Среда Visual Studio, C#

Методичка по выполнению лабораторных работ по направлению:

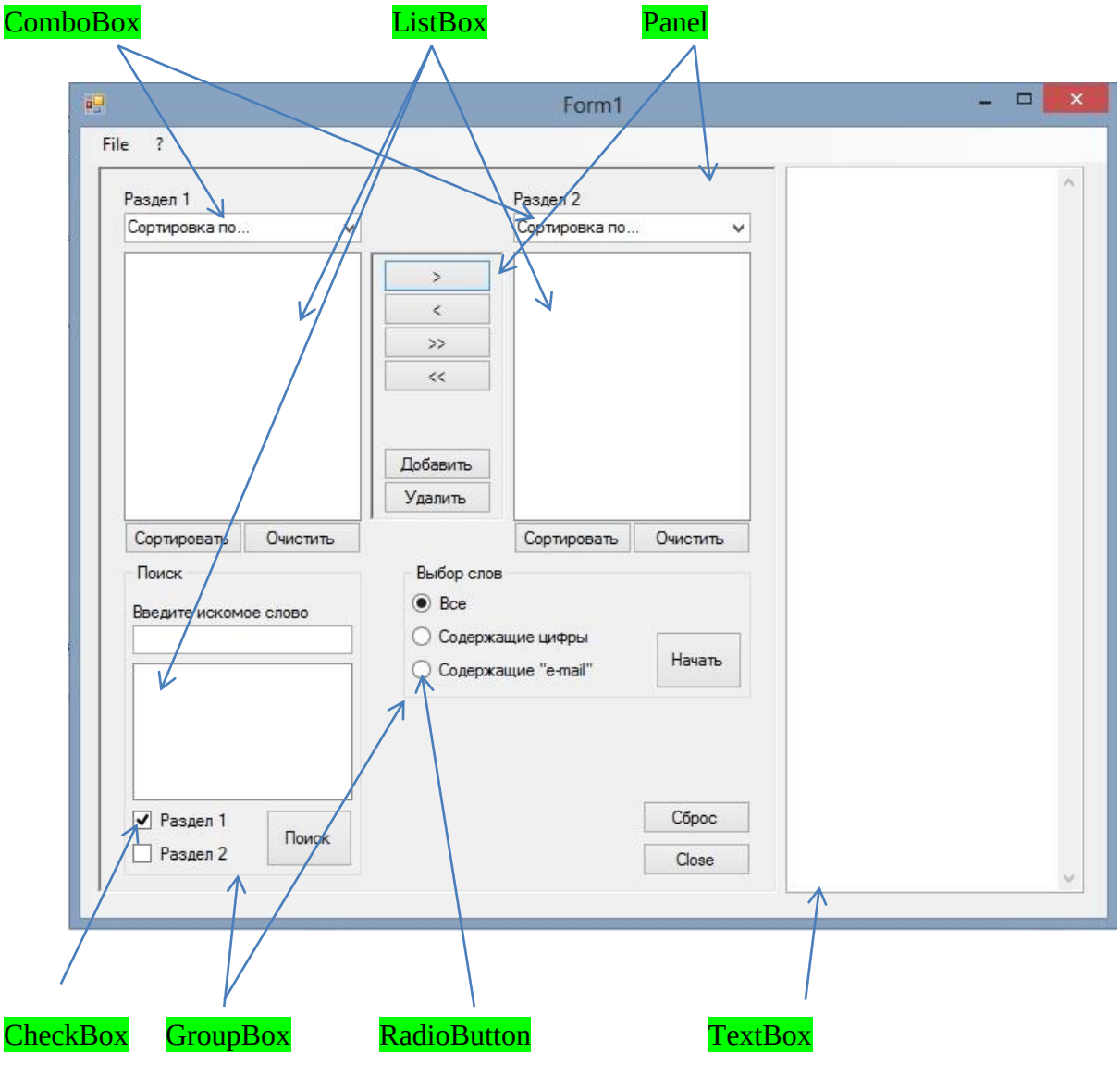
Объектно-визуальное программирование

Лабораторная работа № 1:

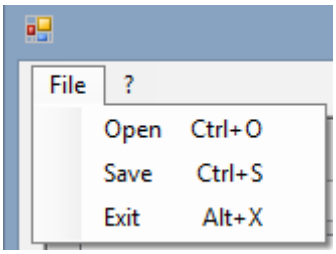
Тема: Работа со списками (компоненты ListBox, ComboBox).
Работа с компонентами RadioButton, CheckBox, GroupBox, Panel.

Задание:

1. Создайте визуальную часть приложения, как показано на рисунке (или произвольным образом, сохранив в наличии все указанные компоненты). У компонентов Panel установите значения свойства BorderStyle (Fixed3D – выпуклая, утопленная). У компонентов RadioButton, CheckBox и ComboBox установите начальные значения свойств Checked, Checked и Text соответственно (см. рисунок).



2. Создайте главное меню следующего вида:



По команде Open: должно появляться окно диалога открытия файла (используем OpenFileDialog), содержимое выбранного файл загружается в TextBox.

По команде Save: содержимое Раздела 2 сохраняется в выбранном файле (используем SaveFileDialog). Для этого надо последовательно перезаписать все строки из ListBox2 в файл. Для обращения к строкам ListBox используется: **ListBox1.Items**.

По команде Exit закрывается приложение.

По команде ? выдается информация об авторах данного программного продукта.

```
private void openToolStripMenuItem_Click(object sender, EventArgs e)
            {
                  OpenFileDialog dlg = new OpenFileDialog();

                  if (dlg.ShowDialog() == DialogResult.OK)
                  {
                        StreamReader reader = new StreamReader(dlg.FileName,
Encoding.Default);
                        txtFileContents.Text = reader.ReadToEnd();
                        reader.Close();
                  }

                  dlg.Dispose();
            }

            private void saveToolStripMenuItem_Click(object sender, EventArgs e)
            {
                  SaveFileDialog dlg = new SaveFileDialog();

                  if (dlg.ShowDialog() == DialogResult.OK)
                  {
                        StreamWriter writer = new StreamWriter(dlg.FileName);
                        for (int i = 0; i < lstSection2.Items.Count; i++)
                        {
                              writer.WriteLine((string)lstSection2.Items[i]);
                        }
                        writer.Close();
                  }

                  dlg.Dispose();
            }



            private void aboutToolStripMenuItem_Click(object sender, EventArgs e)
            {
                  MessageBox.Show("Information about application and developer");
            }
```

3. Создайте обработчики события “Нажатие на кнопку” для кнопок Сброс, Выход.

По кнопке *Сброс* – возврат формы в исходное состояние: очистка рабочей области (TextBox), области поиска, Раздела 1 и Раздела 2, а также установка CheckBox, RadioButton и ComboBox в начальное состояние.

По кнопке *Close*– выход из программы.

```
private void btnReset_Click(object sender, EventArgs e)
            {
                  lstSection1.Items.Clear();
                  lstSection2.Items.Clear();
                  lstSearchResults.Items.Clear();
                  txtFileContents.Clear();
                  chkSection1.Checked = true;
                  chkSection2.Checked = false;
                  rdoAll.Checked = true;
            }
private void exitToolStripMenuItem_Click(object sender, EventArgs e)
            {
                  Application.Exit();
            }
```

4. По кнопке *Начать*: содержимое TextBox разбивается на отдельные слова (признак окончания очередного слова - встреча пробела), которые последовательно помещаются в Раздел 1 в зависимости от значения радиокнопок. Один из способов реализации:

- из TextBox последовательно считываются строки;

- очередная считанная строка делится на слова;

- каждое слово проверяется на соответствие критерию выбора (определяется значениями радиокнопок (RadioButton): если соответствует -помещается в Раздел 1 (ListBox1). Чтобы поместить в ListBox данные (STRING) воспользуемся **ListBox1.Items.Add(**STRING**).**

```
private void btnStart_Click(object sender, EventArgs e)
            {
                  lstSection1.Items.Clear();
                  lstSection2.Items.Clear();

                  lstSection1.BeginUpdate();
                  string[] strings = txtFileContents.Text.Split(new char[] { '\n',
'\t', ' '}, StringSplitOptions.RemoveEmptyEntries);
                  foreach (string s in strings)
                  {
                        string str = s.Trim();
                        if (str == String.Empty)
                              continue;
                        if (rdoAll.Checked)
                        {
                              lstSection1.Items.Add(str);
                        }
                        if (rdoDigits.Checked)
                        {
                              if (Regex.IsMatch(str, @"\d"))
                                    lstSection1.Items.Add(str);
                        }
                        if (rdoEmails.Checked)
                        {
                              if (Regex.IsMatch(str, @"\w+@\w+\.\w+"))
                                    lstSection1.Items.Add(str);
```

```
                }

            }
            lstSection1.EndUpdate();
        }
```

5. Реализовать:
- перемещение данных (полное или только выделенных строк) из одного Раздела в другой,
- удаление выделенных строк из Разделов 1 и 2,
- добавление данных в Раздел 2.

5.1. Для перемещения строк из одного Раздела в другой (*полное*) необходимо вначале добавить все элементы из одного ListBox в другой (с помощью метода Add), а затем очистить ListBox (Clear), из которого мы производили перемещение. Чтобы узнать количество элементов в ListBox, можно воспользоваться ListBox1.Items.**Count**.

Для возможности выделять несколько строк в списке ListBox, установите его свойство SelectionMode = MultiExtended.

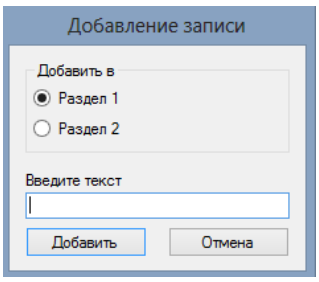
5.2. Для перемещения строк из одного Раздела в другой (частично), вначале необходимо найти выделенную строку (ListBox1.**SelectedItems[**номер выделенной строки**]** – true, если строка выделена). После того, как строка найдена, добавляем ее в другой раздел. Затем эту строку удаляем (используем ListBox1.Items.**RemoveAt(**номер удаляемой строки**)**).

5.3. Удаление выделенных строк из Раздела1(2) или одновременно из того и другого производится аналогично перемещению строк (частичное). Т.е. вначале мы ищем выделенные строки (Selected), затем удаляем найденную строку (RemoveAt). После этого переходим в другой раздел и проделываем то же самое. Предварительно (перед поиском) надо делать проверку наличия данных в Разделе (если пустой – проверка другого раздела …)

```
private void MoveSelectedItems(ListBox lstFrom, ListBox lstTo)
        {
            lstTo.BeginUpdate();
            foreach (object item in lstFrom.SelectedItems)
            {
                lstTo.Items.Add(item);
            }
            lstTo.EndUpdate();

            DeleteSelectedItems(lstFrom);
        }

        private void MoveAllItems(ListBox lstFrom, ListBox lstTo)
        {
            lstTo.Items.AddRange(lstFrom.Items);
            lstFrom.Items.Clear();
        }

        private void btnMoveToSection2_Click(object sender, EventArgs e)
        {
```

```
        MoveSelectedItems(lstSection1, lstSection2);
}

private void btnMoveToSection1_Click(object sender, EventArgs e)
{
        MoveSelectedItems(lstSection2, lstSection1);
}

private void btnMoveAllToSection2_Click(object sender, EventArgs e)
{
        MoveAllItems(lstSection1, lstSection2);
}

private void btnMoveAllToSection1_Click(object sender, EventArgs e)
{
        MoveAllItems(lstSection2, lstSection1);
}

private void btnDelete_Click(object sender, EventArgs e)
{
        DeleteSelectedItems(lstSection1);
        DeleteSelectedItems(lstSection2);
}
```

5.4. Для добавления данных в Раздел1(2) создадим модальную форму следующего вида.



По кнопке Добавить:

- данная форма закрывается;

- на главной форме в список выбранного Раздела помещается введенный текст.

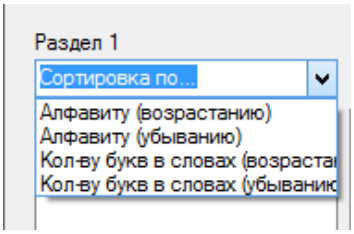
```
private void btnAdd_Click(object sender, EventArgs e)
        {
                AddItemDialog dlg = new AddItemDialog();
                if (dlg.ShowDialog() == DialogResult.OK)
                {
                        switch (dlg.Section)
                        {
                                case 1: lstSection1.Items.Add(dlg.UserInput); break;
                                case 2: lstSection2.Items.Add(dlg.UserInput); break;
                                default: break;
                        }
                }
        }
```

6. Компонентам ComboBox следует присвоить значения:

Алфавиту (по возрастанию)

Алфавиту (по убыванию)

Длине слова (по возрастанию)



Длине слова (по убыванию)

7. Создайте обработчики для кнопок Сортировать, Очистить.
Очистить – очищает Раздел1(2) (т.е. ListBox1(2)). Чтобы произвести эту операцию используйте метод **Clear**.

```
private void btnClearSection1_Click(object sender, EventArgs e)
        {
                lstSection1.Items.Clear();
        }
```

Сортировать – производит сортировку данных, находящихся в Разделе1(2) по заданному критерию. Для организации сортировки необходимо сначала определить тип сортировки (т.е. по алфавиту (по возрастанию или убыванию) или по длине слов (по возрастанию или убыванию). Для этого используется проверка значения **ComboBox1.SelectedIndex**. Затем произвести сортировку данных любым известным способом сортировки. Для обращения к строкам ListBox используется: **ListBox1.Items.**.

Примечание: во время сортировки курсор мыши должен изменять свою форму (становиться песочными часами).

```
this.Cursor = Cursors.Default;
```

По окончании сортировки курсор должен восстанавливать свой исходный вид.

```
private void SortListBox(ListBox list, SortOrder order)
        {
                IComparer<string> comparer = null;
                switch (order)
                {
                        case SortOrder.Alphabetically:
                                comparer = new StringAlphabeticalComparer();
                                break;
                        case SortOrder.AlphabeticallyDesc:
                                comparer = new StringAlphabeticalDescComparer();
                                break;
                        case SortOrder.WordLength:
                                comparer = new StringWordLengthComparer();
                                break;
                        case SortOrder.WordLengthDesc:
                                comparer = new StringWordLengthDescComparer();
                                break;
                        default:
                                MessageBox.Show("Выберите критерий сортировки!",
"Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                                return;
                }

                this.Cursor = Cursors.WaitCursor;

                string[] stringsArray = new string[list.Items.Count];
                list.Items.CopyTo(stringsArray, 0);
                Array.Sort(stringsArray, comparer);
                list.Items.Clear();
                list.Items.AddRange(stringsArray);
```

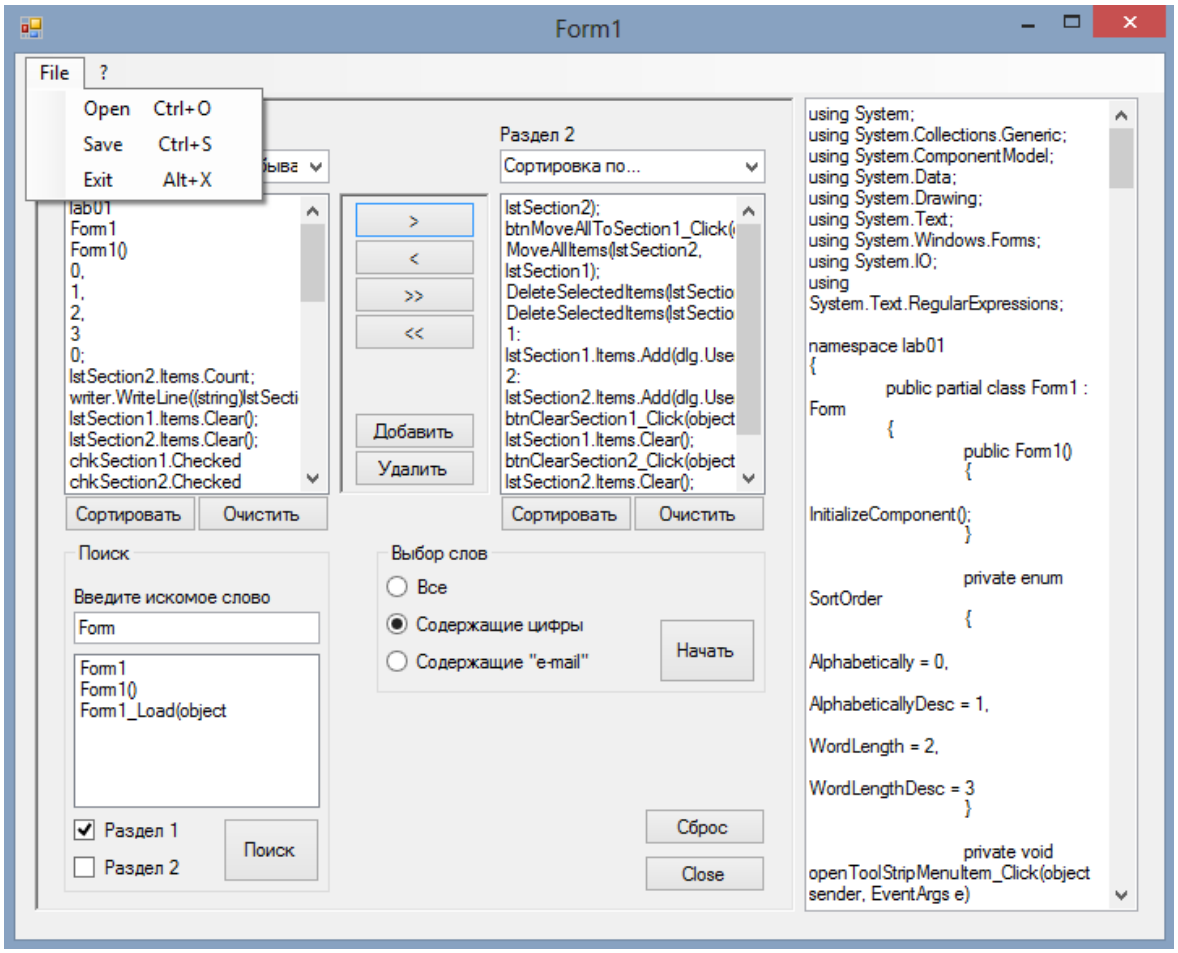
```
                this.Cursor = Cursors.Default;
        }

        private void btnSortSection1_Click(object sender, EventArgs e)
        {
                SortListBox(lstSection1,
(SortOrder)cboSortOrderSection1.SelectedIndex);
        }

        private void btnSortSection2_Click(object sender, EventArgs e)
        {
                SortListBox(lstSection2,
(SortOrder)cboSortOrderSection2.SelectedIndex);

        }
```

8. Реализовать поиск данных.

Для поиска необходимо вначале ввести ключ поиска (textBox1), очистить ListBox3, затем определить область поиска (т.е. Раздел1, Раздел2 или оба, для этого используйте свойство Checked компонента CheckBox). После этого начинаем последовательно просматривать элементы соответствующего раздела (ListBox….**Items**). При нахождение нужного элемента добавляем его в ListBox3.

```
private void btnFind_Click(object sender, EventArgs e)
        {
                lstSearchResults.Items.Clear();
                string textToFind = txtToFind.Text;

                if (chkSection1.Checked)
                {
                        foreach (string s in lstSection1.Items)
                        {
                                if (s.Contains(textToFind))
                                        lstSearchResults.Items.Add(s);
                        }
                }

                if (chkSection2.Checked)
                {
                        foreach (string s in lstSection2.Items)
                        {
                                if (s.Contains(textToFind))
                                        lstSearchResults.Items.Add(s);
                        }
                }

        }
```

Пример работы программы:


Form1
File
?
Open Ctrl+O
Save Ctrl+S
Exit Alt+X
Раздел 2
Сортировка по...
lab01
Form1
Form1()
0,
1,
2,
3
0;
lstSection2.Items.Count;
writer.WriteLine((string)lstSecti
lstSection1.Items.Clear();
lstSection2.Items.Clear();
chkSection1.Checked
chkSection2.Checked
lstSection2);
btnMoveAllToSection1_Click(
MoveAllItems(lstSection2,
lstSection1);
DeleteSelectedItems(lstSectio
DeleteSelectedItems(lstSectio
1:
lstSection1.Items.Add(dlg.Use
2:
lstSection2.Items.Add(dlg.Use
btnClearSection1_Click(object
lstSection1.Items.Clear();
btnClearSection2_Click(object
lstSection2.Items.Clear();
>
<
>>
<<
Добавить
Удалить
Сортировать
Очистить
Сортировать
Очистить
Поиск
Введите искомое слово
Form
Form1
Form1()
Form1_Load(object
Раздел 1
Раздел 2
Поиск
Выбор слов
Все
Содержащие цифры
Содержащие "e-mail"
Начать
Сброс
Close
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO;
using
System.Text.RegularExpressions;
namespace lab01
{
public partial class Form1 :
Form
{
public Form1()
{
InitializeComponent();
}
private enum
SortOrder
{
Alphabetically = 0,
AlphabeticallyDesc = 1,
WordLength = 2,
WordLengthDesc = 3
}
private void
openToolStripMenuItem_Click(object
sender, EventArgs e)

Лабораторные работы № 2, 3:

Тема:

1. Работа с компонентами MenuStrip; ToolStrip; Panel; ColorDialog; OpenFileDialog; SaveFileDialog; PictureBox; ImageList; TrackBar; ComboBox.

2. Создание графического редактора, позволяющего:
   - Загружать, редактировать, создавать, сохранять изображения;
   - Рисовать с помощью мыши (При нажатии левой кнопки мыши и её перемещении отображается кривая движения указателя мыши. При нажатии правой кнопки мыши появляется стирательная резинка. При двойном щелчке - стирается весь рисунок).
   - Задавать цвет, толщину и стиль линии;

Теория: Компонент MenuStrip

Для быстрого вызова команд можно использовать так называемые быстрые клавиши. Для этого надо установить свойство ShowShortCutKeys, выбрав значение True. Также установить свойство ShortCutKeys, выбрав значение из списка (или набрать). При этом нужно следить, чтобы быстрые клавиши не повторялись во избежание коллизий.

Можно использовать любые готовые иконки либо создать их самостоятельно. Для этого в свойствах необходимо найти Image и дважды нажать на значение свойства, появится окно «Выбор ресурса». В окне выберете контекст ресурса (Локальный или Файл ресурсов проекта). Локальный – если вы хотите установить собственную иконку, Файл ресурсов проекта – если вас устраивают стандартные иконки (windows theme).

Компонент ToolStrip. Представляет собой специальный контейнер для создания панелей инструментов. Может управлять любыми вставленными в него дочерними элементами: группировать, выравнивать по размерам, располагать элементы в несколько рядов.

Специально для ToolStripPanel разработан компонент ToolStripButton (кнопка панели инструментов, отсутствует в палитре компонентов). Для добавления в панель компонента ToolStripButton надо: щелкнуть правой кнопкой мыши на ToolStripPanel и выбрать Button|Label|SplitButton|DropDownButton|Separator|ComboBox|TextBox|ProgressBar.

На кнопки можно поместить изображения. Для этого надо установить свойство Image

Компонент PictureBox. Служит для размещения на форме одного из трех поддерживаемых типов изображений: растрового изображения, значка и метафайла.

Растровое изображение – это произвольные графические изображения в файлах со стандартным расширением .bmp. Значки (иконки) – небольшие растровые изображения, снабженные специальными средствами, регулирующими их прозрачность. Расширение файлов значков, обычно, .ico . (Метафайл – это изображение, построенное на

графическом устройстве с помощью специальных команд, которые сохраняются в файле с расширением .wmf .)

Компонент TrackBar – ползунок. Для плавного изменения числовой величины (во многом схож с компонентом ScrollBar).

Некоторые свойства:
- Maximum – определяет максимальное значение диапазона изменения;
- Minimum – определяет минимальное значение диапазона изменения;
- Value: integer – определяет текущее положение ползунка;
- Orientation – ориентация компонента (горизонтальная, вертикальная);

События мыши. Для выполнения какого-либо действия с помощью щелчка левой кнопки мыши для большинства случаев достаточно запрограммировать обработчик событий OnClick, для реакции на двойной щелчок используется событие OnDbClick. Для более совершенного управления мышью лучше использовать обработчики следующих событий:

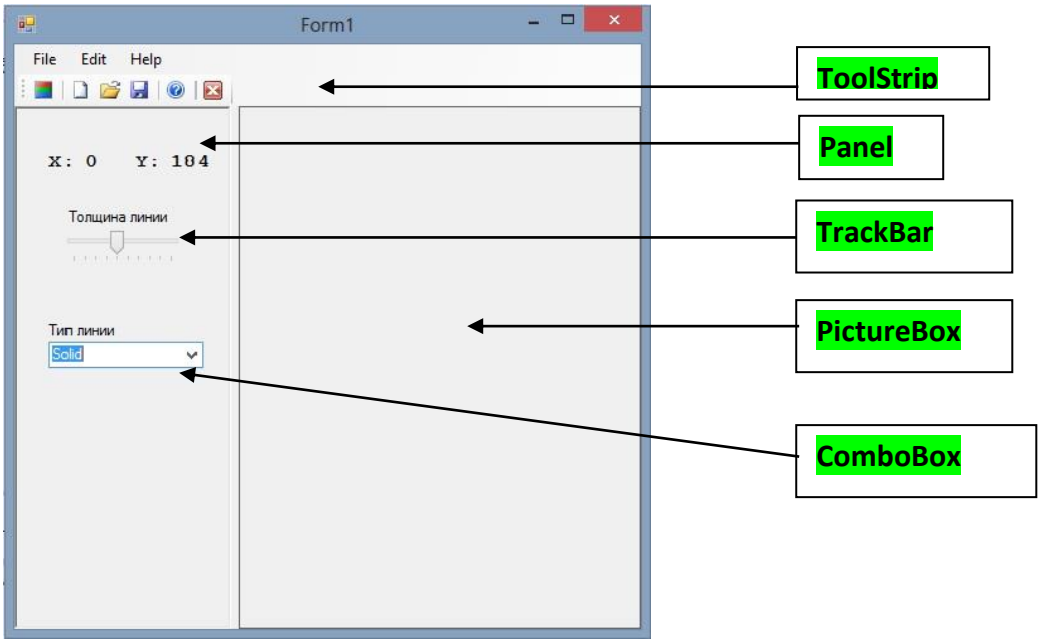
- OnMouseDown. Вызывается при нажатии любой кнопки мыши.
- OnMouseMove. Вызывается при перемещении мыши.
- OnMouseUp. Вызывается при отпускании какой-нибудь кнопки мыши.

Процедуры обработки этих событий получают следующие параметры:

- Sender. Представляет объект, который получил это событие (на каком объекте щелкнули мышью).
- Button. Имеет одно из трех значений: MouseButtons.Right, MouseButtons.Left, MouseButtons.Middle. Используется для определения того, какую кнопку мыши нажал пользователь.
- X, Y. Координаты указателя мыши в пикселях относительно клиентной области окна с координатами (0,0) в верхнем левом углу.

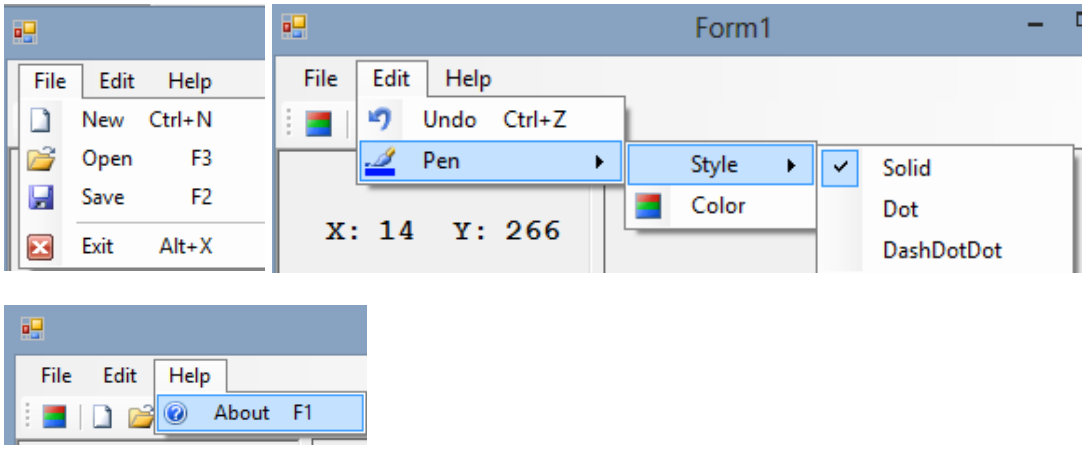
Задание 1:

Вид формы после создания визуальной части показан на рисунке:



Создание визуальной части.

1. Создайте главное меню следующего вида:



2. Поместите на форму компоненты ToolStrip. Добавьте требуемое количество кнопок для дублирования следующих команд главного меню: Color, New, Open, Save, About, Exit. Поместите на кнопки соответствующие изображения. Попробуйте разместить и текст (в готовом Приложении оставьте на кнопках только картинки).

3. Поместите на форму компоненты Panel, ComboBox, TrackBar. Панель разместите в левой, правой или в нижней части формы.

4. Поместите на форму компонент PictureBox. На него поместите четыре компонента

X: 65 Y: 9

label для отображения координат указателя мыши в виде:

Создание невизуальной части.

5. Запрограммируйте работу всех пунктов меню File. Для этого поместите на форму компоненты OpenFileDialog и SaveFileDialog (см.пример в лабораторной 1).

6. Запрограммируйте работу пункта About меню Help: должна появляться форма с кратким описанием возможностей Вашего графического редактора (см.пример в лабораторной 1).
7. В разделе private опишите переменную Dragging типа boolean, которая принимает значение True, если мышь находиться в режиме рисования. В обработчике события OnCreate формы установите значение этой переменной равным False.

```
private bool _dragging = false;
```

8. В процедуре обработки события OnMouseDown в случае нажатия левой кнопки установите режим рисования (переменная Dragging), установите графический

указатель в позицию указателя мыши и отобразите текущие координаты указателя мыши в соответствующих компонентах label.

```
if (e.Button == MouseButtons.Left)
{
    _dragging = true;
    _oldLocation = e.Location;
    _currentPath = new GraphicsPath();
}
```

9. В процедуре обработки события OnMouseMove, если установлен режим рисования, проводится прямая линия из текущей позиции графического указателя в новую позицию указателя мыши. После выполнения метода AddLine графический указатель автоматически перемещается в конечную точку. В процедуре OnMouseMove получают новые значения координаты указателя мыши, отображаемые в соответствующих компонентах Label.

```
lblX.Text = e.X.ToString();
lblY.Text = e.Y.ToString();
if (_dragging)
{
          Graphics g = Graphics.FromImage(picDrawingSurface.Image);
          _currentPath.AddLine(_oldLocation, e.Location);
          g.DrawPath(_currentPen, _currentPath);
          _oldLocation = e.Location;
          g.Dispose();
          picDrawingSurface.Invalidate();
}
```

10. В процедуру обработки события OnMouseUp добавьте отмену использования мыши для рисования.

```
if (e.Button == MouseButtons.Left)
{
          _dragging = false;
          try
          {
          _currentPath.Dispose();
          }
          catch { };
}
```

11. Запрограммируйте работу пункта меню Undo (стирает рисунок) по двойному щелчку левой кнопки мыши.

```
Graphics g = Graphics.FromImage(picDrawingSurface.Image);
g.Clear(_bgColor);
g.Dispose();
picDrawingSurface.Invalidate();
```

Задание 2. Создание дополнительных сервисных возможностей

1. Запрограммируйте работу всех пунктов подменю Pen меню Edit. Эти команды устанавливают стиль и цвет рисования. По команде Color должно открываться окно диалога для выбора цвета (компонент ColorDialog закладки Dialogs).

```
ColorDialog dlg = new ColorDialog();
if (dlg.ShowDialog() == DialogResult.OK)
{
    _fgColor = dlg.Color;
```

```
        _currentPen.Color = _fgColor;
    }
```

2. Сделайте выбор толщины линии с помощью компонента TrackBar. У TrackBar установите свойства Minimum=1 Maximum=10 Value =5

```
    _currentPen.Width = trkPenWidth.Value;
```

3. Запрограммируйте «стирательную резинку». При нажатии правой кнопки мыши изображение стирается. Величина резинки определяется текущей толщиной линии. Для отображения резинки можно использовать нестандартный курсор.

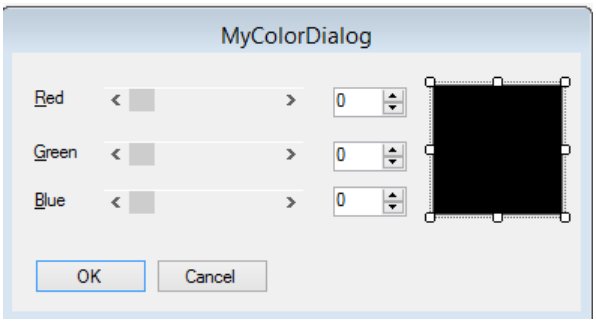
```
if (e.Button == MouseButtons.Right)
{
        if (picDrawingSurface.Cursor == _penCursor)
        {
                picDrawingSurface.Cursor = _eraserCursor;
                _currentPen.Color = _bgColor;
        }
        else
        {
                picDrawingSurface.Cursor = _penCursor;
                _currentPen.Color = _fgColor;
        }
}
```

Лабораторная работа 3.

Тема: Создание диалогового окна выбора цвета линии.

Задание 1:

1. Разместите на форме три метки с названиями Red, Green, Blue, три компонента HScrollBar, три компонента NumericUpDown, label, кнопки Cancel и OK. Форма должна иметь следующий вид:



2. Для компонентов HScrollBar установите свойство Minimum равное 0, Maximym – 255 (диапазон изменения оттенков цветов), LargeChage – 1 (величина прокрутки при щелчке на полосе).
3. Для компонентов NumericUpDown установите свойства Maximum, Minimum.
4. Для связывания компонентов HScrollBar и NumericUpDown во время работы приложения установите одинаковыми свойства Tag для каждой пары объектов.

```
InitializeComponent();
   sbrRed.Tag = nudRed;
   sbrGreen.Tag = nudGreen;
   sbrBlue.Tag = nudBlue;

   nudRed.Tag = sbrRed;
   nudGreen.Tag = sbrGreen;
   nudBlue.Tag = sbrBlue;
```

5. Напишите процедуру UpdateColor, которая смешивает цвета на основании текущих положений всех бегунков и показывает этот цвет в окне Label. Для смешивания цветов используйте макрос RGB с тремя параметрами целого типа: интенсивности красного, зеленого и голубого цвета. Этот макрос возвращает цвет, полученный в результате смешивания. Опишите процедуру UpdateColor.

```
Color c = Color.FromArgb(sbrRed.Value, sbrGreen.Value, sbrBlue.Value);
lblCurrentColor.BackColor = c;
```

6. Запрограммируйте событие ValueChange для объектов ScrollBar, NumericUpDown в соответствии с положением бегунка. В этом же обработчике необходимо обновить цвет смешивания с помощью процедуры UpdateColor.

```
private void nudColor_ValueChanged(object sender, EventArgs e)
```

```
{
    NumericUpDown numericUpDown = (NumericUpDown)sender;
    ScrollBar scrollBar = (ScrollBar)numericUpDown.Tag;
    scrollBar.Value = (int)numericUpDown.Value;
}

private void sbrColor_ValueChanged(object sender, EventArgs e)
{
    UpdateColor();
    ScrollBar scrollBar = (ScrollBar)sender;
    NumericUpDown numericUpDown = (NumericUpDown)scrollBar.Tag;
    numericUpDown.Value = scrollBar.Value;
}
```

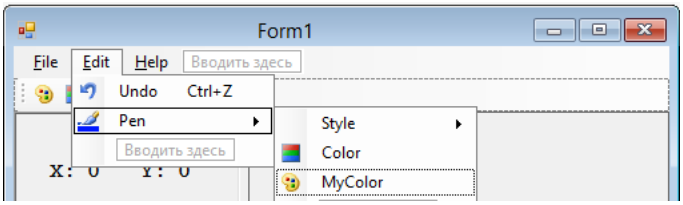
7. Для кнопки OK выполните сохранение текущих положений бегунков.

```
if (dlg.ShowDialog() == DialogResult.OK)
{
    _fgColor = dlg.Color;
    _currentPen.Color = _fgColor;
}
```

8. Для кнопки Cancel выполните сброс текущих положений бегунков в последнее сохраненное положение.

Задание 2:

1. Добавьте в подменю Pen меню Edit команду MyColor. По этой команде должно появляться созданное в Задании1 диалоговое окно выбора цвета и устанавливаться цвет линии.



2. Добавьте в панель компонента ToolStripButton новую кнопку (переместите ее в начало панели), соответствующую команде MyColor. Создайте и поместите на нее изображение.

Лабораторная работа №4

Тема: Работа с компонентами Panel; ComboBox, ComboBox, ListBox, splitter, Image, StatusBar.

*Задание 1.*Создать форму как показано на рисунке 1.

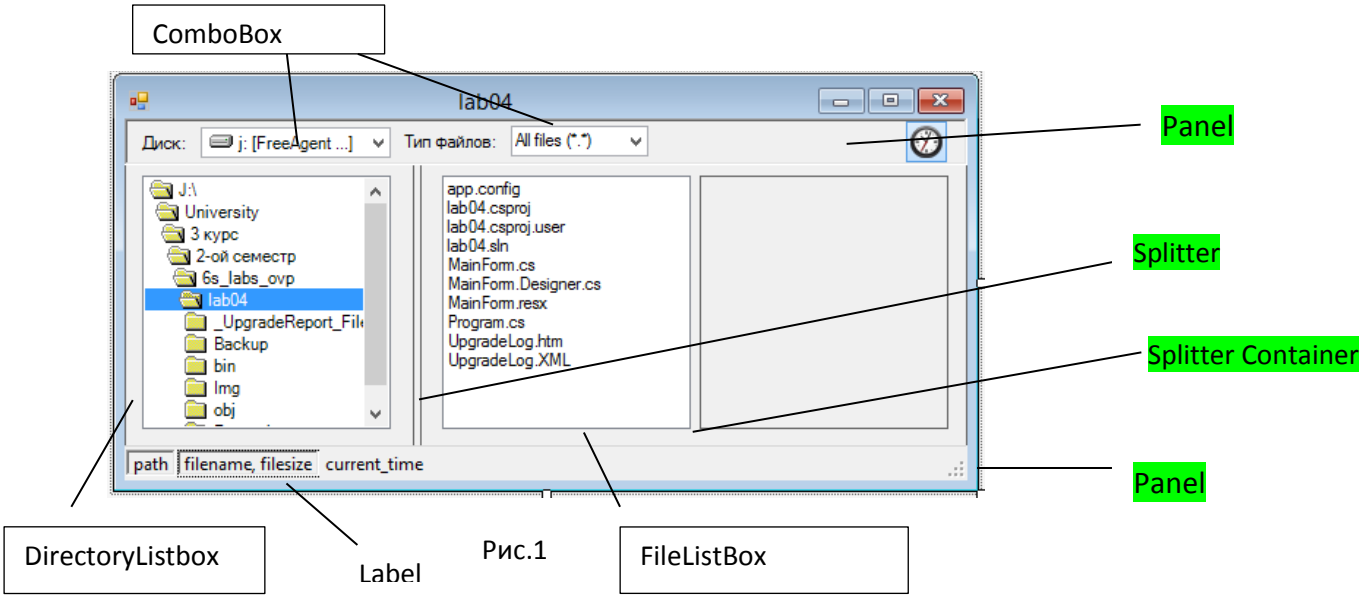


Рис.1

1. Рекомендуемый порядок размещения на форме компонентов Panel, DirectiryListBox, FileListBox и Splitter:
   - Panel1, Panel2 – в верхней и нижней частях клиентской области ;
   - DirectoryListBox1– слева на клиентской области (свойство Align);
   - Splitter;
   - FileListBox1 – на всю оставшуюся клиентскую область (свойство Align).
2. Для установки связей между списками DriveComboBox1, FilterComboBox1, DirectoryListBox1 и FileListBox1 определите следующие свойства (в Инспекторе объектов или в обработчике onCreate формы): DirList (для DriveComboBox1), *FileList* и *DirLabel* (для DirectoryListBox1), *FileList* (для FilterComboBox1),
3. Для вывода в строку состояния имени и размера выделенного файла создайте обработчик события onChange для FileListBox1
   Для получения имени выделенного файла используйте свойства Items и ItemIndex (номер выделенного файла) компонента FileListBox1.

Задание 2.: Изменить Form1 на рис. 1 (задание 1) следующим образом: использовать для реализации строки состояния вместо компонента Panel компонент StatusBar. Структура компонента StatusBar показана на рис. 2.

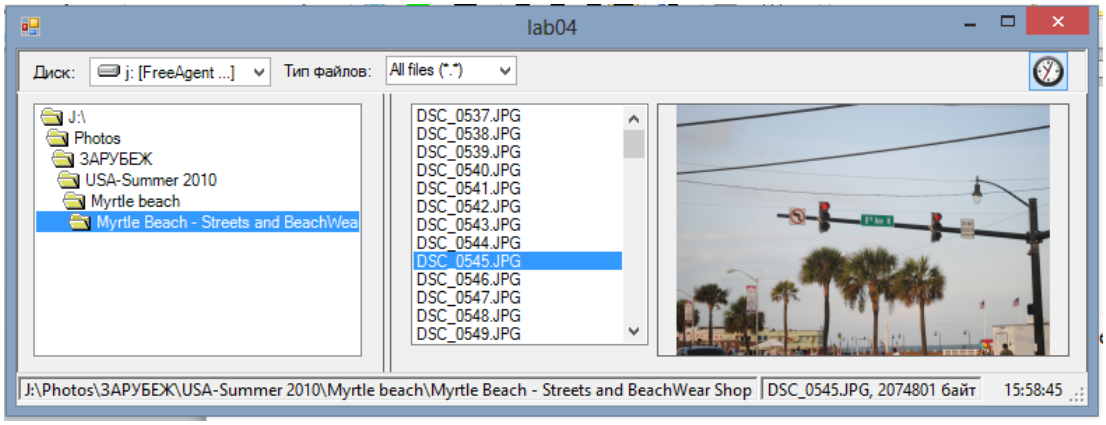


Рис. 2

Размер первых двух панелей зависит от значения свойства Directory компонента DirectoryListBox1. Размер третьей панели, отображающей текущее время, не меняется.

Задание 3:

1. Поместить на форму компонент PictureBox как показано на рис.2. Отображать внутри компонента графическое изображение только в том случае, если в FileListBox1 выбрано изображение.
2. Поместить на Panel1 компонент RadioButton (быстрая кнопка) с изображением часов (или с любым другим).

   При нажатии на эту кнопку:

   - время в строке статуса должно выключаться;
   - состояние кнопки - нажата.

Лабораторная работа № 5

Тема: Работа с компонентом TabControl, TabPages; События MouseEventArgs, DragEventArgs.

Задание: разобраться с работой указанных в теме компонентов и событий.

1. На вкладке Header нужно реализовать событие: по нажатию кнопки текст, написанный на кнопке должен переноситься на label.
2. На вкладке Drag'n'Drop требуется реализовать событие переноса Drag and drop: В TextBox вводим текст/слово, и переносим на label.
3. Добавить 3-ю секцию Editor, создать еще одну страницу PageControl, изменить вид открытки.

Теория: Компонент HeaderControl

(Панель заголовков).

Панель заголовков позволяет разместить на форме заголовки произвольных элементов. Порядок и размеры этих заголовков можно менять.

```
private void header1_SectionTracking(object sender, HeaderSectionWidthConformableEventArgs ea)

        {
            HeaderSection section = ea.Item;
            section.Width = ea.Width;

            switch (section.Text)
            {
                case "Кнопки":
                    btnUpper.Width = ea.Width;
                    btnMiddle.Width = ea.Width;
                    btnLower.Width = ea.Width;

                    lblUpper.Left = section.Header.Left + section.Width;
                    lblMiddle.Left = section.Header.Left + section.Width;
                    lblLower.Left = section.Header.Left + section.Width;
                    break;
                case "Текст":
                    lblUpper.Width = ea.Width;
                    lblMiddle.Width = ea.Width;
                    lblLower.Width = ea.Width;
                    break;
            }
        }
```

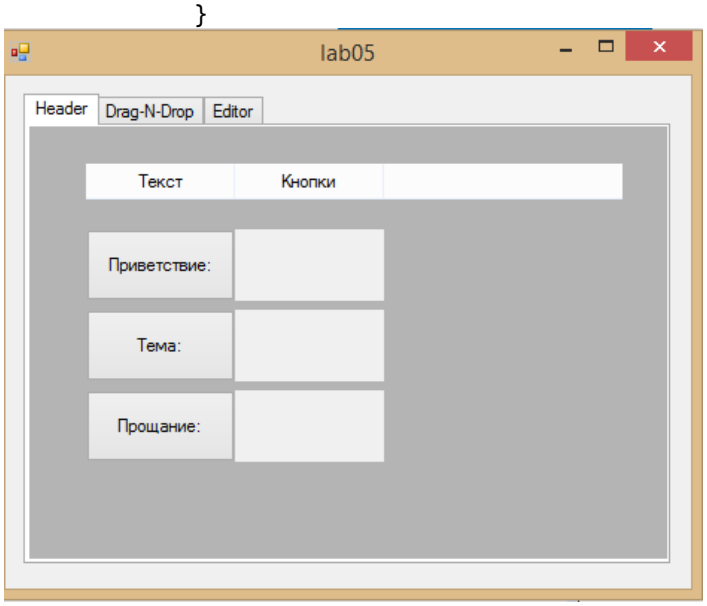
Этот алгоритм работы данного компонента позволяет подстраивать размеры других объектов (в данном случае кнопки и метки) под размеры разделов заголовка.

Сами разделы задаются в свойстве Sections.

Каждый из разделов заголовка может работать как кнопка.

Для этого значение свойства Clickable устанавливается True и записываются соответствующие действия в метод SectionClick(object, event)

```
private void header1_SectionClick(object sender, HeaderSectionEventArgs ea)
            {
                  HeaderSection section = ea.Item;

                  switch (section.Text)
                  {
                        case "Кнопки":
                              if (isLabelsEmpty)
                              {
                                    string temp = btnUpper.Text;
                                    btnUpper.Text = btnMiddle.Text;
                                    btnMiddle.Text = btnLower.Text;
                                    btnLower.Text = temp;
                              }
                              else
                              {
                                    MessageBox.Show("Сначала нажмите заголовок
\"Текст\"", "Information",
                                    MessageBoxButtons.OK,
MessageBoxIcon.Information);
                              }
                              break;
                        case "Текст":
                              isLabelsEmpty = true;
                              lblUpper.Text = string.Empty;
                              lblMiddle.Text = string.Empty;
                              lblLower.Text = string.Empty;
                              break;
                  }
            }
```



Коллекция TabPages.

Число страниц многостраничного блокнота TabControl указывается в редакторе коллекции TabPages. При создании приложения программист может переходить на

другие страницы, щелкая на их ярлыках мышью, переключение страниц программным способом является более трудной и не слишком удобной для разработчика задачей.

При переходе на другую страницу с ярлыком возникает событие `HeaderSectionEventArgs`. Эти событие можно использовать, например, для проверки состояния размещенных на странице элементов или управления доступом к определенным страницам.

Доступ к определенным страницам осуществляется через параметр события item типа `HeaderSection`, представляющий собой элемент, к которым можно обращаться. Свойство доступно во время выполнения программы.

Каждая страница является объектом класса `TabPages` и может содержать другие компоненты. Для добавления и удаления страниц используется редактор.

Для добавления страницы при разработке приложения следует щелкнуть на нём правой кнопкой мыши и в появившемся контекстном меню выбрать пункт *«Добавить вкладку»*. Для удаления страницы нужно выбрать пункт *«Удалить вкладку»*.

Механизм перетаскивания.

1. Для объекта создается обработчик события `MouseEventArgs`. В нем проверяется нажата ли именно левая кнопка, и вызывается метод `DoDragDrop`.

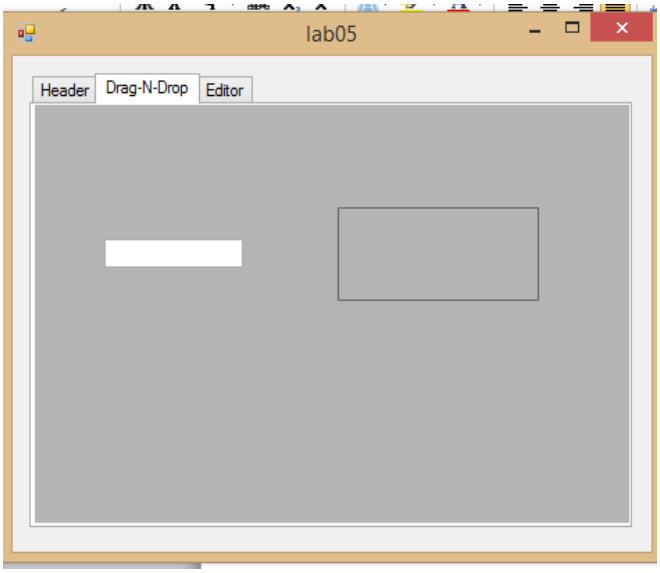
```
private void textBox1_MouseDown(object sender, MouseEventArgs e)
{
        label1.BackColor = SystemColors.ActiveBorder;
        textBox1.DoDragDrop(textBox1.Text, DragDropEffects.Move);
}
```

2. Для объекта создается обработчик события `DragEventArgs`. В нем проверяется, является ли перетаскиваемый объект полем ввода Text.

```
private void label1_DragEnter(object sender, DragEventArgs e)
        {
                if (e.Data.GetDataPresent(DataFormats.Text))
                {
                        e.Effect = DragDropEffects.Move;
                }
        }
```

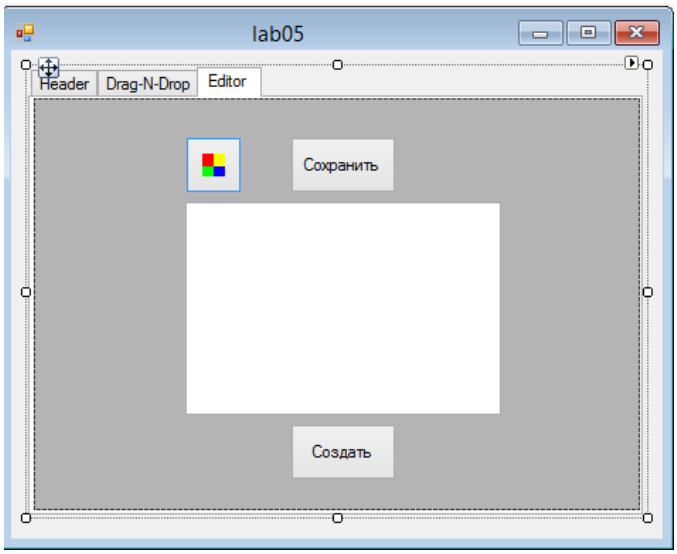
3. Объект должен обрабатывать еще событие `DragEventArgs`, генерируемое, когда происходит сброс перетаскиваемого объекта. В нем происходит итоговое действие: в заголовок панели записывается текст, находящийся в текстовом поле.

```
private void label1_DragDrop(object sender, DragEventArgs e)
        {
                label1.Text = (string)e.Data.GetData(DataFormats.Text);
                textBox1.Text = string.Empty;
                label1.BackColor = Color.AliceBlue;
        }
```



Создание открытки

Создать страницу следующего вида:



TextBox и Button. По нажатию на цветную кнопку возникает диалоговое окно ColorDialog, выбранным цветом которого будет закрашен TextBox. В TextBox можно печатать какое-нибудь поздравление, а при нажатии кнопки «Создать» и «Сохранить» динамически создается новая страница следующего содержания:

С помощью свойств TextBox изменяем скучный серый фон, например, на красный.

TextBox сделать только для чтения, при наведении на него курсор должен меняться, например, на такой: ∅.

В TextBox помещается текст, который мы ввели в поле TextBox на вкладке Editor. Изменяется шрифт, становиться более крупным, и текст должен располагаться по середине.

Лабораторная работа № 6

Задание 1: Перетаскивание графических объектов с помощью мыши (При нажатии левой кнопки мыши внутри границ фигуры и перемещении мыши, графический объект перетаскивается на новое место в соответствии с движением мыши).

Реализация перетаскивания любого объекта Shape возложена на три обработчика событий этого объекта: MouseDown, MouseMove и MouseUp.

1. С помощью компонента PictureBox на палитре компонентов создайте на форме три фигуры: жёлтый прямоугольник со скругленными краями, красный круг и синий квадрат. Для выбора формы фигуры используйте свойство Shape, для определения цвета и шаблона заливки внутри области - свойство Brush, для определения цвета и стиля контура – свойство Pen. Для всех созданных объектов класса Shape используйте одни и те же обработчики событий.
2. В разделе private опишите переменную Dragging типа boolean, которая принимает значение True во время выполнения операции "щелкнуть и перетащить". В обработчике инициализации формы установите значение этой переменной равным False.
3. В процедуре обработки события MouseDown объекта Shape следует выполнить следующие действия:
   - Установить флажок Dragging в True, если нажата левая кнопка мыши.
   - Нарисовать точечный прямоугольник для перетаскивания объекта Shape.
   - Сохранить координаты мыши в момент нажатия левой кнопки мыши для их последующего использования при перерисовке объекта Shape в новом месте в обработчике события MouseUp. Переменные опишите в разделе private.
   - Сохранить координаты мыши в момент нажатия левой кнопки мыши для определения начального положения мыши при перемещении для обработчика события MouseMove (эти переменные изменяют свои значения в этом обработчике событий). Переменные опишите в разделе private.
4. В процедуре обработки события MouseMove в случае, если установлен режим перетаскивания, необходимо выполнить следующие действия:
   - Установить в качестве начальных координат для следующего перемещения мыши текущие координаты мыши.
   - Определить заливку прямоугольника для перетаскивания.
5. Процедура обработки события MouseUp в случае, если установлен режим перетаскивания, должна выполнить следующее:
   - Сбросить флаг Dragging для отмены операции перетаскивания.
   - Удалить последний контур выделения объекта.
   - Переместить фигуру к её новому местоположению, изменив значения свойств Left и Top объекта Shape.

Задание 2. Реализация интерфейса Drag&Drop. Изменение формы фигуры при переносе

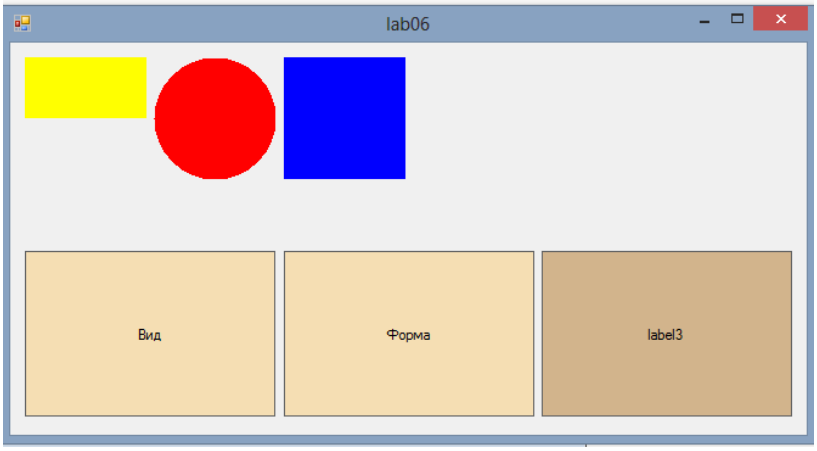
1. Поместите на форму 3 компонента Label, 2 компонента Shape (один из них квадрат, другой эллипс)

Вид	Форма	Красный квадрат

2. Запрограммируйте работу обработчика события MouseEventArgs метки Форма так, чтобы принимались только компоненты типа Shape при перетаскивании компонента на метку

3. В обработчике события MouseEventArgs метки запрограммируйте изменение формы.(квадрат становится окружностью, окружность – квадратом)

4. Запрограммируйте работу обработчика события MouseEventArgs метки Вид так, чтобы принимались только компоненты типа Shape и в третьей метке появляется информация о цвете и форме фигуры при перетаскивании.

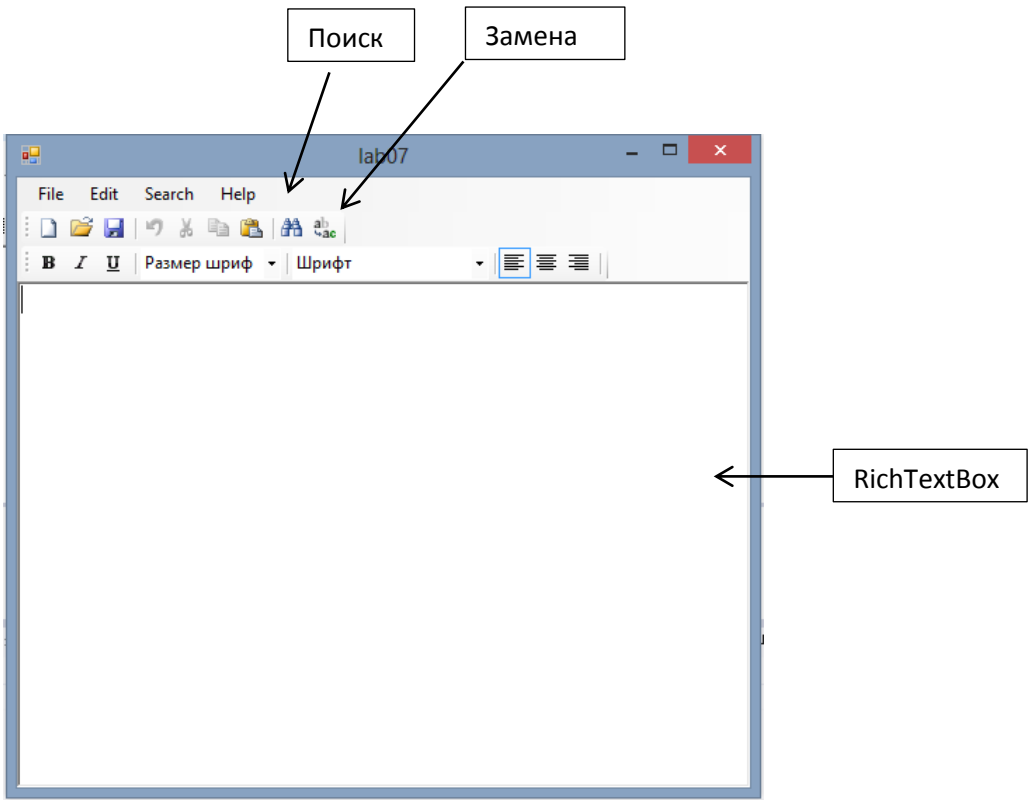


Лабораторная работа №7

Тема:

1. Работа с компонентами: RichTextBox; FindDialog, ReplaceDialog (закладка Dialogs)
2. Создание простейшего текстового редактора, предоставляющего следующие возможности:
   - создание, сохранение, открытие документа (меню File);
   - отмена последних изменений; вырезать, скопировать, вставить фрагмент текста (меню Edit);
   - поиск и замена фрагментов текста (меню Search): компоненты FindDialog, ReplaceDialog (закладка Dialogs);
   - изменение стиля (жирный, курсив, подчеркнутый), размера и имени шрифта выделенного фрагмента текста;
   - выравнивание (справа, центр, слева) выделенного фрагмента текста или текущей строки.

Задание 1. Создайте визуальную часть Приложения как показано на рисунке (или произвольным образом, но с сохранением всех компонентов и функций редактора). Командам главного меню сопоставьте соответствующие картинки. Кнопки “Поиск” и “Поиск и Замена” должны создаваться динамически (т.е. видны только в работающем Приложении).



Рассмотрим динамическое создание объектов на примере создания кнопки ToolStripButton: кнопки Find.

1) Опишите кнопку в разделе private и обработчик события Инициализации.

```
private ToolStripButton btnFind;
```

2) Напишите текст создания кнопки Find:

```
btnFind = new ToolStripButton();
btnFind.DisplayStyle =
System.Windows.Forms.ToolStripItemDisplayStyle.Image;
btnFind.Image = Properties.Resources.FindDialog;
btnFind.ImageTransparentColor = System.Drawing.Color.Magenta;
btnFind.Name = "btnFind";
btnFind.Size = new System.Drawing.Size(23, 22);
btnFind.Text = "&Find";
btnFind.ToolTipText = "Find";
btnFind.Click += new EventHandler(findToolStripMenuItem_Click);
toolStrip2.Items.Add(btnFind);
```

3) Напишите текст обработчика (включая заголовок):

```
private void findToolStripMenuItem_Click(object sender, EventArgs e)
        {
                FindDialog dialog = new FindDialog();
                dialog.Show(this);
        }
```

Каждой кнопке панели инструментов сопоставьте соответствующую подсказку, которая будет появляться при установке на кнопке курсора мыши.

Задание 2. Реализуйте все функции текстового редактора, указанные в п.2 темы лабораторной работы (т.е. напишите обработчики для всех кнопок панели инструментов, показанной на рисунке)

Теория:

Компонент RichTextBox – многострочное редактируемое текстовое поле.

Свойства:

- Lines: String; - содержит набор строк текста (аналогично одноименному свойству компонента TextBox);
- Text: String; - отображает содержимое свойства Lines в виде одной длинной строки;
- SelAttributes: TextAttributes – определяет шрифтовые атрибуты выделенного текста:

  - Size – размер шрифта:

  - FontStyle – стиль шрифта: Bold – жирный

    Italic – курсив

    Underline – подчеркнутый.

  - Name – имя шрифта: Arial, Courer New, Times New Roman, MS Serif, …

    Например, Name:=’Arial’;

- Length: integer; - задает длину в символах выделенной части текста;
- Text: String; - содержит выделенный текст. Установка нового значения Text заменяет выделенный текст на новый, а если нет выделения, вставляет его в позицию курсора;
- Свойство Alignment определяет горизонтальное выравнивание текста абзаца относительно границ компонента.

Методы:

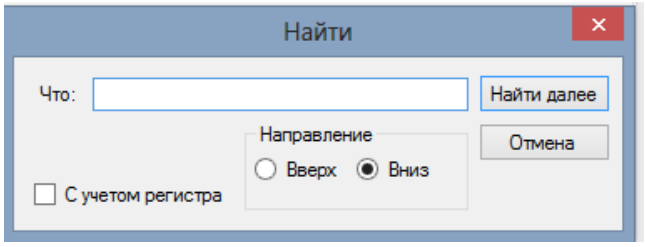
- procedure Clear –удаляет весь текст;
- function FindText (…): integer; - ищет в тексте строку SearchStr и возвращает индекс первого ее символа при удачном поиске: StartPos – начало поиска; Length – длина строки; Options указывает, будет ли поиск идти по целым словам и надо ли учитывать регистр букв (например, [stMatchCase]).
- процедуры Undo; Cut; Copy; Paste.
- Focus – установка фокуса.

Компонент FindDialog – окно для поиска фрагмента текста.

Свойство FindText: String; определяет образец поиска.

Метод Show (стандартный для всех классов диалоговых окон) создает и показывает на экране диалоговое окно поиска. Возвращает значение True при успешном результате диалога с пользователем.

Событие Find: EventArgs возникает всякий раз, когда пользователь щелкает на кнопке Найти далее. Обработчик события должен найти образец в тексте и показать его пользователю.



Компонент ReplaceDialog – окно для поиска и замены фрагмента текста.

Класс ReplaceDialog является прямым потомком класса FindDialog и наследует от него большую часть свойств. Дополнительно в компоненте определено свойство ReplaceText: String, в котором содержится текст замены, и событие Replace, которое возникает при щелчке на кнопке Заменить или Заменить все.