Оглавление

[Основы компоновки Windows Forms 1](#_Toc79414894)

[Задание 1 4](#_Toc79414895)

[Дополнительное задание 1 4](#_Toc79414896)

[Разработка графического редактора. 5](#_Toc79414897)

[Компоновка формы 5](#_Toc79414898)

[Описание классов, кодирование 7](#_Toc79414899)

[Задание 2 13](#_Toc79414900)

[Дополнительное задание 2 13](#_Toc79414901)

[Разработка графического редактора 2. 13](#_Toc79414902)

[Настройка PictureBox'а 13](#_Toc79414903)

[Выбор цвета, добавление пиктограмм на кнопки выбора кистей. 16](#_Toc79414904)

[Задание 3 18](#_Toc79414905)

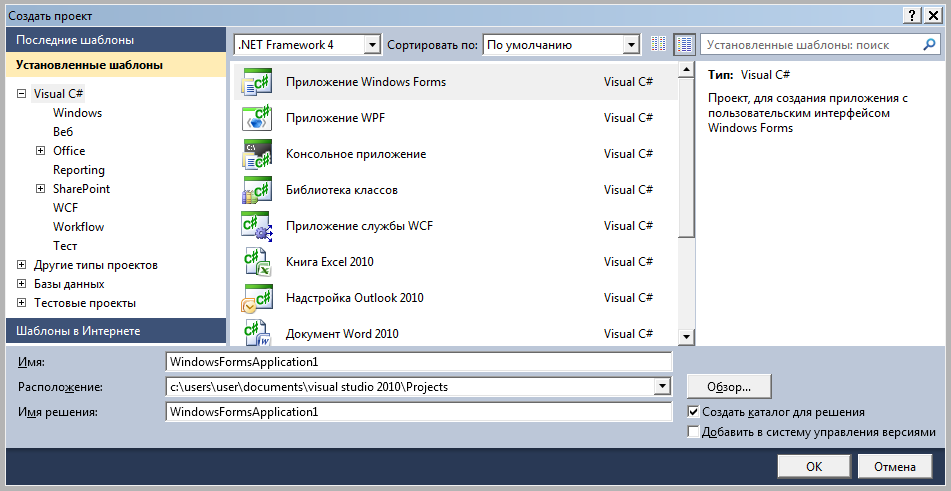
[Дополнительное задание 3 18](#_Toc79414906)

[Дополнительные материалы 18](#_Toc79414907)

# Основы компоновки Windows Forms

***Данный раздел предназначен для тех, кто не имеет опыта работы с Windows Forms, если это не так, можете сразу переходить к следующему.***

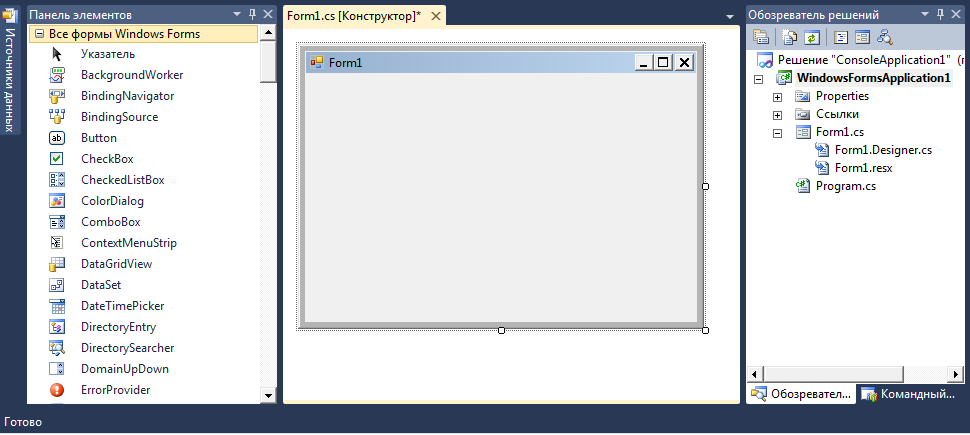
Для создания проекта выберите пункт меню Файл->Создать->Проект. Откроется окно, показанное ниже:



*Рисунок 1 – Выбор типа создаваемого проекта в Visual Studio*

Можно заметить, что в левой части диалогового окна показано дерево, группирующее шаблоны по языкам программирования и технологиям, также в правом левом углу есть поле поиска. Правая панель в этом диалоговом окне содержит дополнительную информацию о выбранном шаблоне проекта. Кроме того, пользователь может выбрать версию платформы .NET. Версия фреймворка для этого приложения выбирается с помощью выпадающего меню в верхней части диалогового окна.

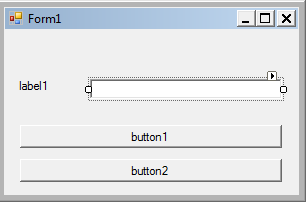
Создадим Приложение Windows Forms, после чего нам откроется визуальный редактор, показанный на рисунке 2:



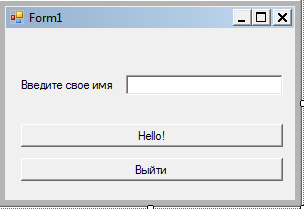
*Рисунок 2 – Вид окна по умолчанию для созданного проекта Windows Forms.*

В левой части окна расположена панель элементов (если такой панели нет, ее следует включить в пункте меню Вид ->Панель элементов), представляющая базовый набор элементов управления.

Создадим простое приложение, приветствующее пользователя, для этого перетащим элементы TextBox, Label, Button на форму, чтобы она приняла следующий вид:



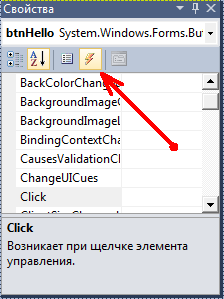
Настроим созданные компоненты через окно свойств (по умолчанию располагается в правой части экрана, если его нет, нужно включить через меню вид). Путем задания свойства Text изменим отображаемый текст на компонентах:



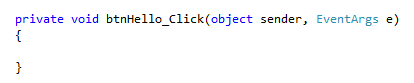
Это же свойство отвечает за заголовок окна (по умолчанию Form1), измените его на любой другой текст.

Дополнительно рекомендуется изменить значения свойства Name таким образом, чтобы оно отображало смысловое назначение элемента, а так же его тип, например для текстового поля с именем можно дать имя tbName (сокращение от Text Box Name), либо textName, для кнопок btnHello и т.п.

Кроме свойств, на панели свойств можно настраивать реакцию элементов на некоторые предопределенные события:



Определим реакцию кнопок на клик мышки, для этого создадим обработчики события Click. Для этого достаточно сделать двойной клик по событию Click в панели свойств, и Visual Studio сама сгенерирует пустой обработчик этого события в классе формы:



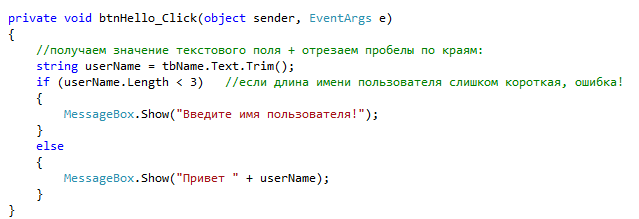
Создайте обработчики события Click для обоих кнопок.

В теле функции-обработчика нажатия для кнопки Выход определим следующий код:

Application.Exit();

Application является статическим классом, который предоставляет методы и свойства  для управления приложением, например методы для запуска и остановки приложения, для обработки сообщений Windows и свойства для получения сведений о приложении. Этот класс не наследуется.

В качестве обработчика нажатия кнопки Hello используйте следующий код:



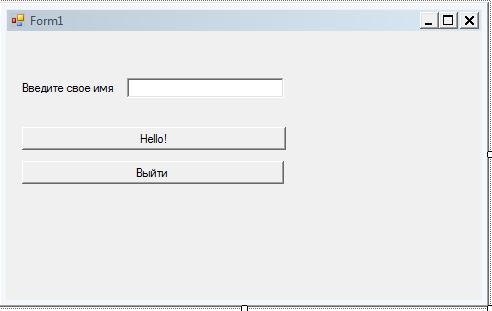
Сохраните проект, запустите, проверьте его работу.

## Задание 1

Выполните действия, описанные в разделе [«Основы компоновки Windows Forms»](#_Создание_оконного_приложения)

## Дополнительное задание 1

При растягивании формы элементы управления остаются на месте, из-за чего окно принимает вид, как на рисунке, показанном ниже:



Требуется исправить это поведение, для этого нужно либо запретить изменение размеров формы, либо «приклеить» границы компонентов к границам формы, либо использовать панели для компоновки элементов, например TableLayoutPanel или аналогичные (на вкладке Контейнеры в панели инструментов). Реализуйте любой из этих способов.

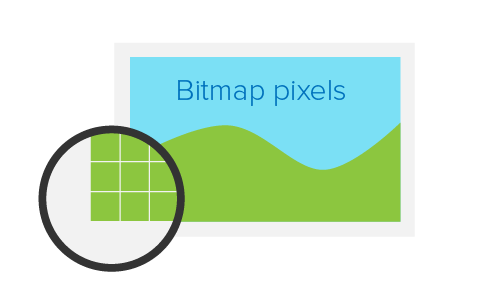
Для решения рассмотрите следующие свойства:

* у формы – FormBorderStyle, MaximizeBox, MinimizeBox, MaximumSize, MinimumSize, SizeGripStyle;
* для остальных элементов управления: Anchor, Dock, MaximumSize, MinimumSize.

# Разработка графического редактора.

Целью этой данной лабораторной работы будет создание упрощенного графического редактора. При выполнении заданий этого раздела рекомендуется использовать материал лекций, а также статьи из раздела «Дополнительные материалы»

Существуют разные способы кодирования графической информации, мы остановимся на самом простом – Bitmap, представляющий собой двумерный массив точек, цвет каждой из которых задается независимо.



Пример Bitmap изображения

## Компоновка формы

Предлагаемый здесь порядок размещения элементов имеет рекомендательный характер, однако, в случае выбора другого дизайна, функционал должен сохраниться.

Для создаваемого редактора можно выделить следующие области на форме:

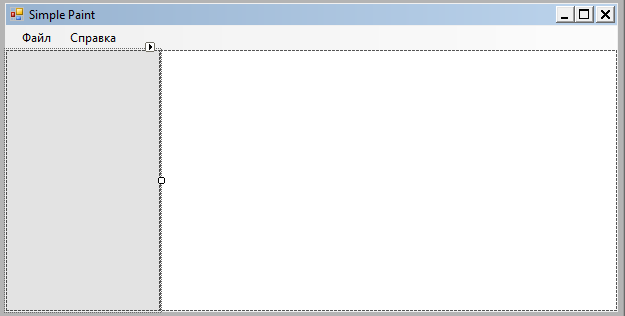
* Главное меню (Пункты меню «Файл», «Правка», «Вид», «Справка» и т.п.) – располагается вверху рабочей области.
* Область инструментов – содержит набор кистей, цветовую палитру и другие инструменты для работы с графикой. Может располагаться в любом месте, например слева рабочей области.
* Область рисования – Занимает все оставшееся пространство.

Разместите их на форме, последовательно размещая элементы MenuStrip, и 2 элемента Panel из панели элементов.

Добавьте пункты меню, которые на ваш взгляд должен иметь графический редактор (Обязательно: Файл -> Создать, Сохранить, Открыть, Выход; Справка –> О программе)

Задайте одной из панелей имя toolsPanel (англ. - панель инструментов) через свойство Name, а также установите свойство Dock = Left, это прикрепит панель к левой половине рабочей области.

Для другой панели задайте имя drawPanel (англ. – панель рисования), установите свойство Dock = Fill. В результате форма должна приобрести примерно следующий вид:

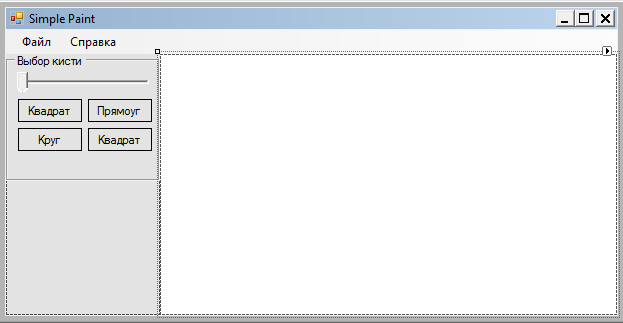


На панель toolsPanel перетащите элемент GroupBox { Dock = Top, Text = Выбор кисти}.

Для рисования важен размер кисти, а так же ее форма. Для задания размера используем компонент TrackBar { Dock = Top, Autosize= false}, для которого в дальнейшем нужно экспериментальным путем определить свойства Minimum (должно быть больше 0), Maximum, LargeChange, SmallChange, TickChange, TickStyle, а так же высоту компонента. Рекомендуется изменить имя компонента со стандартного trackBar1 на осмысленное.

Для осуществления выбора кисти добавим обычные кнопки Button, зададим свойство Text в соответствии типом добавляемых кистей. Можете поэкспериментировать со стилем отображения FlatStyle, размерами.

После проведенных манипуляций, форма должна приобрести следующий вид:



В следующих лабораторных работах текстовое описание кистей будет заменено графическими пиктограммами, а размещение возможных средств рисования станет автоматическим.

Последним шагом на данном этапе станет размещение элемента PictureBox {Dock = Fill} на панели рисования. Этот компонент служит хостом для изображений разных форматов, в том числе интересующего нас BMP.

Теперь перейдем к «оживлению» интерфейса.

## Описание классов, кодирование

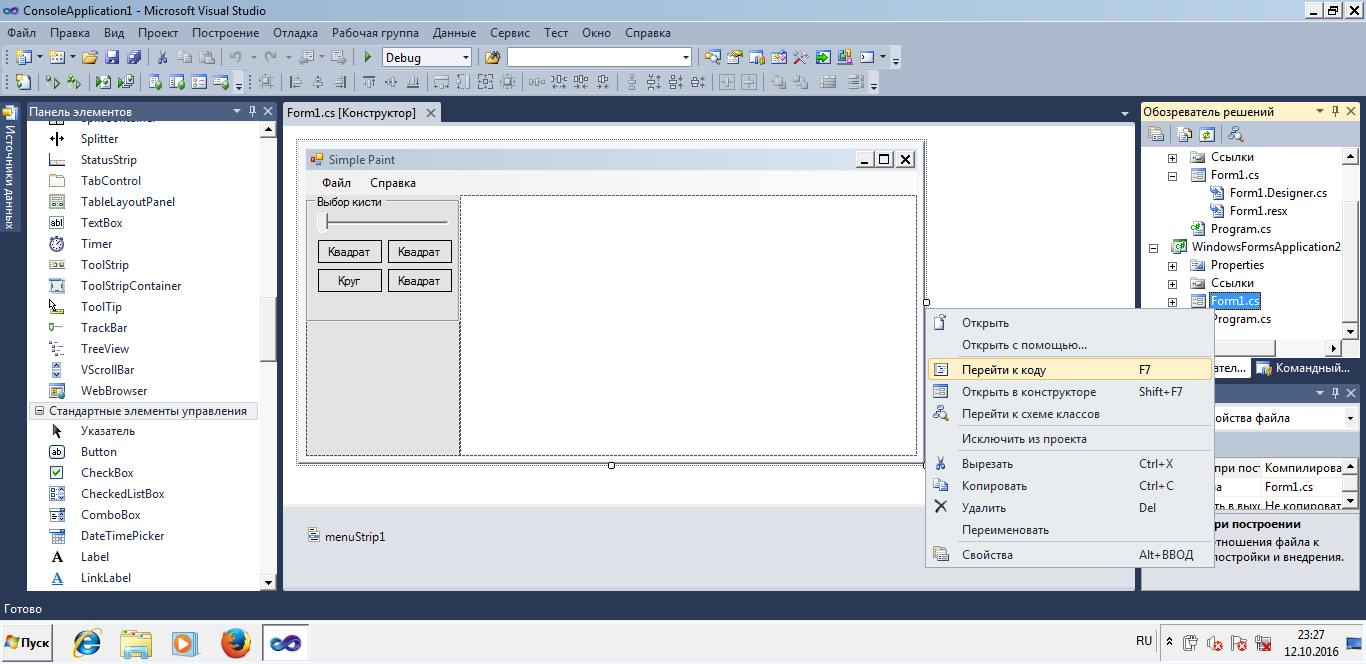
Перед кодированием, рассмотрим общий алгоритм работы программы:

1. При запуске программы необходимо инициализировать область рисования, для этого выполнить ее заливку каким-либо цветом, цвет заливки можно определить как свойство формы. Эти же действия необходимо производить при создании нового файла. назовем его \_defaultColor. В языке C# и в целом в платформе .Net для хранения цвета определен тип данных Color.
2. Рисование. Последовательность действий для рисования определяется следующим образом: Выбор кисти, клик мышкой по поверхности рисования (должен быть отрисован примитив, реализуемый кистью), далее, до тех пор, пока мышь зажата, необходимо на каждое перемещение мышки по поверхности рисовать соответствующую фигуру. При отпускании кнопки мыши рисование прекращается.

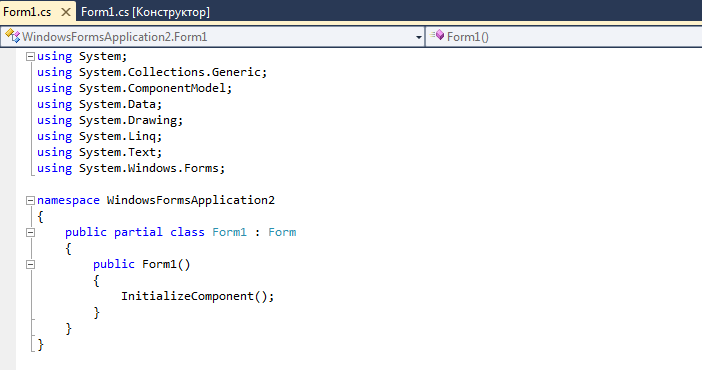
Параметрами рисования здесь являются Кисть, Цвет, Размер.

Теперь перейдем к реализации описанного алгоритма.

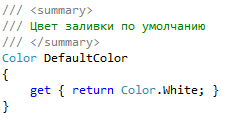
Для начала откроем исходный код нашей формы – файл Form1.cs. Сделать это можно или нажав клавишу F7, если вы находитесь в режиме дизайна, или через контекстное меню в Обозревателе решений:



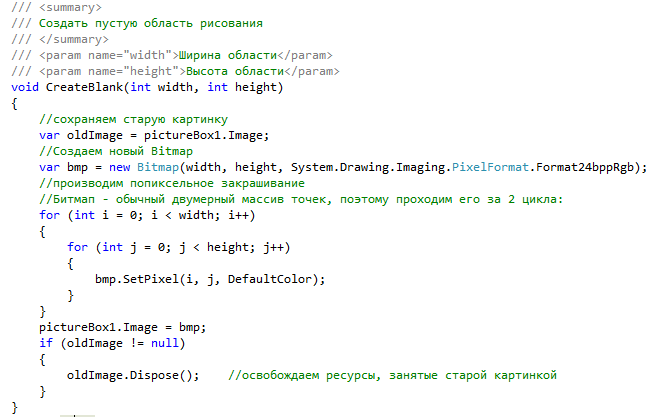
В результате вы должны увидеть примерно следующее:



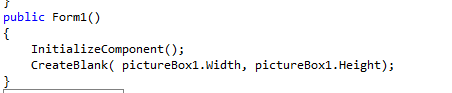
Для создания области рисования добавим необходимо определить цвет заливки. Удобнее сделать это, определив, свойство, отвечающее за цвет по умолчанию, для этого добавьте описание свойства DefaultColor:



Теперь перейдем непосредственно к инициализации области рисования. Т.к. этот функционал может быть задействован в нескольких местах (при загрузке формы, через пункты меню), лучше вынести данный код в отдельный метод. Действия здесь сводятся к созданию объекта Bitmap, его попиксельному закрашиванию, удалению старого объекта, если он был. Это можно получить с использованием следующего кода:

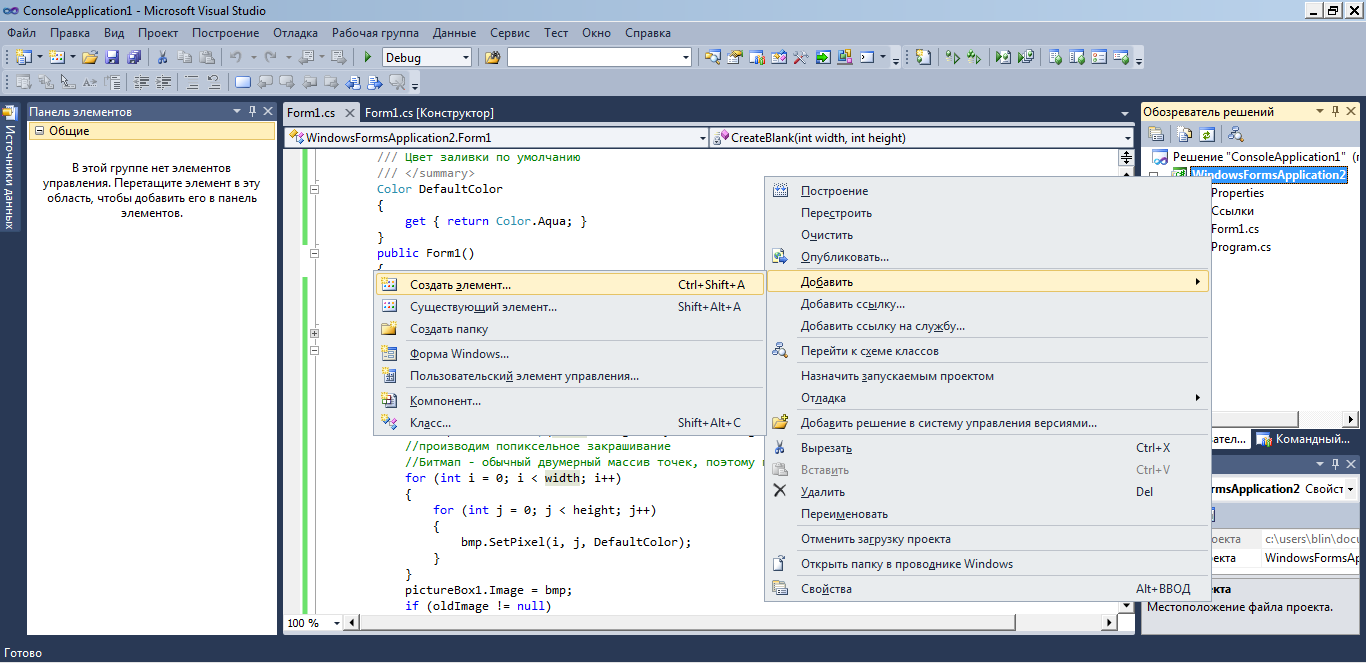


Чтобы заливка произошла сразу при запуске программы, вызовем нашу функцию в конструкторе Form1, сразу после InitializeComponent();

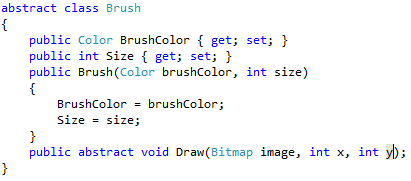


Запустите программу, попробуйте изменить значение свойства DefaultColor, к каким результатам это приведет?

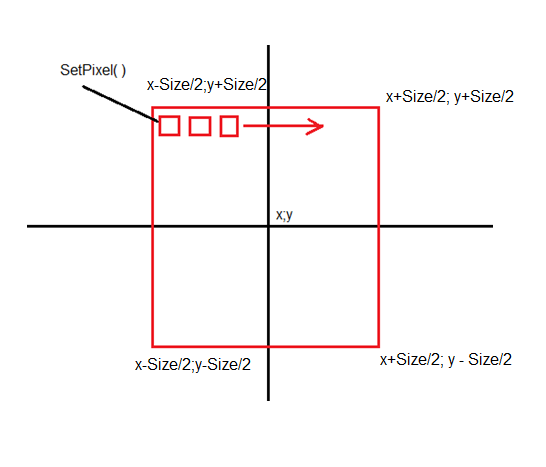
Перейдем непосредственно к рисованию. Для этого необходимо определить абстрактный класс «Кисть», а также его конкретные реализации. Чтобы добавить новый класс в проект, необходимо кликнуть правой кнопкой мыши по проекту -> Добавить -> Создать элемент.



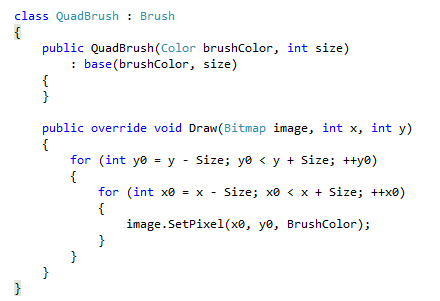
Выбрать тип элемента Class, дать ему имя Brush (кисть). Поскольку само понятие кисть не дает сведений о его поведении, сделаем класс абстрактным. На данном этапе для нас имеет значение только один метод – нарисовать, назовем его Draw, в качестве параметра передадим изображение, на котором будет производиться рисование. Кроме того, кисть обладает еще двумя характеристиками – размером и цветом, отобразим их в виде свойств:



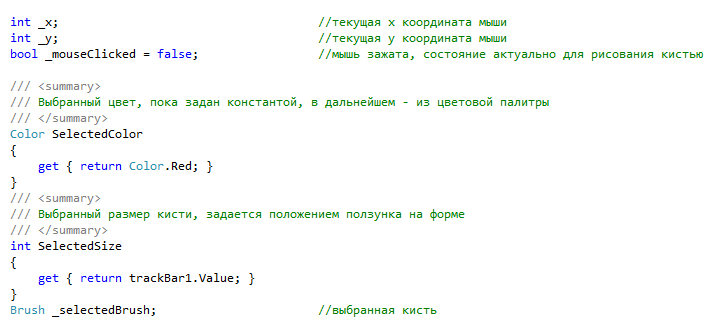
Создадим на базе этого класса кисть, рисующую квадрат. Будем считать, что параметры x,y функции Draw отвечают за центр отображаемой геометрической фигуры, переменная – член класса Size – определяет длину стороны квадрата. Тогда, для того чтобы нарисовать квадрат, нужно попиксельно закрасить каждую строку Bitmap изображения в заданных пределах:



Чтобы реализовать это поведение, создадим новый класс, и дадим ему имя QuadBrush:



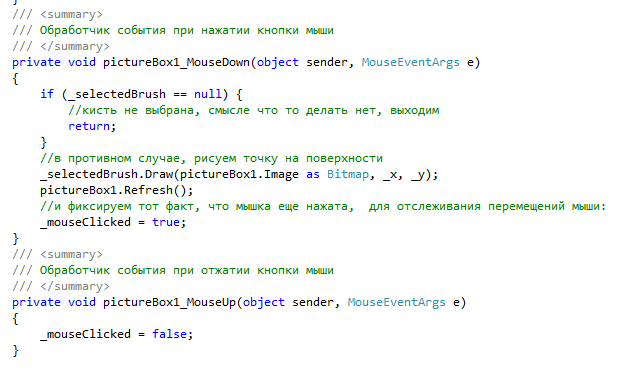
Для того чтобы использовать данную кисть, необходимо реализовать поведение мыши и выбор кисти для рисования. Возвращаемся в Form1.cs и добавляем в класс Form1 следующие переменные и свойства:

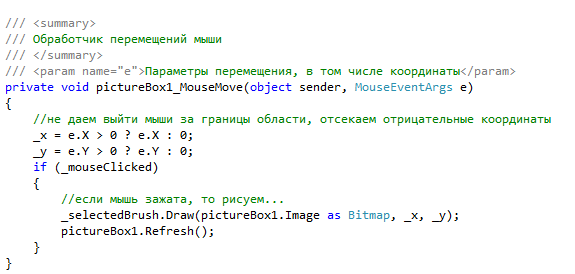


Осталось добавить обработчики нажатий кнопок, начнем с выбора кисти, для этого выберем кнопку, отвечающую за квадратную кисть и добавим обработчик события Click. В теле метода пропишем:

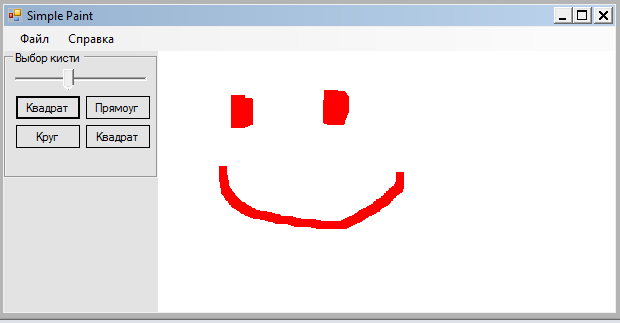


Описываем обработчики событий MouseUp, MouseDown, MouseMove для pictureBox'а:





Запускаем, проверяем работу:



## Задание 2

1. Выполните действия, описанные в разделе «Разработка графического редактора»
2. Напишите код для пунктов меню «Создать», «Выход»
3. Добавить минимум 3 собственных реализаций формы кисти. Примеры фигур приведены ниже.

|  |  |
| --- | --- |
| Круг | Для рисования можно использовать уравнение окружности:  R2 = x2 + y2 |
| Окружность |
| Косая черта |  |
| Снежинка |  |
| Распылитель  (баллончик из Paint'а) | Для получения эффекта баллончика можно использовать набор точек, координаты которых случайно сгенерировано. |
| Кисть, реализующая эффект прозрачности, т.е. результирующий цвет = функция от исходного цвета и цвета кисти, форма кисти при этом – произвольная. |  |
| Фигуры, заданные в полярных координатах (звезда, кардиоида, трех-четырехлистник и т.п.)  Примеры функций вращения:  p = a + b \* fi  p = a sin (2fi / 3)  p = a sin ( 3 fi )  p = 3 sqrt (cos(7 fi) )  p = 2 sin (2 fi )  p = 3 sqrt (cos(3 fi) )  p = 2a ( 1 + cos (fi) )  p = l / (1 - e cos fi ) | Подробнее здесь <https://habr.com/ru/post/519954/> и в Яндексе по запросу «фигуры в полярных координатах» |

## Дополнительное задание 2

1. В текущей реализации присутствует ряд ошибок, например, текущий алгоритм не обрабатывает выход за пределы области рисования, что вызывает ошибку. Исследуйте приложение на наличие других ошибок, исправьте их.
2. В текущей реализации размер кисти фиксируется при создании объекта, который в свою очередь происходит при выборе кисти. Движение ползунка после этого ни на что не влияет. Исправьте.
3. Добавьте реализацию ластика – стирательной резинки. (по сути, обычная кисть, которая вместо рисования производит закрашивание цветом по умолчанию).

# Разработка графического редактора 2.

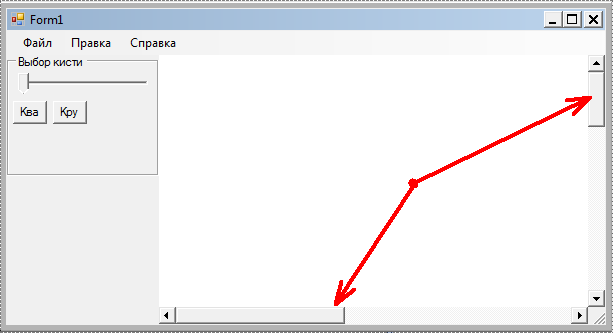
## Настройка PictureBox'а

В текущей версии область рисования ограничена в размерах, независимо от фактического размера изображения. Изменим это поведение и добавим возможность ручного задания размера полотна.

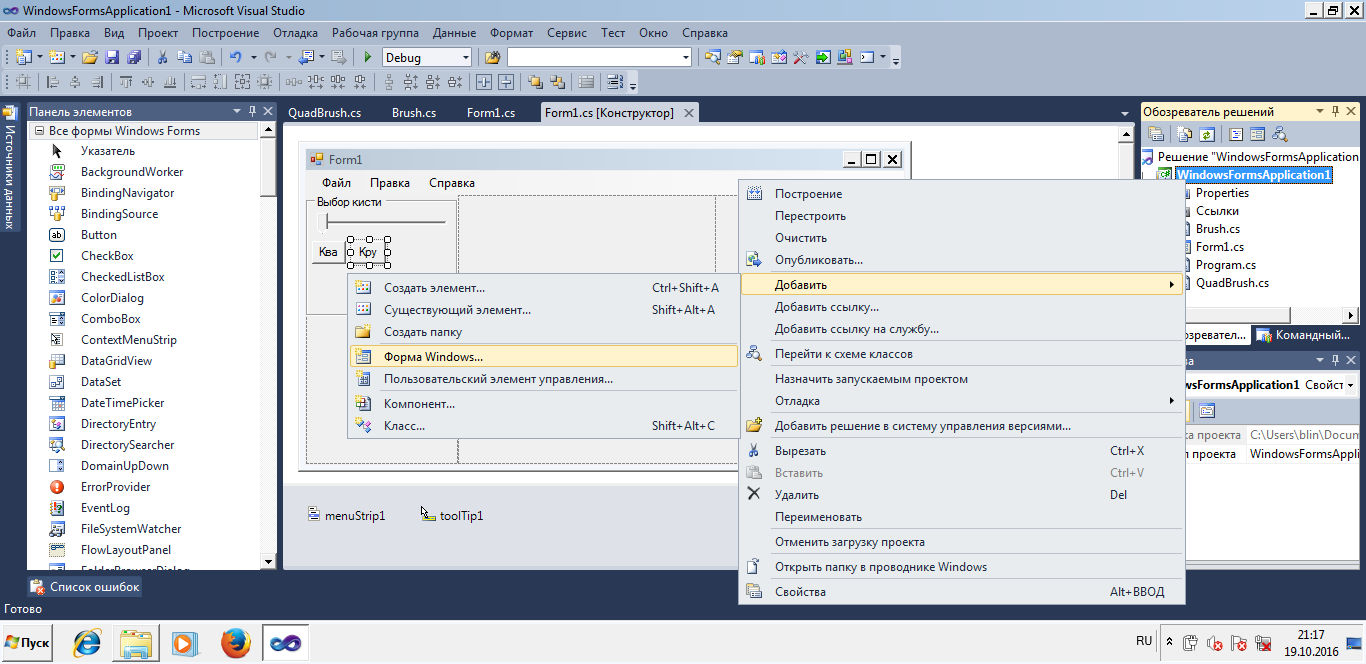
Для начала, рассмотрим поведение компонента PictureBox, на котором происходит рисование. Сейчас для удобства работы его границы приклеены к границам приложения (свойство Dock = Fill), и при изменении размеров формы также изменяется размер полотна для рисования, т.е. его фактический размер ограничен размером формы. Однако, в случае если загружаемое на него изображение будет слишком большим, то «лишняя» часть будет скрыта от редактирования. В реальности же размер изображений не ограничен. Чтобы убрать ограничение размера PictureBox'а изменим значение свойства Dock на None.

Далее, выберем компонент drawPanel (созданный в разделе Компоновка формы), установим его свойство AutoScroll = True. Теперь при расположении на панели элементов, выходящих за ее границы, будут автоматически добавляться полосы прокрутки.

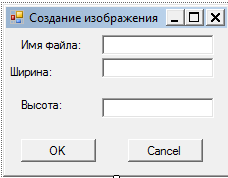
Размер полотна задается в конструкторе формы, при вызове метода CreateBlank. Чтобы проверить работу областей прокрутки, задайте в качестве параметров этой функции заведомо большие значения, например ширина = 2000, высота = 1000. Запустим приложение, если все сделано верно, должны появиться полосы прокрутки:



Реализуем возможность настройки размера изображения. Для этого, модифицируем пункт меню «Создать». При создании нового файла будем запрашивать размеры у пользователя, кроме того, добавим поле для ввода имени файла. Для этого добавим новую форму в проект:



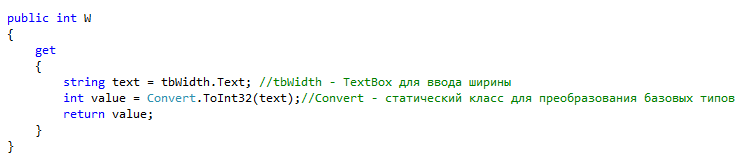
С учетом знаний, приобретенных в разделе «Основы компоновки Windows Forms», приведите форму к следующему виду:



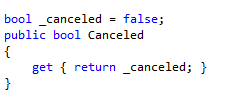
Чтобы получить доступ к введенным значениям извне существует 2 способа:

* предоставить прямой доступ к TextBox'ам и считать из них введенные значения - плохой способ, т.к. кроме введенных значений будет получен доступ ко всем остальным полям и свойствам TextBox'а.
* Создать методы или свойства, предоставляющие введенные значения – в этом случае пользователь получит только то, что ему нужно + при изменении способа ввода необходимо будет изменить только реализацию этих методов и свойств.

Воспользуемся вторым способом. Создадим 3 свойства: для имени файла, для ширины и высоты, назовем их FileName, W, H соответственно. Поместим определение свойства Width в класс создаваемой формы:

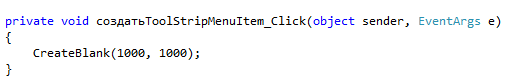
 Создайте по аналогии оставшиеся 2 свойства. Учитывайте, что для свойства FileName типом будет не int, а String.

Также нужно учитывать, что пользователь может нажать кнопку Cancel, этот выбор нужно запомнить. Для этого добавим еще одну переменную \_canceled, и свойство, предоставляющее доступ к ней:

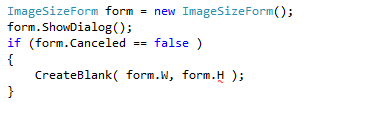


При нажатии обоих кнопок (OK, Cancel) должно происходить закрытие формы. Для закрытия формы используйте вызов функции Close(); Не забудьте изменить значение переменной \_canceled при нажатии кнопки «отмена» на true;

Вернемся к основной форме. Во-первых, добавим в класс формы переменную для хранения имени файла. Во-вторых, изменим код создания нового файла. Сейчас он должен выглядеть примерно так:

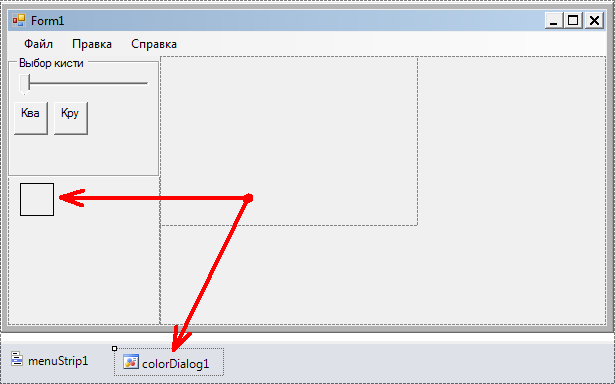


Изменим тело этого метода - используем только что созданную форму для получения размеров от пользователя:

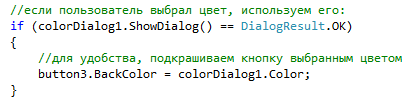


## Выбор цвета, добавление пиктограмм на кнопки выбора кистей.

В текущей реализации цвет зафиксирован в свойстве SelectedColor основной формы. Добавим возможность выбора. Для самой простой реализации нам понадобится кнопка, для выбора цвета и цветовая палитра, из которой этот выбор будет производиться. За цветовую палитру отвечает компонент ColorDialog. Разместим эти компоненты на форме:



В обработчике события Click только что созданной кнопки вызываем показ цветовой палитры:

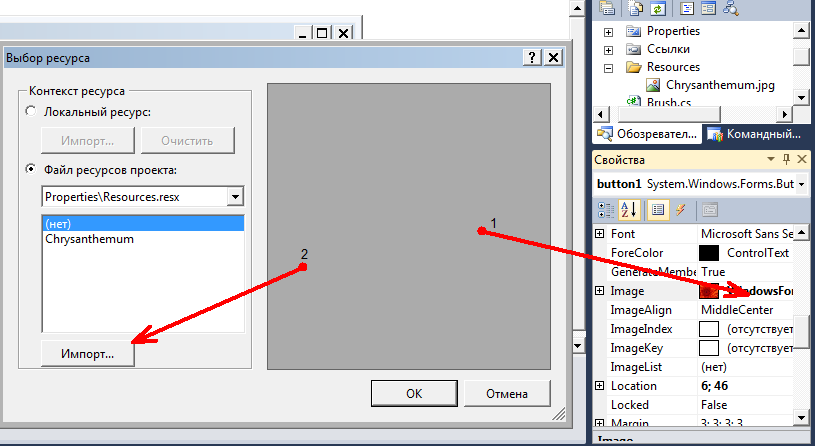


И изменяем реализацию свойства SelectedColor на:



Запустите приложение, проверьте работу выбора цвета, попробуйте нарисовать рисунок, используя несколько цветов.

Чтобы изменить внешний вид кнопок выбора кисти, можно заменить текст на изображение, для этого выберите кнопку, откройте панель свойств, найдите Image ->…->Import и загрузите подходящую иконку.

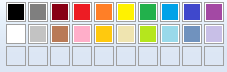


## Задание 3

1. Выполните действия, описанные в разделе «Разработка графического редактора 2»

## Дополнительное задание 3

1. Если посмотреть на пример формы для задания размеров изображения, видно, что отсутствует хоть какое-нибудь выравнивание (надпись «Ширина» уехала влево, разные вертикальные отступы). Для решения задачи типового размещения компонентов существуют так называемые панели компоновки, которые автоматически позиционируют размещенные на них элементы управления. В Visual Studio они представлены классами FlowLayoutPanel и TableLayoutPanel. Используйте их для автоматического позиционирования элементов на форме создания изображения. Сделайте то же самое для размещения кистей на основной форме.
2. Текущий способ выбора цвета не очень удобен, попробуйте разместить часть цветовой палитры на форме (дополнительным плюсом будет, если эти компоненты на форме будут генерироваться программно):



1. В текущей реализации цвет кисти фиксируется при создании объекта, который в свою очередь происходит при выборе кисти. Выбор цвета для уже созданной кисти ни на что не влияет. Исправьте.

# Дополнительные материалы

<http://professorweb.ru/my/csharp/charp_theory/level1/infocsharp.php>

<http://metanit.com/sharp/>

<http://cybern.ru/category/csharp>

<http://www.programmer-lib.ru/csharp.php>

<http://mycsharp.ru/post/4/2013_04_14_chto_takoe_programmirovanie_eto_slozhno_chto_nuzhno_znat_.html>

<https://www.tutorialspoint.com/csharp/index.htm>