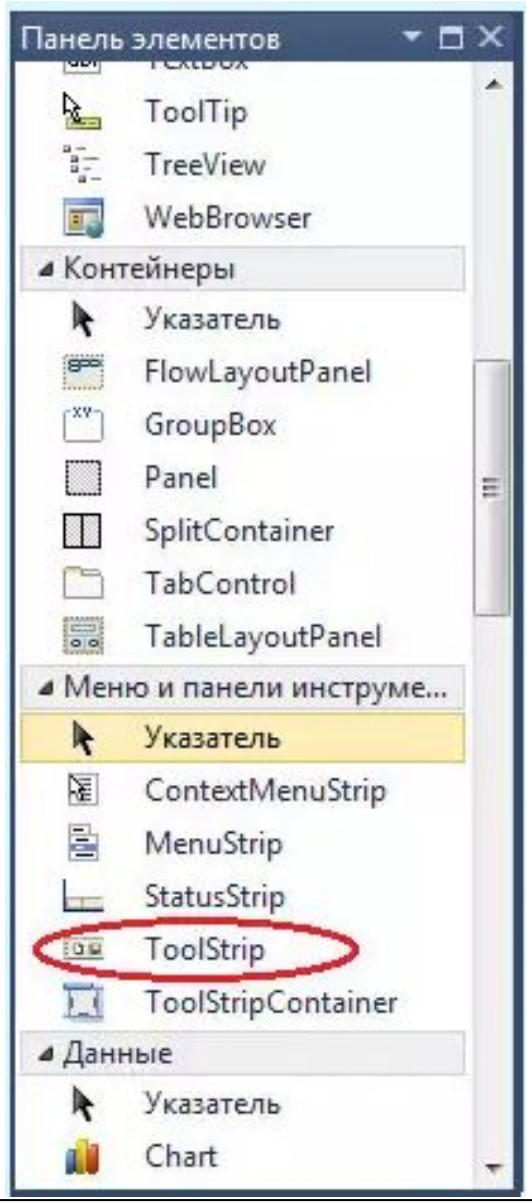
Работа с файлами

Объектно-ориентированное программирование

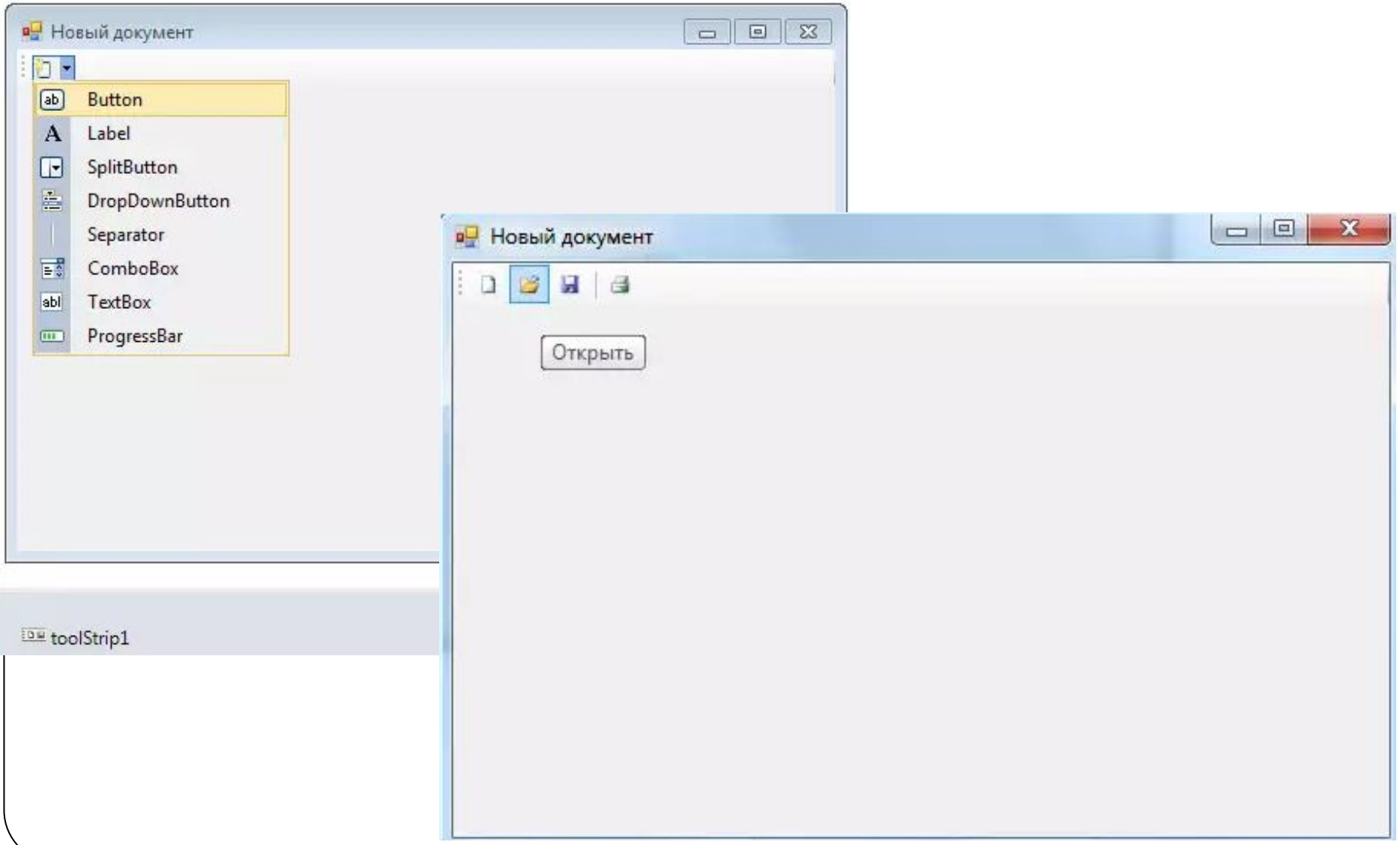
Компонент ToolStrip



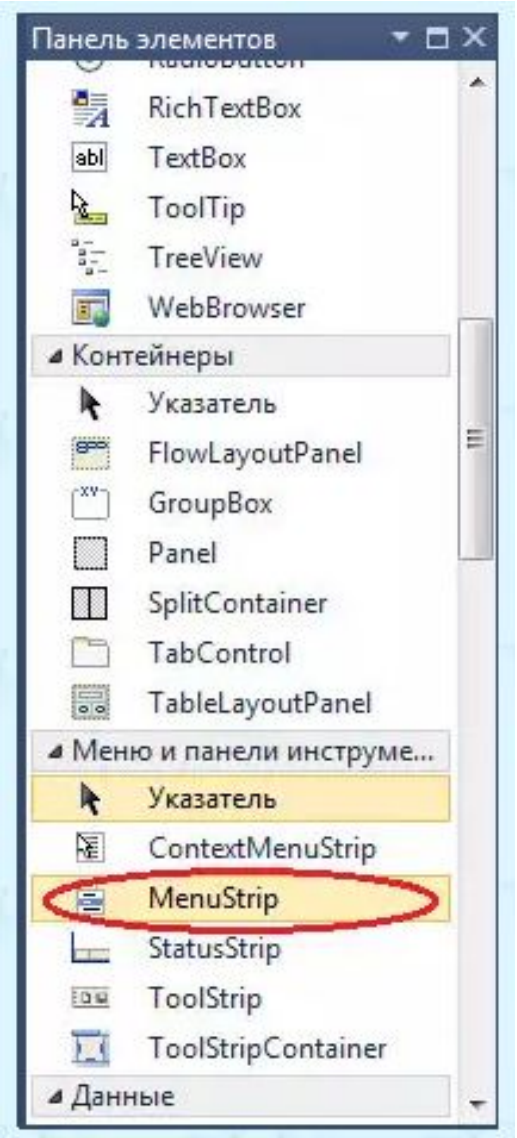
Свойства компонента ToolStrip

Свойство	Описание
Buttons	**Items** - коллекция компонентов, находящихся в панели инструментов
Dock	Граница родительского компонента (формы), к которой "привязана" панель инструментов: к верхней (**Top**), левой (**Left**), нижней (**Bottom**) или правой (**Right**)
Visible	Свойство позволяет скрыть (**False**) или сделать видимой (**True**) панель инструментов
Enabled	Свойства управляет доступом к компонентам панели инструментов. Если значение свойства равно **False**, то все компоненты недоступны

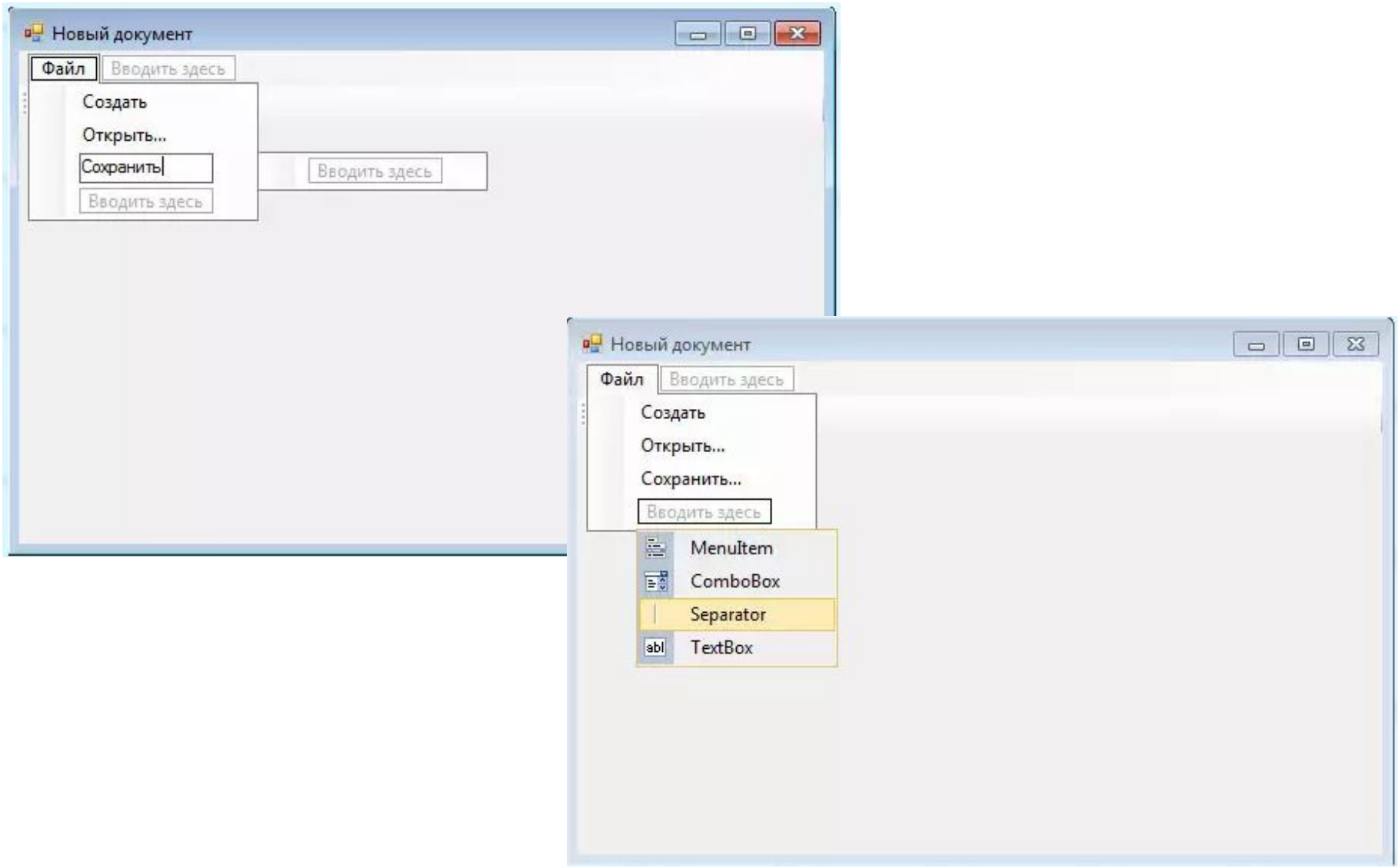
Компонент ToolStrip



Компонент MenuStrip



Компонент MenuStrip



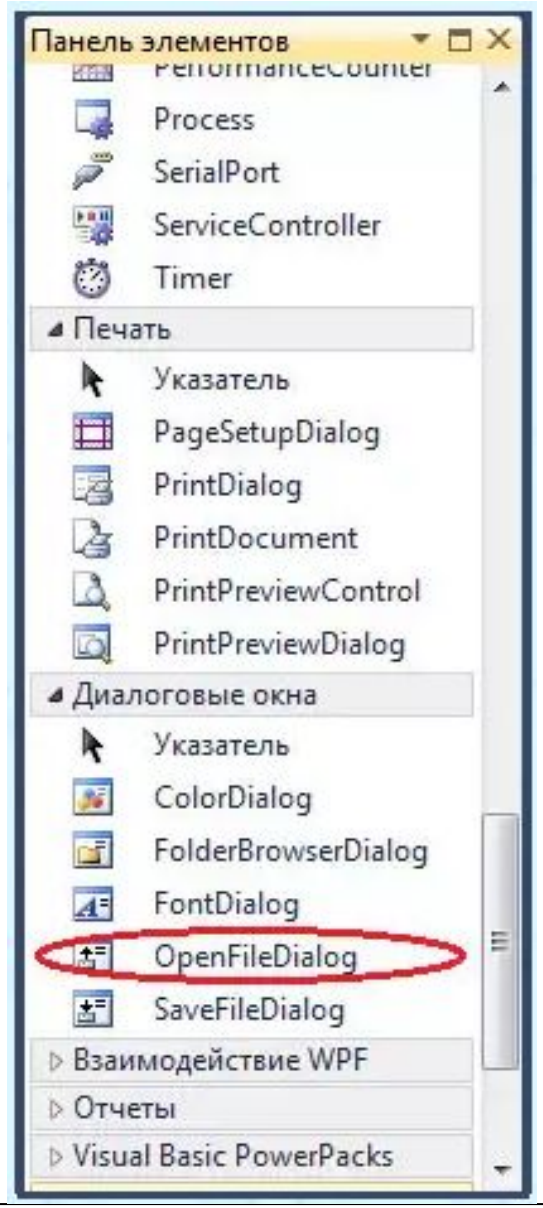
Свойства объекта MenuStrip

Свойство	Описание
Text	Название элемента меню
Image	Картинка, которая отображается рядом с командой
Enabled	Признак доступности элемента меню. Если значение свойства равно **False**, то элемент меню недоступен (в результате щелчка на элементе меню событие **Click** не происходит, название элемента меню отображается инверсным, по отношению к доступному пункту меню, цветом)
Checked	Признак того, что элемент меню выбран. Если значение свойства равно **True**, то элемент помечается галочкой. Свойство **Checked** обычно применяется для тех элементов меню, которые используются для отображения параметров, например, пункт меню ***Отображать строку состояния***
ShortcutKeys	Свойство определяет комбинацию клавиш, нажатие которой активизирует выполнение команды

Свойства объекта MenuStrip

Свойство	Значение
fileToolStripMenuItem.Text	Файл
toolStripMenuItem1.Text	Создать
toolStripMenuItem1.Image	
toolStripMenuItem2.Text	Открыть
toolStripMenuItem2.Image	
toolStripMenuItem3.Text	Сохранить
toolStripMenuItem3.Image	
toolStripMenuItem4.Text	Печать
toolStripMenuItem4.Image	
toolStripMenuItem5.Text	Выход
toolStripMenuItem5.ShortutKeys	Alt+F4
paramToolStripMenuItem.Text	Параметры
toolStripMenuItem6.Text	Панель инструментов
toolStripMenuItem6.Checked	True
toolStripMenuItem7.Text	Шрифт
helpToolStripMenuItem.Text	Справка
toolStripMenuItem8.Text	Справочная информация
toolStripMenuItem9.Text	О программе

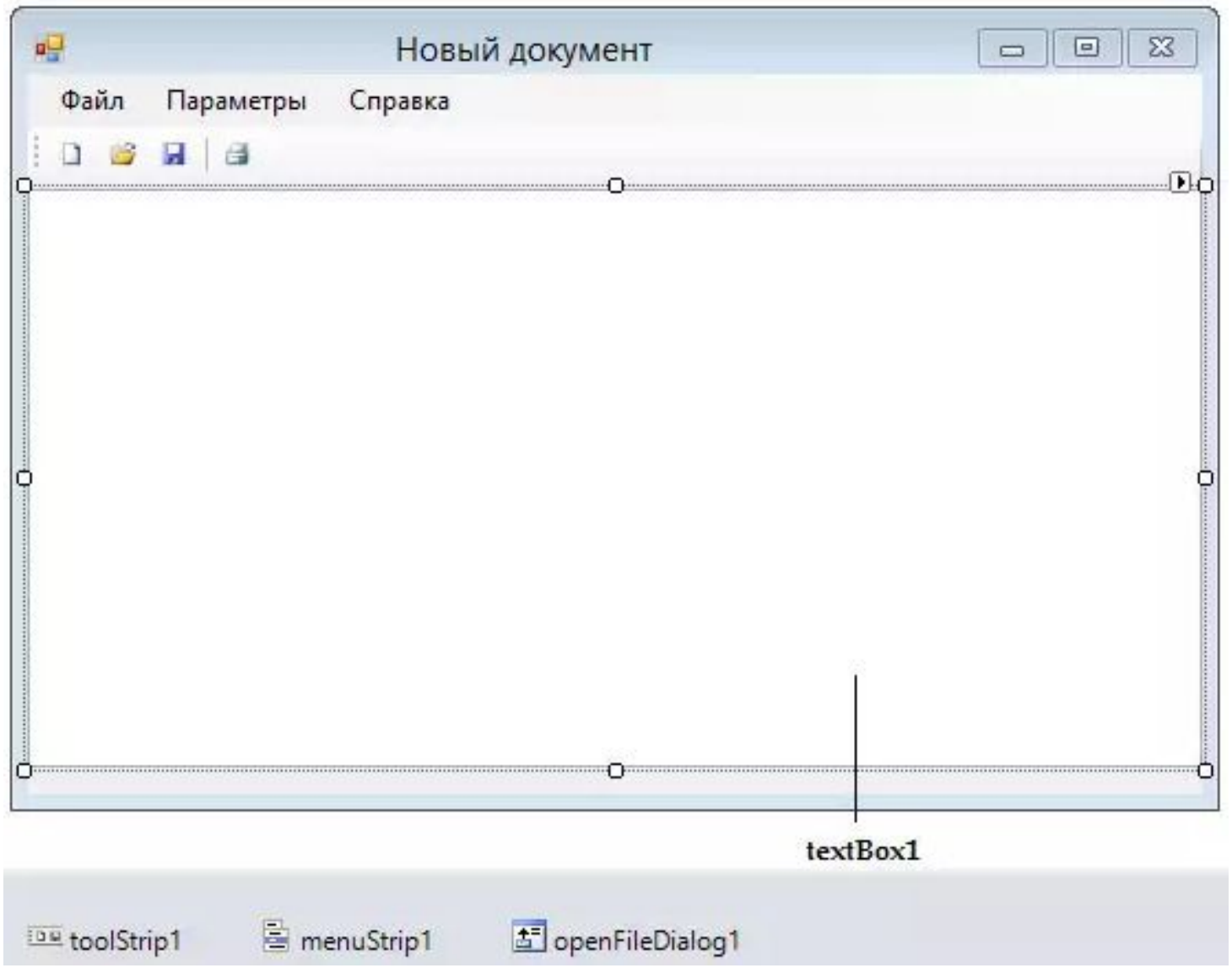
Компонент OpenFileDialog



Свойства компонента OpenFileDialog

Свойство	Описание
Title	Текст в заголовке окна. Если значение свойства не задано, то в заголовке отображается текст ***Открыть***
InitialDirectory	Каталог, содержимое которого отображается при появлении диалога на экране
Filter	Свойство задает один или несколько фильтров файлов. В окне отображаются только те файлы, имена которых соответствуют выбранному фильтру. Фильтр задается строкой вида ***Описание|Маска***. Например, фильтр ***Текст|*.txt*** задает, что в окне диалога следует отображать только текстовые файлы. Фильтр может состоять из нескольких элементов, например: ***Текст|*.txt|Все файлы|*.****
FilterIndex	Если фильтр состоит из нескольких элементов, например, ***Текст|*.txt|Все файлы|*.****, то значение свойства задает фильтр, который будет использоваться в момент появления диалога на экране
FileName	Имя выбранного пользователем файла
RestoreDirectory	Признак необходимости отображать содержимое каталога, указанного в свойстве **InitialDirectory**, при каждом появлении окна. Если значение свойства равно **False**, то при следующем появлении окна отображается содержимое каталога, выбранного пользователем в предыдущий раз

Пример


Новый документ
Файл
Параметры
Справка
textBox1
toolStrip1
menuStrip1
openFileDialog1

Пример

- **private:**
- **String^ fn; // имя файла**
- **bool textChanged; // true - в текст внесены изменения**

Пример

```
// выбор в меню Файл команды Открыть
private: System::Void toolStripMenuItem2_Click
   (System::Object^ sender, System::EventArgs^ e)
{
    System::Windows::Forms::DialogResult dr;
     int r;
```

Пример

```
r = 0;  //SaveText();
// сохранить текст, находящийся в поле компонента
// функция, которая будет использована в дальнейшем для сохранения предыдущего содержимого компонента
```

Пример

```
if (r == 0) // если нечего сохранять...
{
openFileDialog1->FileName = String::Empty;
// отобразить диалог Открыть
dr = openFileDialog1->ShowDialog();
```

Пример

```
if (dr == System::Windows::Forms::DialogResult::OK)
{
fn = openFileDialog1->FileName;
// отобразить имя файла в заголовке окна
this->Text = fn;
```

Пример

```
try
{
// считываем данные из файла
   System::IO::StreamReader^ sr = gcnew
   System::IO::StreamReader(fn);
   textBox1->Text = sr->ReadToEnd();
```

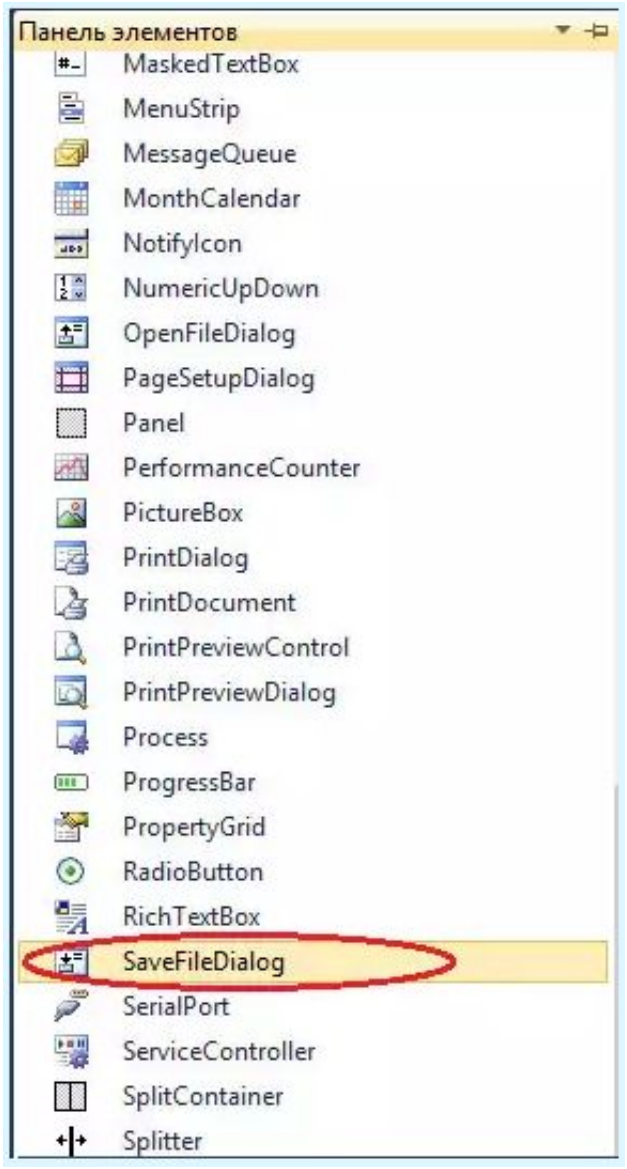
Пример

```
textBox1->SelectionStart = textBox1->TextLength;
sr->Close();
textChanged = false;
}
```

Пример

```
catch ( System::IO::FileLoadException^ e)
{
    MessageBox::Show("Ошибка:\n" + e->Message,
   "MEdit", MessageBoxButtons::OK,
   MessageBoxIcon::Error);
}
                    }
            }
}
```

Компонент SaveFileDialog



Свойства компонента SaveFileDialog

Свойство	Описание
Title	Текст в заголовке окна. Если значение свойства не задано, то в заголовке отображается текст ***Сохранить как***
FileNane	Полное имя файла, которое задал пользователь. В общем случае оно образуется из имени каталога, содержимое которого отображается в диалоговом окне, имени файла, которое пользователь ввел в поле ***Имя файла***, и расширения, заданного значением свойства **DefaultExt**
DefaultExt	Расширение файла по умолчанию. Если пользователь в поле ***Имя файла*** не укажет расширение, то к имени файла будет добавлено расширение (при условии, что значение свойства **AddExtension** равно **True**), заданное значением этого свойства
InitialDirectory	Каталог, содержимое которого отображается при появлении диалога на экране
RestoreDirectory	Признак необходимости отображать содержимое каталога, указанного в свойстве **InitialDirectory**, при каждом появлении окна. Если значение свойства равно **False**, то при следующем появлении окна отображается содержимое каталога, выбранного пользователем в предыдущий раз

Свойства компонента SaveFileDialog

CheckPathExists	Признак необходимости проверки существования каталога, в котором следует сохранить файл. Если указанного каталога нет, то выводится информационное сообщение
CheckFileExists	Признак необходимости проверки существования файла с заданным именем. Если значение свойства равно **True** и файл с указанным именем уже существует, появляется окно запроса, в котором пользователь может подтвердить необходимость замены (перезаписи) существующего файла
Filter	Свойство задает один или несколько фильтров файлов. В окне отображаются только те файлы, имена которых соответствуют выбранному фильтру. Фильтр задается строкой вида ***Описание|Маска***. Например, фильтр ***Текст|*.txt*** задает, что в окне диалога следует отображать только текстовые файлы. Фильтр может состоять из нескольких элементов, например: ***Текст|*.txt|Все файлы|*.****
FilterIndex	Если фильтр состоит из нескольких элементов, например, ***Текст|*.txt|Все файлы|*.****, то значение свойства задает фильтр, который будет использоваться в момент появления диалога на экране

Идентификаторы кнопок

- **dr = saveFileDialog1->ShowDialog();**
- **// отобразить диалог Сохранить**

- **Функции (методы) манипулирования каталогами и файлами принадлежат пространству имен System::IO.**
- **di — объект типа DirectoryInfo,**
- **fi — объект типа FileInfo,**
- **sr — объект типа StreamReader,**
- **sw — объект типа StreamWriter**

Функция (метод)	Результат (значение)
DirectoryInfo (Path)	Создает объект типа DirectoryInfo, соответствующий каталогу, заданному параметром Path
di->GetFiles(fn)	Формирует список файлов каталога — массив объектов типа FileInfo
	Каждый элемент массива соответствует файлу каталога, заданного объектом di (тип DirectoryInfo). Параметр fn задает критерий отбора файлов
di->Exists	Проверяет, существует ли в системе каталог. Если каталог существует, то значение функции равно true, в противном случае — false
di->Create(dn)	Создает каталог. Если путь к новому каталогу указан неправильно, возникает исключение
fi->CopyTo(Path)	Копирует файл, заданный объектом fi (тип FileInfo), в каталог и под именем, заданным параметром Path. Значение метода — объект типа FileInfo, соответствующий файлу, созданному в результате копирования

fi->MoveTo(Path)	Перемещает файл, заданный объектом fi (тип FileInfo), в каталог и под именем, заданным параметром Path
fi->Delete()	Уничтожает файл, соответствующий объекту fi (тип FileInfo)
StreamReader(fn)	Создает и открывает для чтения поток, соответствующий файлу, заданному параметром fn. Значение метода — объект типа StremReader. Поток открывается для чтения в формате UTF-8
StreamReader (fn, encd)	Создает и открывает для чтения поток, соответствующий файлу, заданному параметром fn. Значение метода — объект типа StremReader
	Поток открывается для чтения в кодировке, заданной параметром encd (объект типа System::Text::Encoding). Для чтения текста в кодировке Windows 1251 параметр encd необходимо инициализировать значением System::Text::Encoding:: GetEncoding(1251)
sr->Read()	Читает символ из потока sr (объект типа StremReader). Значение метода — код символа
sr->Read(buf, p, n)	Читает из потока sr (объект типа StremReader) символы и записывает их в массив символов buf (тип Char). Параметр p задает номер элемента массива, в который будет помещен первый прочитанный символ, параметр n — количество символов,

sr->ReadToEnd()	Читает весь текст из потока sr (объект типа StremReader). Значение метода — прочитанный текст
sr->ReadLine()	Читает строку из потока sr (объект типа StremReader). Значение метода — прочитанная строка
StreamWriter(fn)	Создает и открывает для записи поток, соответствующий файлу, заданному параметром fn. Значение метода — объект типа StremWriter. Поток открывается для записи в формате (кодировке) UTF-8
StreamWriter (fn,a,encd)	Создает и открывает для записи поток, соответствующий файлу, заданному параметром fn. Значение метода — объект типа StremWriter

sw->Write(v)	Записывает в поток sw (объект типа StreamWriter) строку символов, соответствующую значению v. В качестве v можно использовать выражение строкового или числового типа
sw->WriteLine(v)	Записывает в поток sw (объект типа StreamWriter) строку символов, соответствующую значению v, и символ "новая строка". В качестве v можно использовать выражение строкового или числового типа
sw->WriteLine	Записывает в поток sw (объект типа StreamWriter) символ "новая строка"
s->Close()	Закрывает поток s