очень осторожно. Нужно стремиться использовать ограниченный набор цветов и уделять внимание их правильному сочетанию. Восприятие цвета у человека очень индивидуально, поэтому не стоит навязывать всем свое видение цвета. Желательно, чтобы основной цвет формы был нейтральным (например, у большинства приложений Microsoft это светло-серый цвет).

5.1. Разработка оконных приложений

Создание любого оконного приложения осуществляется, как правило, в три этапа:

1. Создание интерфейса приложения, т. е. средств взаимодействия пользователя с программой.

2. Разработка сценария работы будущего приложения. На этом этапе определяют, какая информация будет выводиться на экран, какие события будут происходить при использовании различных компонентов, как приложение должно завершить работу, какие результаты и в каком виде сохранить и т. д.

3. Разработка алгоритма решения поставленной задачи.

Большинство приложений в операционной системе Windows выглядят и ведут себя сходным образом. Компания Microsoft предложила рекомендации для разработки программного обеспечения[1], направленные на то, чтобы пользователь не тратил время на освоение нюансов пользовательского интерфейса новой программы, а сразу начал продуктивно ее использовать. Эти рекомендации основаны на психофизиологических особенностях человека и существенно облегчат жизнь будущим пользователям вашей программы.

Приведем некоторые рекомендации по разработке графического интерфейса оконных приложений (пример 5.1).

[1] https://docs.microsoft.com/ru-ru/windows/uwp/design/basics/design-and-ui-intro

5.2. Стандартные диалоги

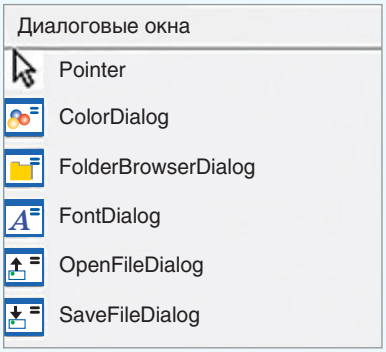
Практически любое приложение Windows использует стандартные диалоги, встроенные в операционную систему, для открытия и сохранения файлов, выбора атрибутов шрифта или установки цвета, поиска текста, печати. В библиотеку VCL включены компоненты, реализующие соответствующие окна Windows. Они размещены на панели **Диалоговые окна** (пример 5.2). В примере 5.3 приведен перечень компонентов для реализации стандартных диалогов.

Объекты на странице **Диалоговые окна** невидимы во время выполнения, поэтому они размещаются в специальной области под формой (пример 5.4). Внешний вид окна диалога зависит от версии Windows.

Вызов и обработка диалогов происходит программно. Для всех диалогов определен метод `ShowDialog()` (пример 5.5). С помощью этого метода происходит открытие окна соответствующего диалога. В свойствах компонента-диалога запоминается выбор пользователя, который затем можно обработать.

Диалоги для открытия и сохранения файлов используются в различных приложениях. Основное свойство компонентов OpenDialog и SaveDialog, в котором возвращается в виде строки имя файла, — это свойство **FileName**. Если задать данное свойство на этапе конструирования в окне инспектора объектов, то при открытии диалога

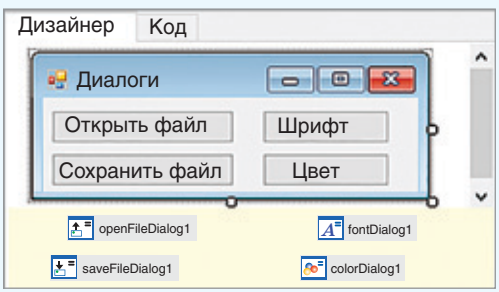
Пример 5.2. Панель **Диалоговые окна**.



Пример 5.3. Список некоторых стандартных диалогов.

Компонент	Назначение
OpenFileDialog	Создание окна диалога «Открыть файл»
SaveFileDialog	Создание окна диалога «Сохранить файл»
FontDialog	Создание окна диалога «Шрифт» — выбор атрибутов шрифта
ColorDialog	Создание окна диалога «Цвет» — выбор цвета

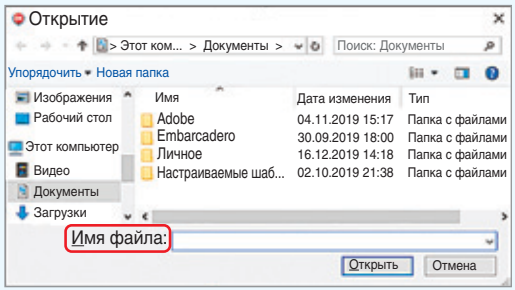
Пример 5.4. Диалоговые компоненты и кнопки для их вызова на форме:



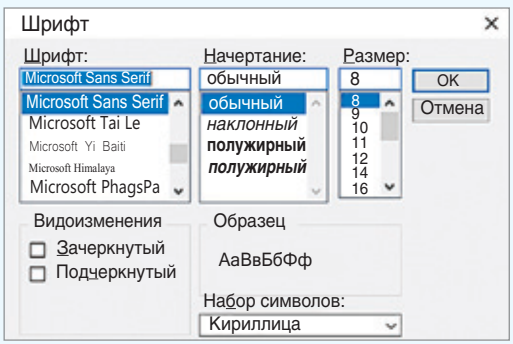
Пример 5.5. Стандартное обращение к диалогу:

```
<имя диалога>. ShowDialog();
```

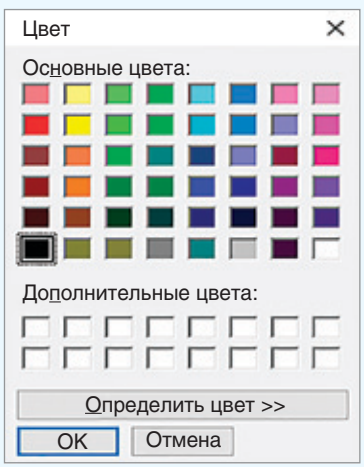
Пример 5.6. Стандартный диалог для открытия файла:



Пример 5.7. Стандартный диалог для настроек шрифта:



Пример 5.8. Стандартный диалог для выбора цвета:



оно будет появляться в строке Имя файла (пример 5.6).

Для вызова стандартного окна установки атрибутов шрифта можно использовать компонент FontDialog (пример 5.7). В окне **Шрифт** пользователь может выбрать имя шрифта, его стиль, размер. Основное свойство компонента — Font.

Для вызова стандартного окна установки цвета используется компонент ColorDialog (пример 5.8). В нем можно выбрать цвет из базовой палитры. Основное свойство компонента ColorDialog — **Color**. Это свойство соответствует тому цвету, который пользователь выбрал в диалоге.

5.3. Создание меню

Практически любое приложение должно иметь меню, которое дает удобный доступ к функциям программы. Существует несколько типов меню:

- **главное меню** с выпадающими списками разделов;
- **каскадные меню**, в которых разделу первичного меню ставится в соответствие список подразделов;
- **контекстные меню**, появляющиеся при нажатии правой клавишей мыши на объекте.

В PascalABC.Net меню создаются компонентами MenuStrip (главное меню) и ContextMenuStrip (контекстное меню), расположенными на панели **Меню и панели инструментов**. Во время выполнения программы сами компоненты не видны, поэтому раз-

мещаются в специальной области под формой (пример 5.9). На этапе выполнения программы главное меню будет помещено на свое стандартное место — наверху формы, контекстное меню появится только после нажатия правой кнопки мыши по тому компоненту, к которому оно относится.

Для добавления новых пунктов меню нужно кликнуть левой клавишей мыши в верхней части формы (там, где обычно располагается меню). Затем заполнить ячейки, соответствующие пунктам меню (пример 5.10).

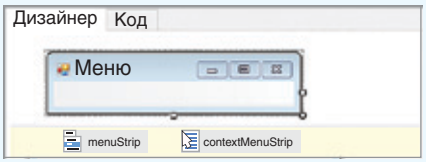
Каждый пункт меню является отдельным объектом. Список всех компонентов, относящихся к меню, можно увидеть в выпадающем списке в инспекторе объектов. Названия пунктов меню прописываются в свойстве Text в окне инспектора объектов (пример 5.11).

Для каждого пункта меню основным событием является событие Click.

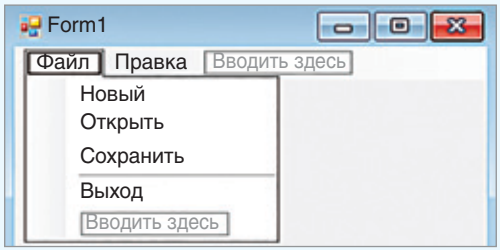
Создание контекстных меню аналогично созданию главного меню. Сначала нужно выбрать компонент на нижней панели, а затем заполнить ячейки. Для того чтобы при щелчке правой кнопкой мыши на некотором компоненте появлялось контекстное меню, нужно написать имя контекстного меню в свойстве ContextMenuStrip для выбранного компонента (пример 5.12).

Написание обработчиков для меню и диалогов будет рассмотрено в следующих пунктах.

Пример 5.9. Меню:

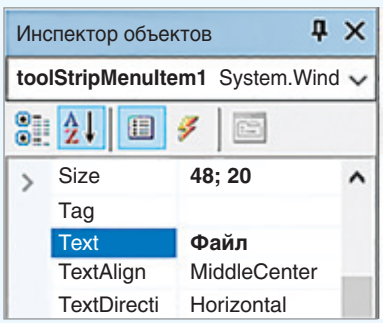


Пример 5.10. Редактирование меню:

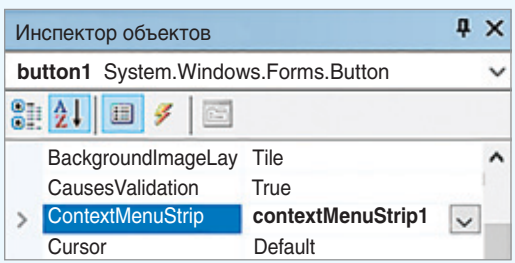


Если в качестве значения свойства Caption ввести «—», то вместо пункта меню появится разделитель.

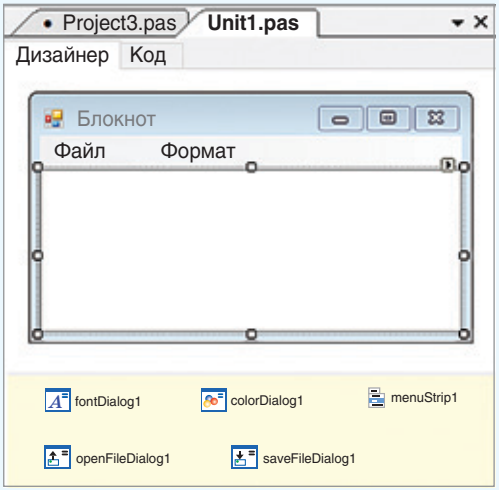
Пример 5.11. Название пункта меню в инспекторе объектов:



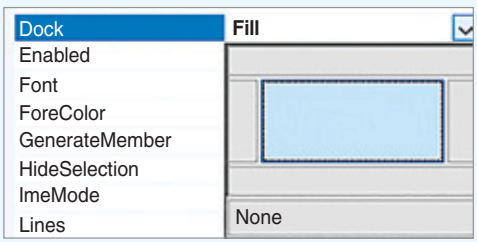
Пример 5.12. Контекстное меню для компонента Button1:



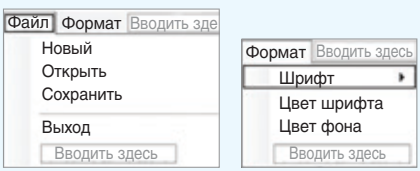
Пример 5.13. Форма на этапе конструирования:



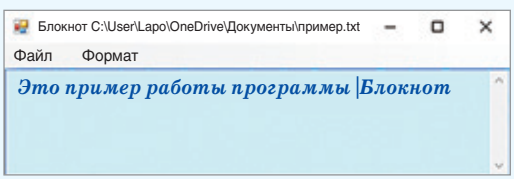
Пример 5.14. Настройка клиентской области:



Пример 5.15. Структура меню:



Пример 5.16. Работающее приложение:



5.4. Создание приложения «Блокнот»

Программа Блокнот должна давать возможность открыть и сохранить текстовый файл, выбрать цвет текста и цвет фона.

Разместить на форме (пример 5.13) следующие компоненты:

- рабочая область для текста — TextBox1;
- диалоги работы с файлами — OpenFileDialog1, SaveFileDialog1;
- диалоги для настройки внешнего вида приложения — FontDialog1, ColorDialog1;
- главное меню — MenuStrip1.

Компонент TextBox1 предназначен для набора и редактирования текста. Текст может набираться в несколько строк, поэтому нужно установить значение true для свойства MiltiLine. Для того чтобы компонент занимал всю клиентскую часть формы, необходимо установить у свойства Dock значение Fill (пример 5.14). Установить значение Vertical для свойства ScrollBars (вертикальная полоса прокрутки).

Структура меню представлена в примере 5.15. Для написания обработчиков пунктов меню нужно в инспекторе объектов выбрать соответствующий пункт меню, перейти на вкладку Events и выбрать событие Click. Поскольку событие Click является событием по умолчанию, то двойной клик по пункту в редакторе меню создаст процедуру-обработчик.

Окно работающего приложения показано в примере 5.16.

Обработчики событий для каждого из пунктов меню представлены в примере 5.17.

Для сохранения и загрузки файлов опишем глобальную переменную F_N:

```
var F_N: String;
```

Обработчик пункта меню **Новый** (StripMenuItem4) очищает строки компонента TextBox1 от введенного ранее текста.

Обработчики пунктов меню **Открыть** (StripMenuItem5) и **Сохранить** (StripMenuItem6) работают с файлом. Имя файла добавляется к заголовку окна.

Обработчик пункта меню **Выход** (StripMenuItem8) закрывает главную форму проекта.

Обработчик пункта меню **Шрифт** (StripMenuItem9) приписывает шрифту, связанному с компонентом TextBox1, свойства, выбранные пользователем.

Обработчики пунктов меню **Цвет текста** (StripMenuItem10) и **Цвет фона** (StripMenuItem11) устанавливают для TextBox1 цвета текста и фона, выбранные пользователем.

Для компонента TextBox определены следующие действия: Копировать (Ctrl + C), Вырезать (Ctrl + X), Вставить (Ctrl + V), Отменить (Ctrl + Z).

5.5. Создание приложения «Графический редактор»

Программа «Графический редактор» должна давать возможность открыть и сохранить файл, выбрать цвет линии и цвет фона, установить толщину линии. Рисование производится выбранным цветом линии при нажатой левой клавише мыши. Клик правой клавишей мыши внутри замкнутой области используется для заливки ограниченной области выбранным цветом фона.

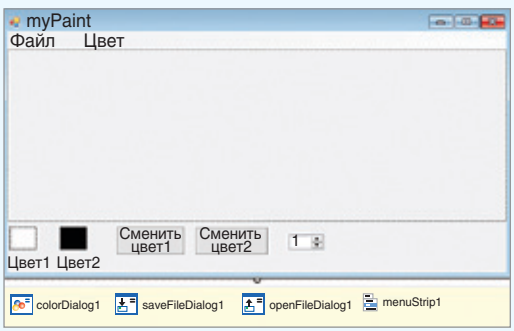
Пример 5.17. Обработчики событий для пунктов меню:

```
var s:string;

procedure
Form1.toolStripMenuItem4_Click
(sender: Object; e: EventArgs);
begin
 //Файл — Новый
 TextBox1.Clear;
end;

procedure
Form1.toolStripMenuItem5_Click
(sender: Object; e: EventArgs);
begin
//Файл — Открыть
 openFileDialog1.ShowDialog();
 s := openFileDialog1.FileName;
 Text := 'Блокнот ' + s;
 TextBox1.Lines := ReadAllLines(s);
end;

procedure
Form1.toolStripMenuItem6_Click
(sender: Object; e: EventArgs);
begin
//Файл — Сохранить
 saveFileDialog1.ShowDialog();
 F_N := saveFileDialog1.FileName;
 WriteAllLines(F_N,
   TextBox1.Lines);
 Text := 'Блокнот ' + F_N;
end;

procedure
Form1.toolStripMenuItem8_Click
(sender: Object; e: EventArgs);
begin
 //Файл — Выход
 close;
end;
```

Пример 5.17. *Продолжение.*

```
procedure Form1.
toolStripMenuItem9_Click
(sender: Object; e: EventArgs);
begin
 //Формат — Шрифт
 fontDialog1.ShowDialog();
 TextBox1.Font := fontDialog1.
Font;
end;
procedure Form1.
toolStripMenuItem10_
Click(sender: Object; e:
EventArgs);
begin
 //Формат — Цвет текста
 colorDialog1.ShowDialog();
 TextBox1.ForeColor :=
   colorDialog1.Color;
end;
procedure Form1.
toolStripMenuItem11_
Click(sender: Object; e:
EventArgs);
begin
 //Формат — Цвет фона
 colorDialog1.ShowDialog();
 TextBox1.BackColor :=
   colorDialog1.Color;
end;
```

Пример 5.18. Форма на этапе конструирования:



Сначала спроектируем форму, разместив на ней следующие компоненты (пример 5.18):

- область для рисования — PictureBox;
- компоненты, отображающие выбранный цвет для рисования и цвет фона — Panel1, Panel2;
- кнопки для смены цвета;
- компонент выбора цвета — ColorDialog1;
- компонент для выбора толщины линии — numericUpDown1 (панель компонентов **Стандартные элементы управления**);
- главное меню — menuStrip1 и компоненты для работы с файлами — OpenFileDialog1, SaveFileDialog1.

На этапе конструирования установить значение свойства BackColor у компонентов Panel1 и Panel2 — Black и White соответственно.

У свойств Value и Minimum для компонента numericUpDown1 установить значение 1.

Структура меню показана в примере 5.19. Создание рисунка и загрузка файла в приложение приведены в примере 5.20.

В обработчике события Load для формы прописаны первоначальные установки для создания графического объекта, заданы параметры кисти и карандаша.

В обработчике события MouseDown для компонента PaintBox1 задаем переменной m_d значение true — кнопка нажата. Здесь же запоминаем координаты точки, поскольку от этой точки начнем строить линию. В обработчи-

ке MouseUp — значение переменной m_d = false — кнопка не нажата.

Для отслеживания траектории движения мыши по компоненту PaintBox1 создаем обработчик события MouseMove. Если кнопка нажата, то можем строить линию. Параметры e.x, e.y возвращают координаты точки, в которой произошло нажатие кнопки.

Для перемещения мыши нужно использовать метод DrawLine (x1, y1, e.x, e.y) — рисование линии, соединяющей две точки. После прорисовки обновляем координаты.

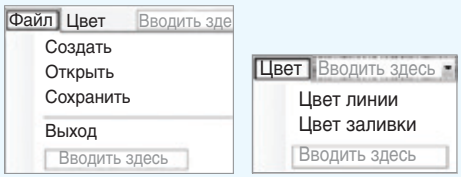
Толщина линии определяется значением свойства Value для компонента numericUpDown1. Обработчик события — ValueChanged.

Обработчики событий для компонентов OpenFileDialog1, SaveFileDialog1 вызываются из соответствующих пунктов меню и аналогичны обработчикам, описанным для программы Блокнот. Для сохранения и загрузки файлов нужно описать глобальную строковую переменную FileName. Приложение может сохранять и загружать файлы формата BMP.

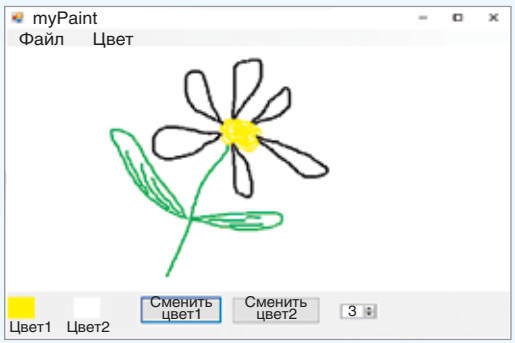
С помощью компонента ColorDialog1 можно выбрать цвет линии или заливки. Пункты меню **Цвет** позволяют выбрать цвет линии или заливки соответственно.

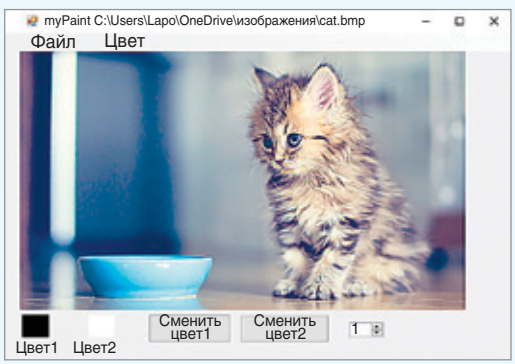
В примере 5.21 приведено описание глобальных переменных, которые используются для создания приложения «Графический редактор». Обработчики всех описанных событий приведены в приложении.

Пример 5.19. Структура меню:



Пример 5.20. Работающее приложение:

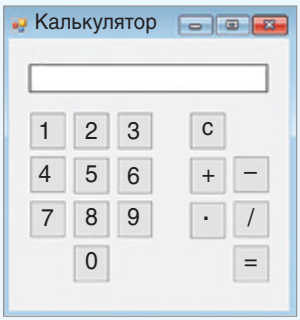




Пример 5.21. Описание глобальных переменных.

```
var gr: Graphics;
   bm: Bitmap;
   p_c: Pen;
   s_b: SolidBrush;
   c_f, c_b: Color;
   w: decimal;
   x1, y1, x2, y2: integer;
   m_d : boolean;
   F_N :string;
```

Пример 5.22. Форма на этапе конструирования:



Пример 5.23. Обработчики событий:

```
//описание глобальных переменных
var n1, n2: integer;
  znak: char;
procedure Form1.button1_Click
(sender: Object; e: EventArgs);
begin
 //приписывание цифры к числу
 TextBox1.Text :=TextBox1.Text+'1';
end;
procedure Form1.button9_Click
(sender: Object; e: EventArgs);
begin
 //приписывание цифры к числу
 TextBox1.Text :=TextBox.Text1+'2';
end;
```

Для остальных цифровых кнопок нужно изменить только '1' на соответствующую цифру.

```
procedure Form1.button12_Click
 (sender: Object; e: EventArgs);
begin
 n1 := StrToInt(TextBox1.Text);
 //запоминание знака операции
 znak := '+';
 TextBox1.Clear;
end;
```

Для остальных кнопок со знаками арифметических действий нужно изменить только '+' на соответствующий знак.

5.6. Создание приложения «Калькулятор»

Создание калькулятора начнем с конструирования формы. На ней нужно разместить: поле TextBox для ввода/вывода чисел, 10 кнопок с цифрами, 4 кнопки с арифметическими действиями, кнопку «=» и кнопку «C» — очистить (пример 5.22).

При нажатии на кнопку с цифрой программа должна дописать эту цифру к числу в поле TextBox. При нажатии на кнопку с арифметическим действием нужно запомнить число, которое в данный момент находится в поле TextBox, и очистить поле для ввода второго числа. Числа будем хранить в двух переменных *n*1, *n*2 типа integer. Знак операции будем хранить в переменной *znak* типа **char**. Переменные описываются как глобальные. При нажатии на кнопку «=» выполняется арифметическое действие и выводится результат.

Кнопки могут содержать рисунок на поверхности (например, изображения с цифрами). Свойство для размещения рисунка — BackgroundImage.

Установить значение FixedSingle для свойства FormBorderStyle формы. В этом случае граница формы не позволит менять ее размеры.

Коды процедур-обработчиков приведены в примере 5.23.

Для каждой кнопки на форме нужно создать обработчик события Click.

Обработчики событий для всех цифровых кнопок будут идентичны.

Обработчики для кнопок арифметических действий будут отличаться

только значением запоминаемой операции.

Основные вычисления происходят в обработчике кнопки «=». Преобразуем в число n2 значение поля Edit и выполняем арифметическую операцию в зависимости от значения переменной znak. После этого обнуляем переменные.

В обработчике кнопки «C» (от англ. clear — очистить) происходит обнуление переменных и очистка поля Edit.

Созданный калькулятор имеет большое количество ограничений в своей работе, поскольку рассчитан на вычисления только с натуральными числами.

Пример 5.23. *Продолжение.*

```
procedure Form1.button16_Click
(sender: Object; e: EventArgs);
begin
 n2 := StrToInt(TextBox1.Text);
 case znak of
  '+': TextBox1.Text:= IntToStr(n1 + n2);
  '-': TextBox1.Text:= IntToStr(n1 - n2);
  '*': TextBox1.Text:= IntToStr(n1 * n2);
  '/': TextBox1.Text:=IntToStr(n1 div n2);
 end;
 n1 := 0;  n2 := 0;
 znak := ' ';
end;
```

```
procedure Form1.button15_Click
(sender: Object; e: EventArgs);
begin
 TextBox1.Clear;
 n1 := 0;  n2 := 0;
 znak :=' ';
end;
```

Упражнения

1 Дополните проект Блокнот следующими возможностями:

1. Пункт меню Сохранить файл заменить двумя: Сохранить и Сохранить как...
2. Добавить Пункт меню Переносить по словам.
3. Добавьте возможность управления полосами прокрутки: установить горизонтальную, вертикальную, обе.

4*. Добавьте следующие диалоги: Параметры страницы, Печать, Поиск, Замена и соответствующие пункты в меню.

5. Добавьте контекстное меню для управления компонентом Memo. Команды контекстного меню, дублирующие команды основного меню, должны вызывать те же обработчики.
6. Предложите свои возможности.

2 Для проекта Графический редактор добавьте следующие возможности:

1. Рисовать отрезки, овалы и прямоугольники.
2. Предложите свои возможности.

3 Для проекта Калькулятор добавьте следующие возможности:

1. Возможность работы с вещественными числами.
2. Возможность вычислять значения функций: извлечение квадратного корня, тригонометрические функции (для градусов и радиан) и др.

3*. Возможность перевода чисел в другие системы счисления.