Создание приложений

11 класс

Сегодня на уроке мы…

- ознакомимся с элементами управления для работы с графикой;
- рассмотрим элементы управления PictureBox, компонентом Timer и их свойствами;
- научимся создавать проекты используя элементы управления для работы с графикой.

Разработка оконных приложений

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

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

1. 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.

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.

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

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, 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.

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.

Рекомендации по созданию оконных приложений.

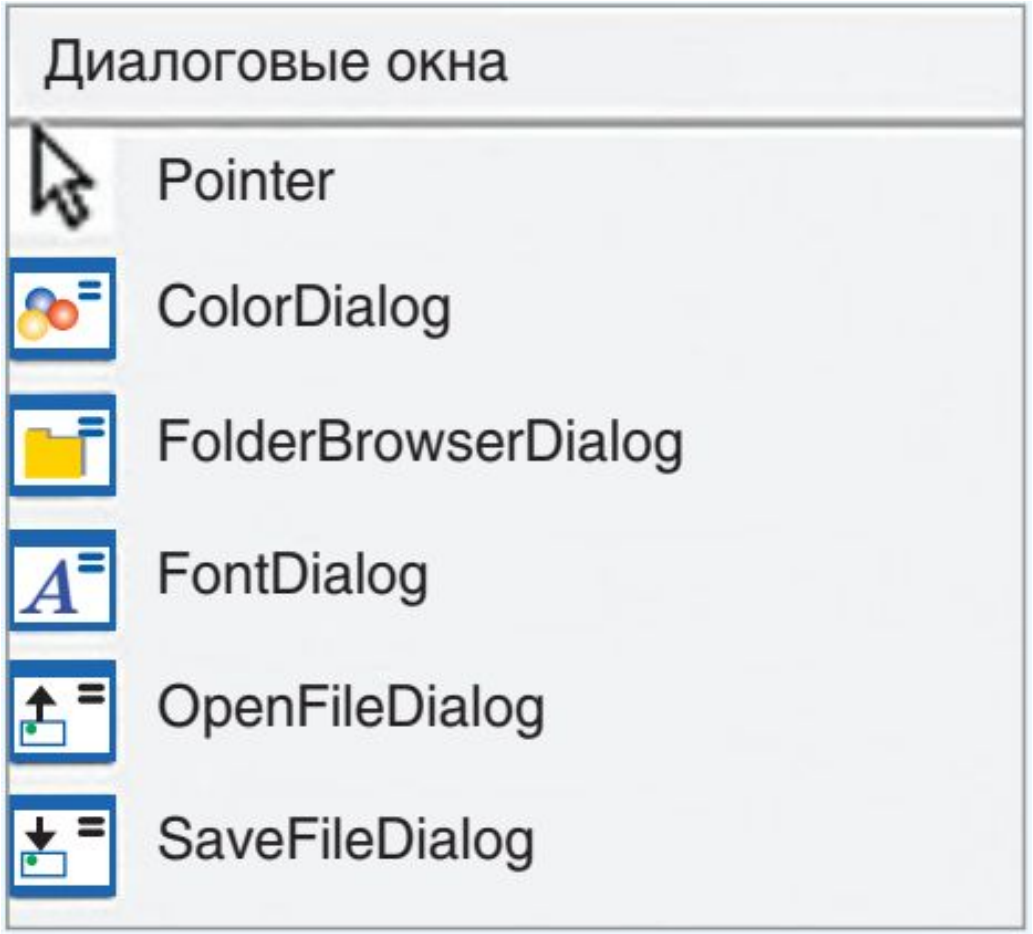
5. При нажатии клавиши F1 должен загружаться файл справки.
6. В программе желательно реализовать возможность настройки и сохранения настроек, чтобы при следующем сеансе работы их не пришлось устанавливать заново.
7. Если результат работы приложения зависит от каких либо параметров, обязательно укажите значения по умолчанию. Они позволят ускорить взаимодействие пользователя с программой, а также являются примером того, в каком формате данные следует вводить.

Рекомендации по созданию оконных приложений.

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).

Стандартные диалоги

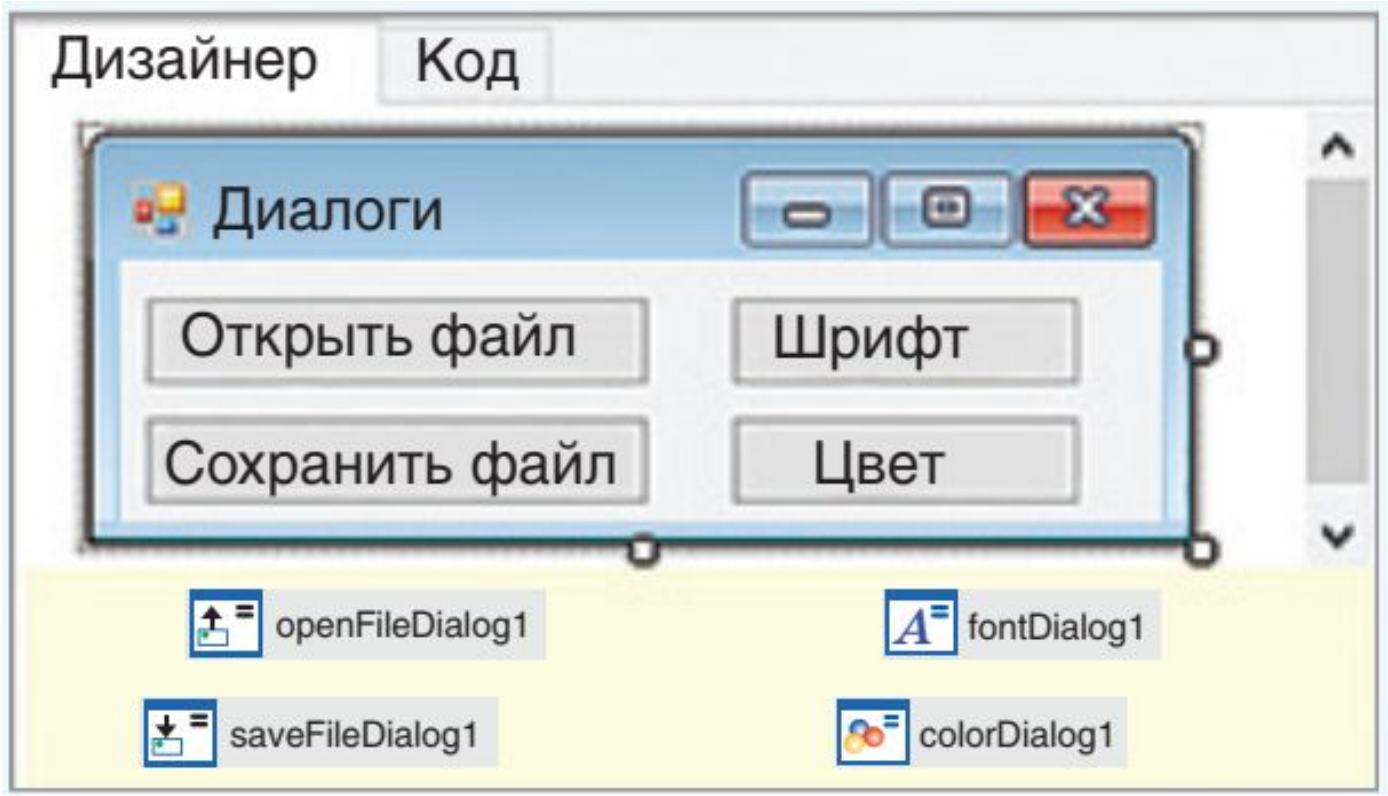
Практически любое приложение Windows использует стандартные диалоги, встроенные в операционную систему, для открытия и сохранения файлов, выбора атрибутов шрифта или установки цвета, поиска текста, печати. В библиотеку **VCL** включены компоненты, реализующие соответствующие окна Windows. Они размещены на панели **Диалоговые окна**.



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

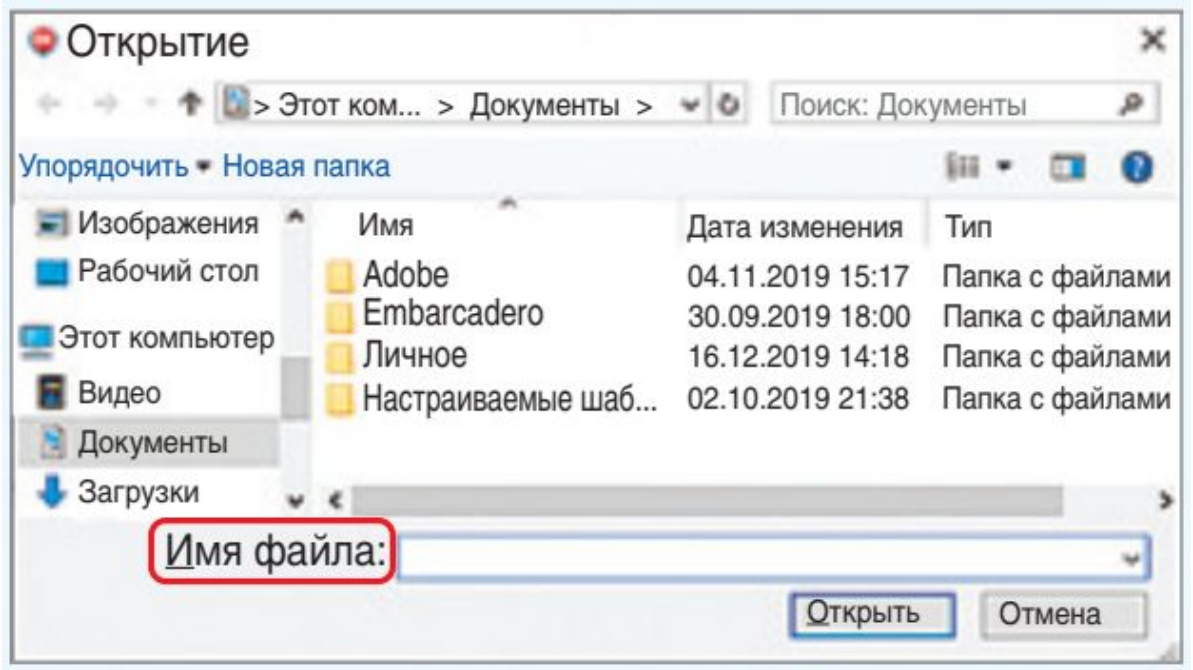
Объекты на странице **Диалоговые окна** невидимы во время выполнения, поэтому они размещаются в специальной области под формой. Внешний вид окна диалога зависит от версии Windows.



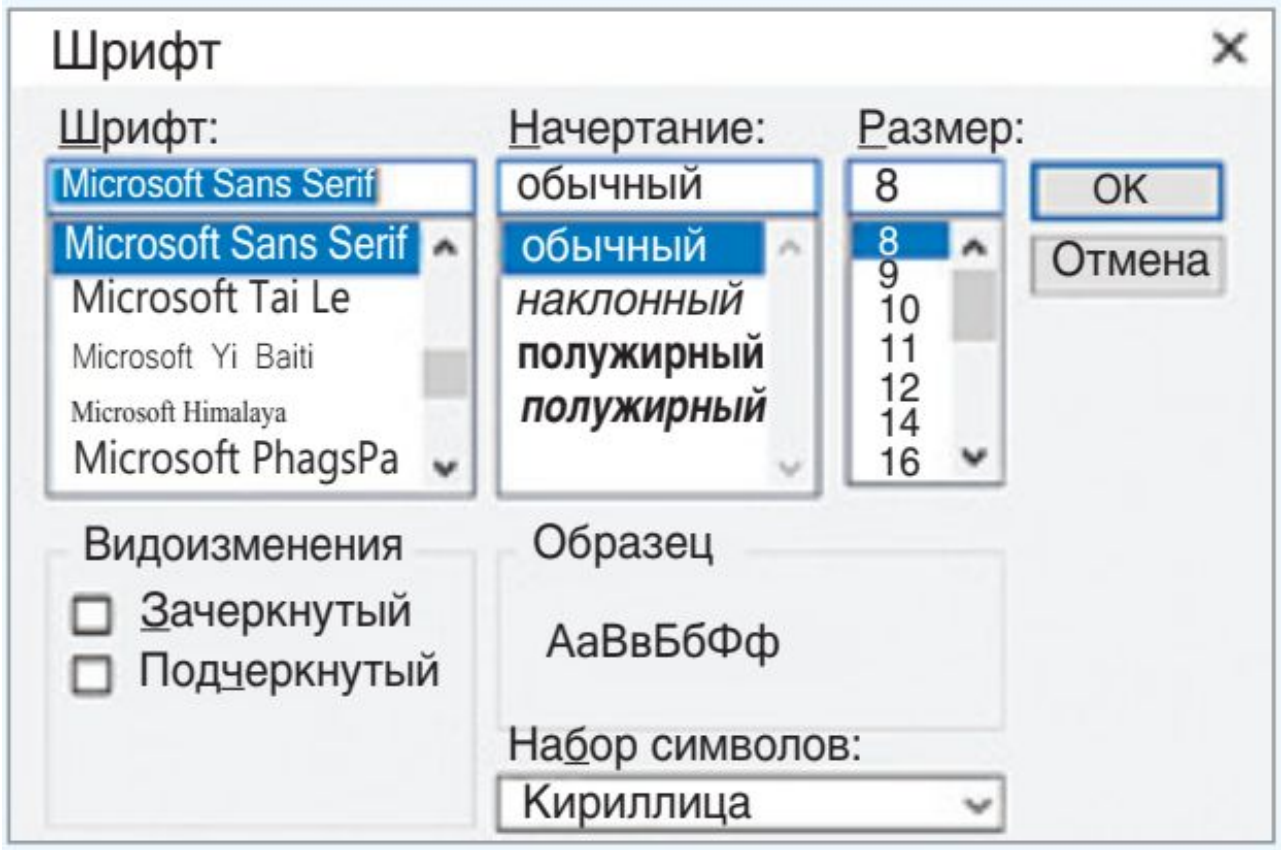
Вызов и обработка диалогов происходит программно. Для всех диалогов определен метод **`ShowDialog()`**. С помощью этого метода происходит открытие окна соответствующего диалога. В свойствах компонента-диалога запоминается выбор пользователя, который затем можно обработать.

```
<имя диалога>.ShowDialog();
```

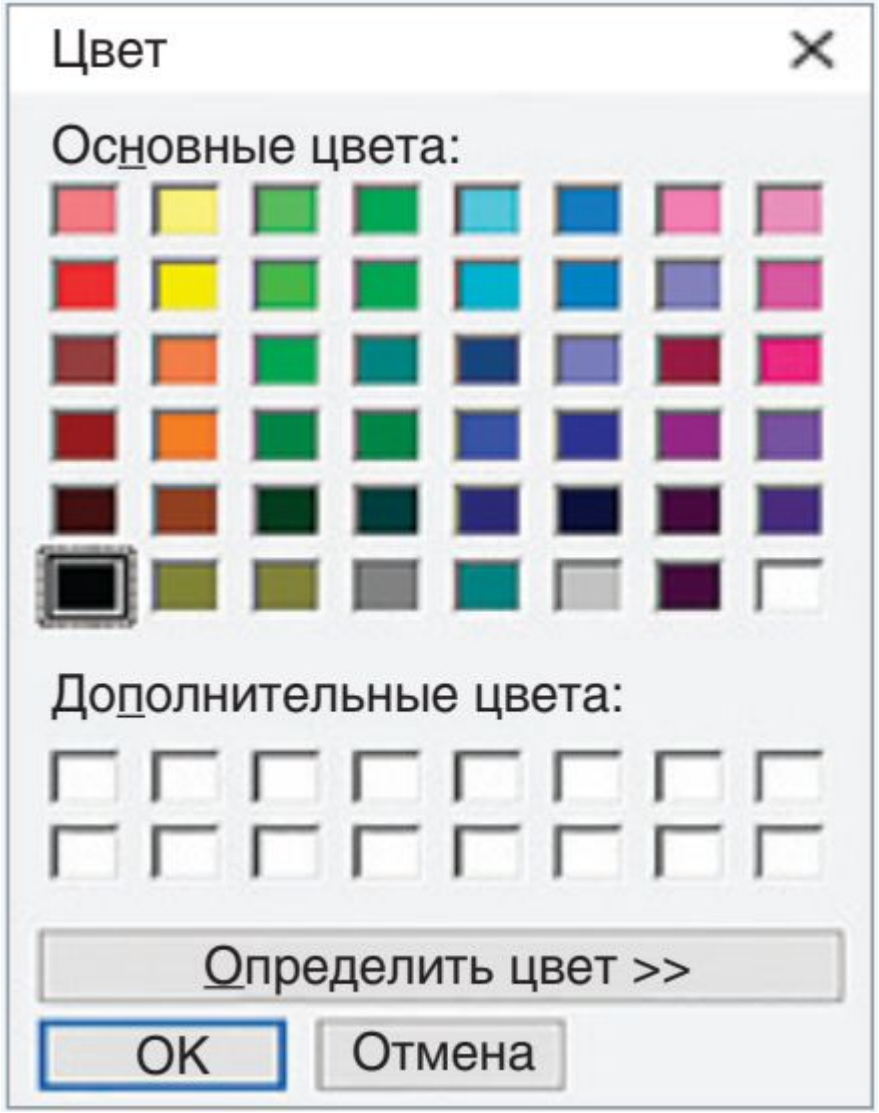
Диалоги для открытия и сохранения файлов используются в различных приложениях. Основное свойство компонентов **OpenDialog** и **SaveDialog**, в котором возвращается в виде строки имя файла, — это свойство ***FileName***. Если задать данное свойство на этапе конструирования в окне инспектора объектов,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



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



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



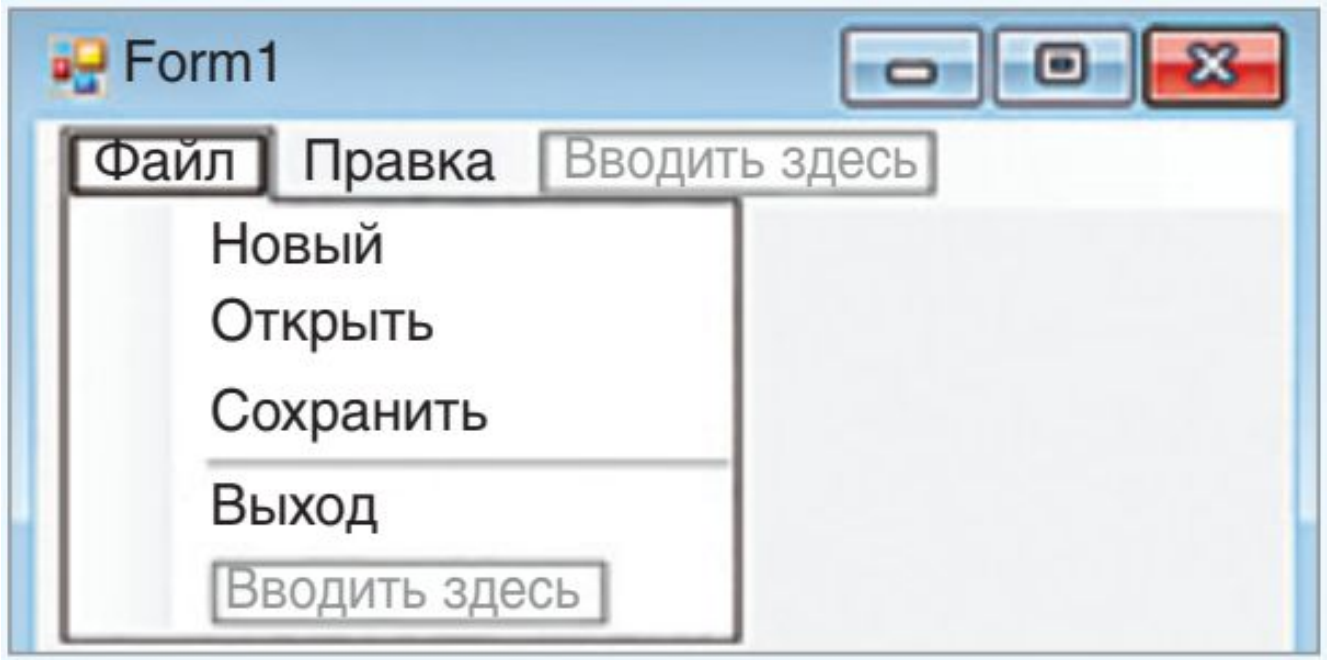
Создание меню

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

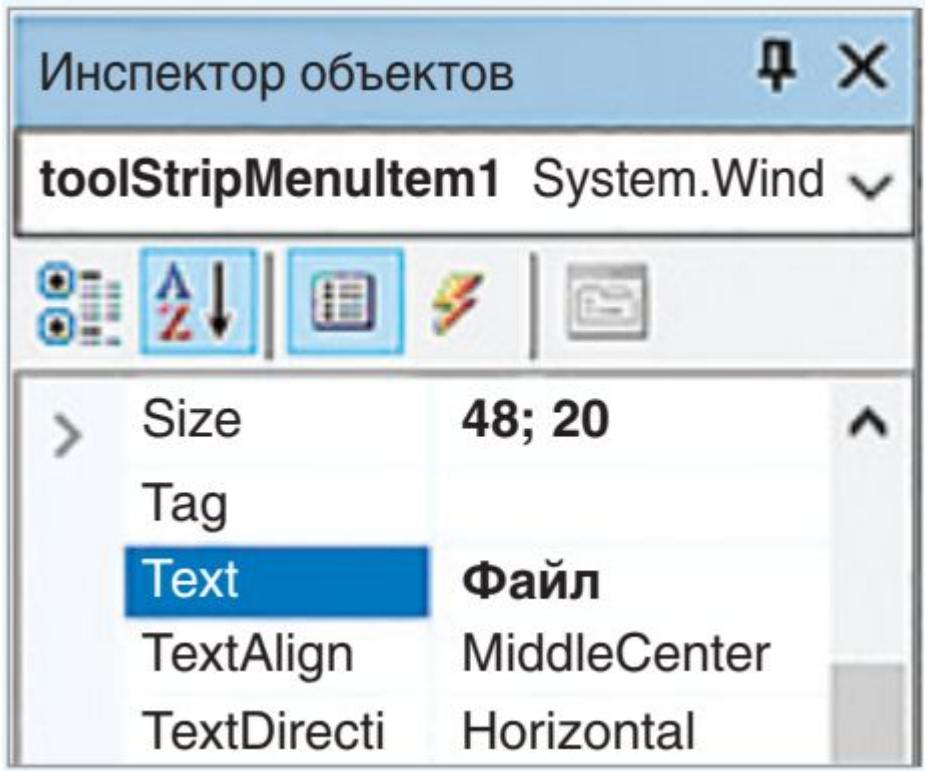
- **главное меню** с выпадающими списками разделов;
- **каскадные меню**, в которых разделу первичного меню ставится в соответствие список подразделов;
- **контекстные меню**, появляющиеся при нажатии правой клавишей мыши на объекте.

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 размещаются в специальной области под формой.

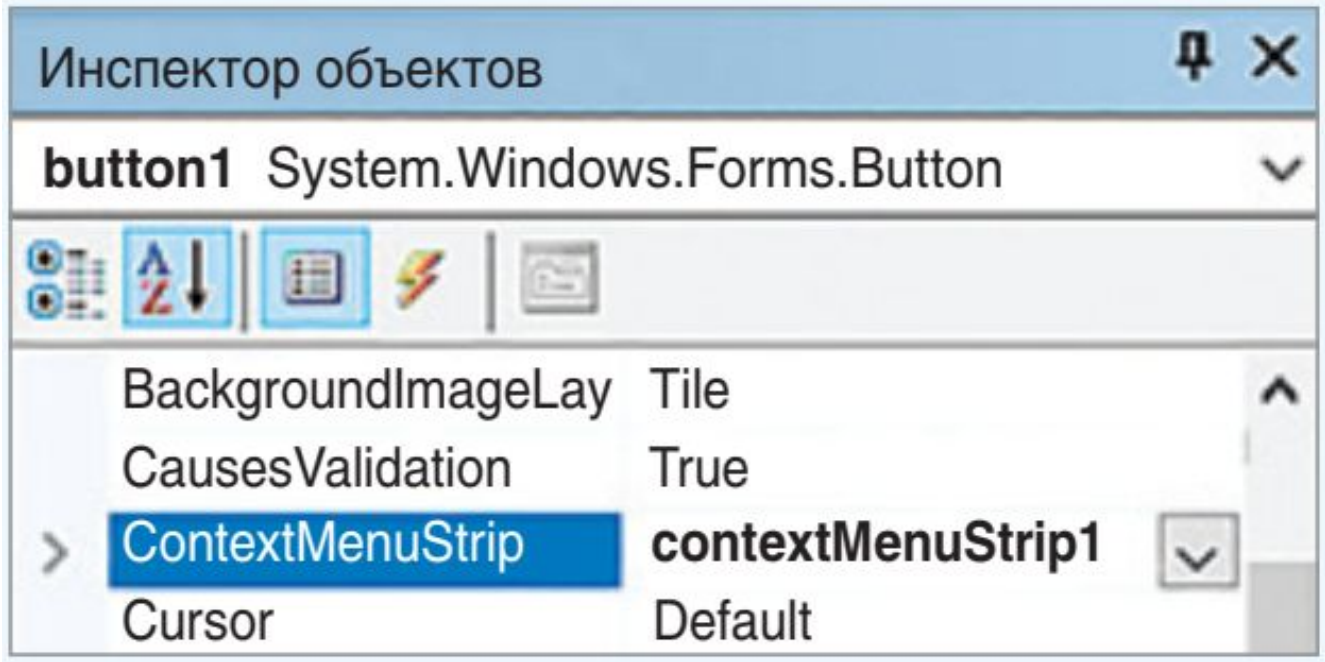
На этапе выполнения программы **главное меню** будет помещено на свое стандартное место — наверху формы, контекстное меню появится только после нажатия правой кнопки мыши по тому компоненту, к которому оно относится. Для добавления новых пунктов меню нужно кликнуть левой клавишей мыши в верхней части формы (там, где обычно располагается меню). Затем заполнить ячейки, соответствующие пунктам меню.



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



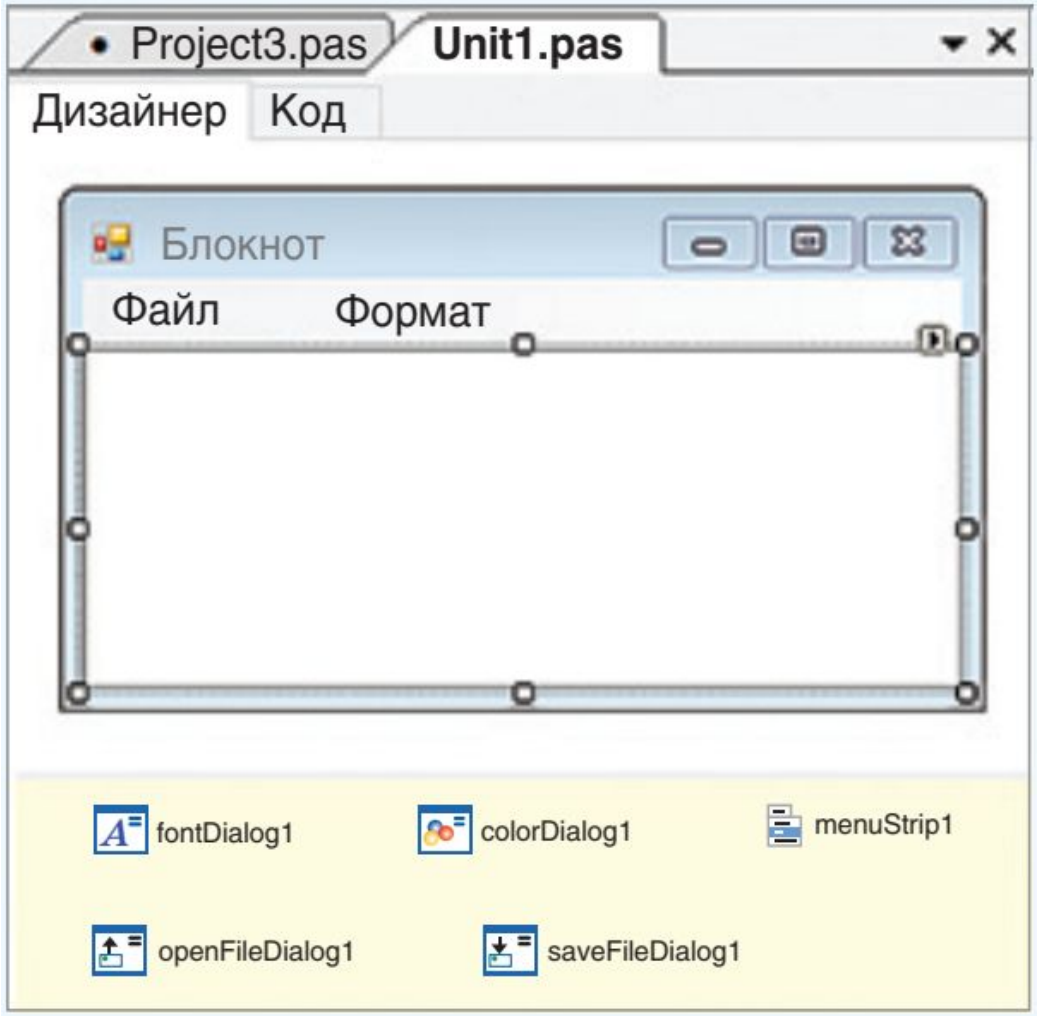
Для каждого пункта меню основным событием является событие ***Click***. Создание контекстных меню аналогично созданию главного меню. Сначала нужно выбрать компонент на нижней панели, а затем заполнить ячейки. Для того чтобы при щелчке правой кнопкой мыши на некотором компоненте появлялось контекстное меню, нужно написать имя контекстного меню в свойстве **ContextMenuStrip** для выбранного компонента.



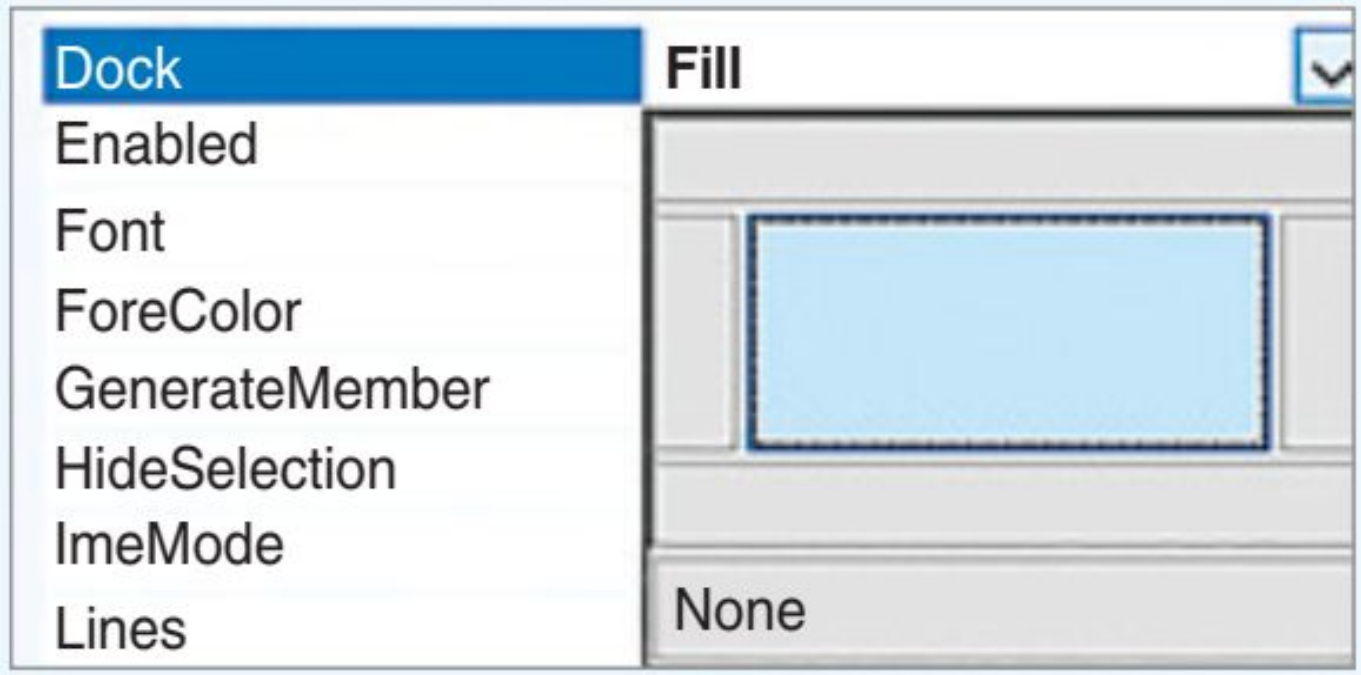
Создание приложения «Блокнот»

Программа Блокнот должна давать возможность открыть и сохранить текстовый файл, выбрать цвет текста и цвет фона. Разместить на форме следующие компоненты:

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

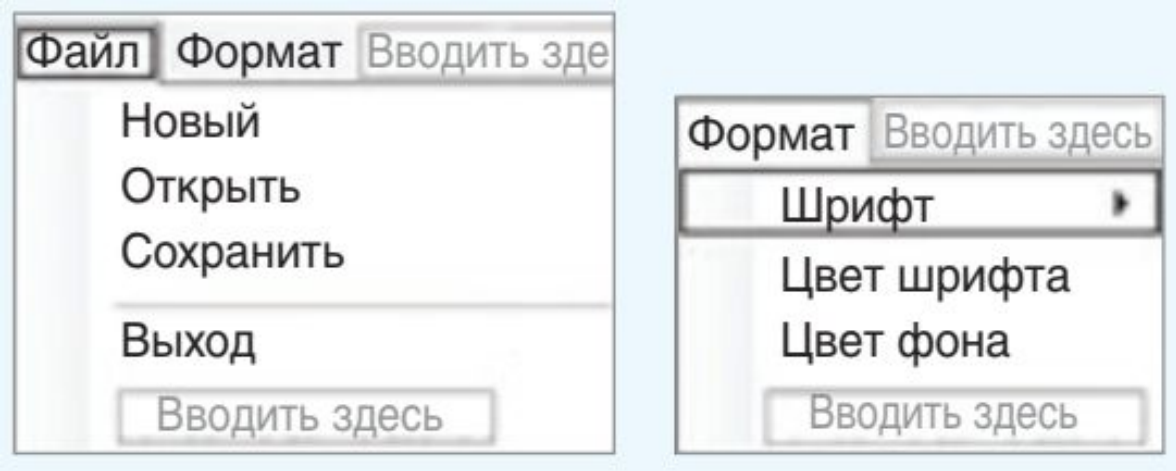


Компонент **TextBox1** предназначен для набора и редактирования текста. Текст может набираться в несколько строк, поэтому нужно установить значение ***true*** для свойства **MiltiLine**. Для того чтобы компонент занимал всю клиентскую часть формы, необходимо установить у свойства **Dock** значение ***Fill***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

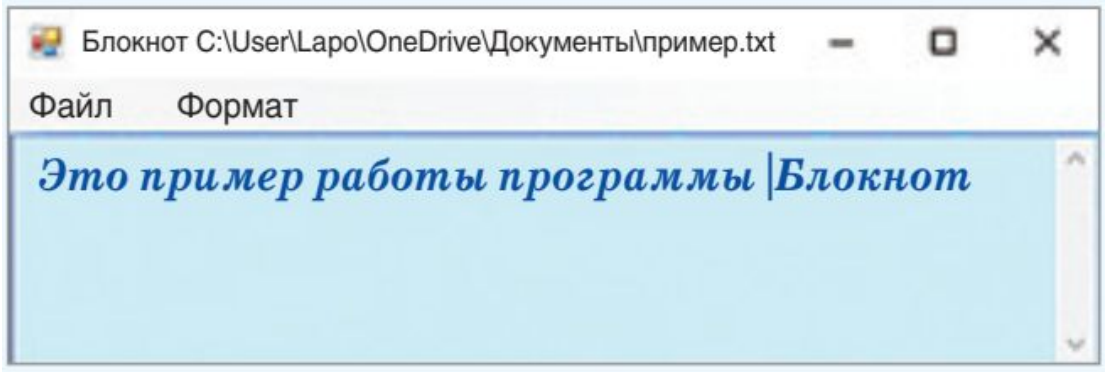


Для написания обработчиков пунктов меню нужно в инспекторе объектов выбрать соответствующий пункт меню, перейти на вкладку **Events** и выбрать событие ***Click***. Поскольку событие ***Click*** является событием по умолчанию, то двойной клик по пункту в редакторе меню создаст процедуру-обработчик.

Структура меню:



Работающее приложение:



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.

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

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

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

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

```
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;
```

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

```
procedure Form1.toolStripMenuItem9_Click(sender: Object;
e: EventArgs);
begin
 //Формат – Шрифт
 fontDialog1.ShowDialog(); TextBox1.Font := fontDialog1.
Font;
end;
```

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

```
procedure Form1.
toolStripMenuItem10_
Click(sender: Object; e:
EventArgs);
begin
 //Формат – Цвет текста
 colorDialog1.ShowDialog();
 TextBox1.ForeColor :=
  colorDialog1.Color;
end;
```

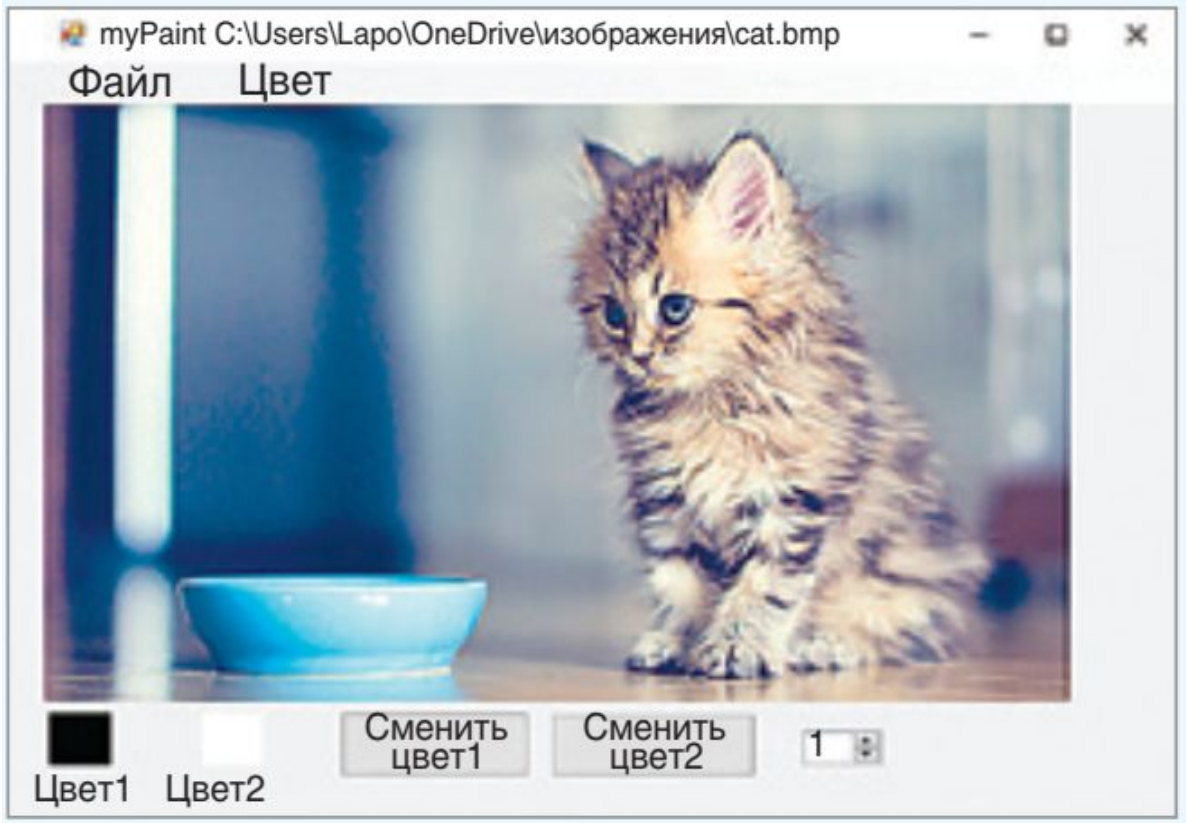
```
procedure Form1.
toolStripMenuItem11_
Click(sender: Object; e:
EventArgs);
begin
 //Формат – Цвет фона
 colorDialog1.ShowDialog();
 TextBox1.BackColor :=
  colorDialog1.Color;
end;
```

Для компонента **TextBox** определены следующие действия:

Копировать ***(Ctrl + C);***
Вырезать ***(Ctrl + X);***
Вставить ***(Ctrl + V);***
Отменить ***(Ctrl + Z).***

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.

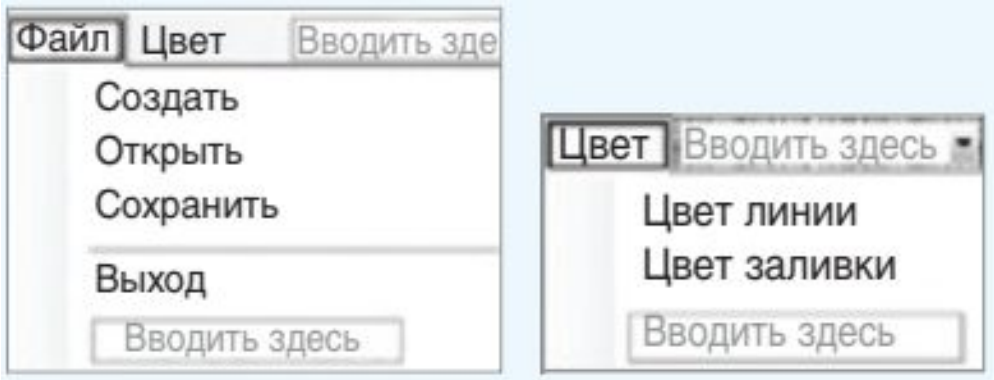


Сначала спроектируем форму, разместив на ней следующие компоненты :

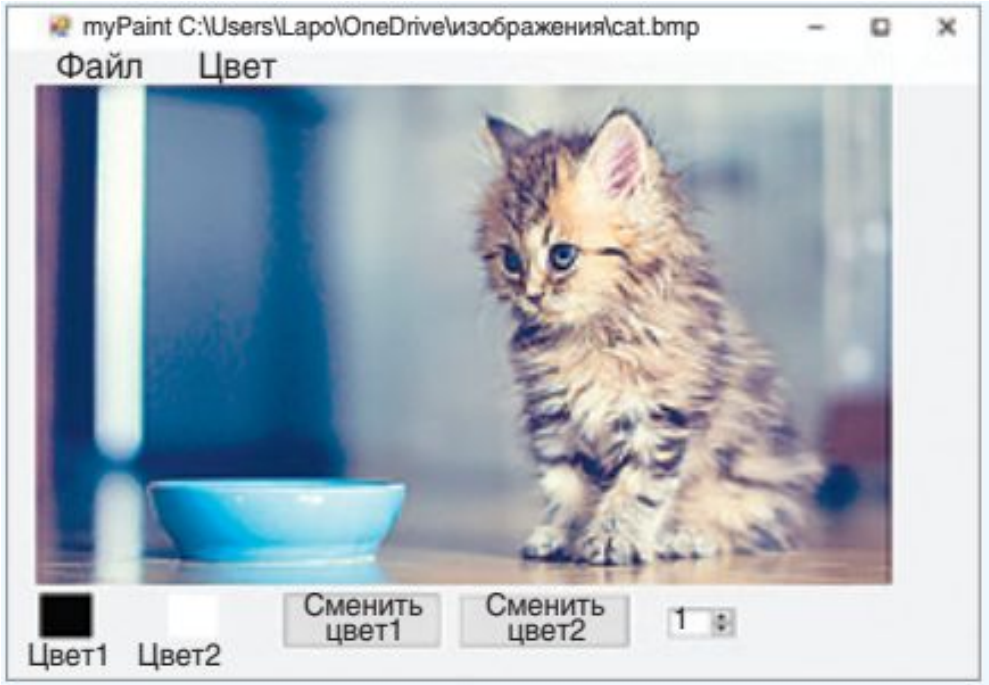
- область для рисования — **PictureBox**;
- компоненты, отображающие выбранный цвет для рисования и цвет фона — **Panel1**, **Panel2**;
- кнопки для смены цвета; компонент выбора цвета — **ColorDialog1**;
- компонент для выбора толщины линии — **numericUpDown1** (панель компонентов Стандартные элементы 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**.

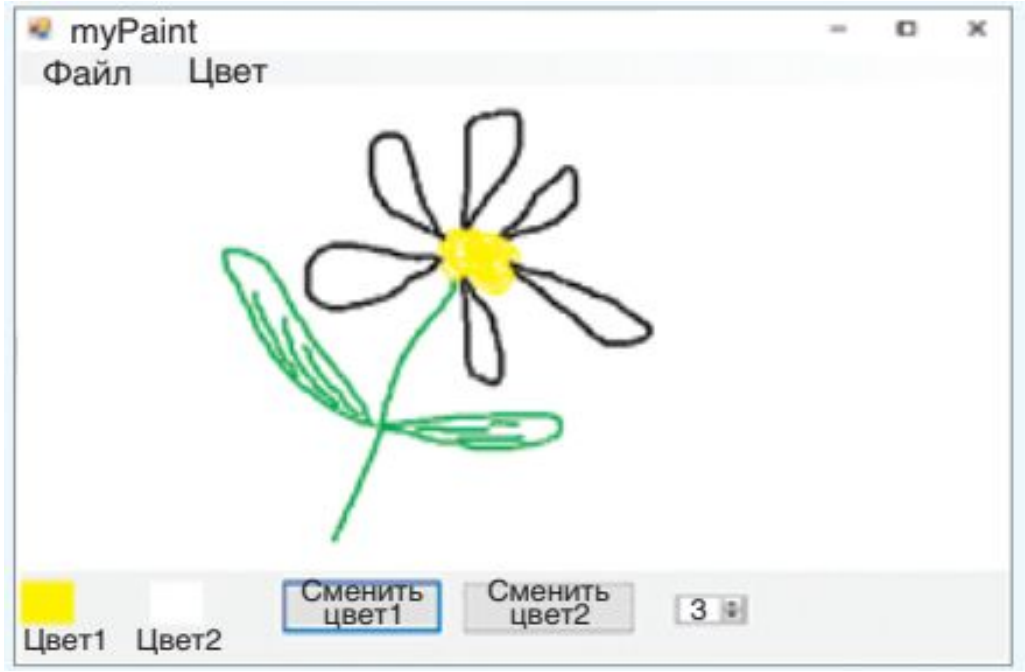
Структура меню:



Загрузка файла в приложение



Создание рисунка



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 обработчике **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**.

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;
```

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

Обработчики событий приложения «Графический редактор»

```
procedure Form1.Form1_Load(sender: Object; e: EventArgs);
begin
 bm := new Bitmap(pictureBox1.Width, pictureBox1.Height);
 pictureBox1.Image := (Image)(bm);
 gr := Graphics.FromImage(pictureBox1.Image);
 gr.Clear(Color.White);
 gr := pictureBox1.CreateGraphics;
 c_f := Color.Black;
 p_c := new Pen(c_f, 1);
 c_b := Color.White;
 s_b := new SolidBrush(c_b);
end;

procedure Form1.numericUpDown1_ValueChanged(sender: Object;
e: EventArgs);
begin
 p_c.Dispose;
 w := numericUpDown1.Value;
 p_c := new Pen(c_f, (integer)(w));
end;
```

```
procedure Form1.button1_Click(sender: Object; e: EventArgs);

begin
 colorDialog1.ShowDialog();
 c_f := colorDialog1.color;
 w := numericUpDown1.Value;
 p_c.Dispose;
 p_c := new Pen(c_f, (integer)(w));
 panel1.BackColor := c_f;
end;

procedure Form1.button2_Click(sender: Object; e: EventArgs);
begin
 colorDialog1.ShowDialog();
 c_b := colorDialog1.color;
 s_b.Dispose;
 s_b := new SolidBrush(c_b);
 panel2.BackColor := c_b;
end;

procedure Form1.pictureBox1_MouseDown(sender: Object;
e: MouseEventArgs);
begin
 m_d := true;
 x1 := e.x; y1 := e.y;
end;
```

```
procedure Form1.pictureBox1_MouseMove(sender: Object;
e: MouseEventArgs);
begin
 if m_d then

 begin
  gr.DrawLine(p_c, x1, y1, e.X, e.Y);
 end;
 //запомнить координаты для рисования следующего отрезка
 x1 := e.X; y1 := e.Y
end;

procedure Form1.pictureBox1_MouseUp(sender: Object;
e: MouseEventArgs);
begin
  m_d := false;
end;

procedure Form1.toolStripMenuItem3_Click(sender: Object;
e: EventArgs);
begin
 gr := pictureBox1.CreateGraphics;
 gr.Clear(c_b);
end;
```

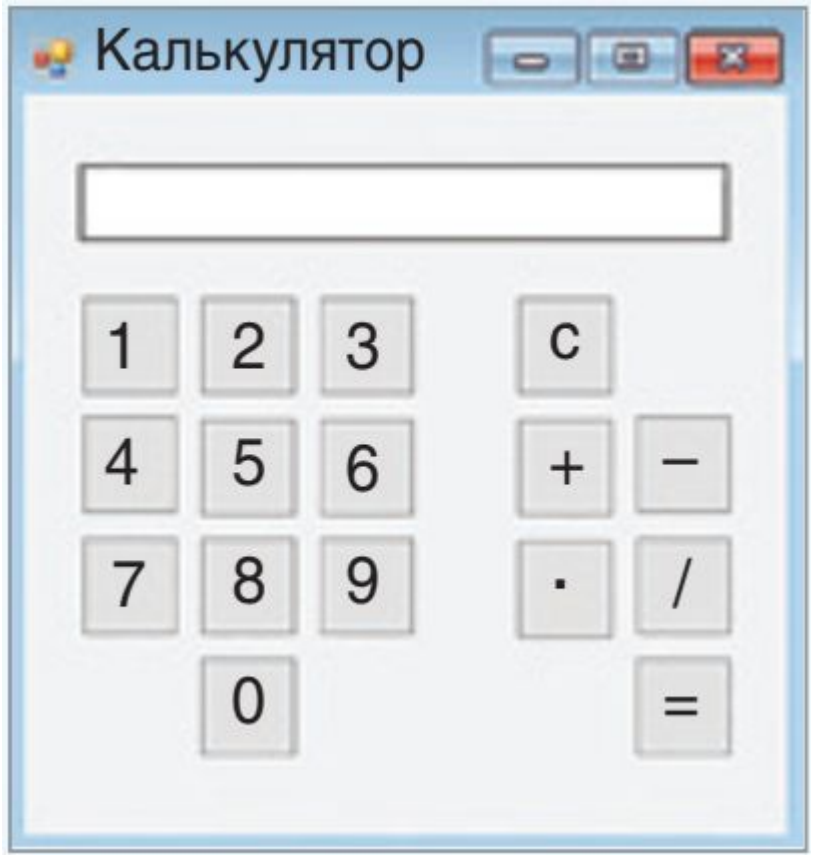
```
procedure Form1.toolStripMenuItem4_Click(sender: Object;
e: EventArgs);
begin
 openFileDialog1.ShowDialog();
 F_N := openFileDialog1.FileName;
 Text := 'myPaint ' + F_N;
 PictureBox1.Load(F_N);
end;

procedure Form1.toolStripMenuItem5_Click(sender: Object;
e: EventArgs);
begin
 saveFileDialog1.ShowDialog();
 F_N := saveFileDialog1.FileName;
 PictureBox1.Image.save(F_N);
 Text := 'myPaint ' + F_N;
end;

procedure Form1.numericUpDown1_ValueChanged (sender: Object;
e: EventArgs);
begin
 p_c.Dispose;
 w := numericUpDown1.Value;
 p_c := new Pen(c_f, (integer)(w));
end;
```

```
procedure Form1.toolStripMenuItem7_Click(sender: Object;
e: EventArgs);
begin
 close;
end;
```

Алгоритм создания приложения «Графический редактор»

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. 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 ***n1***, ***n2*** 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.

Алгоритм создания приложения «Калькулятор»

Домашнее задание

§ 5