# Лабораторная работа №3. Создание приложений с графическим интерфейсом пользователя

## Часть I. Базовые элементы управления

**Задание 1:** Разработайте приложение, выполняющее печать бейджа участника конференции на основе введенных пользователем данных.

1. Создайте новое решение **Лабораторная работа 3** с проектом типа **Windows Forms** в нем.
2. В **Обозревателе решения** выделите файл **Form1.cs** и переименуйте его в **MainForm.cs**. Подтвердите переименовывание ссылок в появившемся диалоговом окне.
3. На панели **Свойства** измените заголовок окна, указав в поле **Text** значение **«Ввод данных»**.
4. Аналогичным образом измените значения следующих свойств:

| Свойство | Значение | Описание |
|---|---|---|
| `Size` | `600;400` | Размер окна (ширина; длина) |
| `StartPosition` | `CenterScreen` | Расположение окна при открытии по центру экрана |
| `MaximazeBox` | `False` | Отключение кнопки «Развернуть» на рамке окна |
| `BackColor` | `Любой темный` | Цвет фона |
| `ForeColor` | `Любой светлый` | Цвет надписей окна |
| `Font` | `Name = Arial`<br>`Size = 12` | Шрифт (название и размер) |

5. Откройте **Панель элементов** и разместите на форме компоненты для создания интерфейса, подобного указанному на рисунке 1, где поля для ввода имени и фамилии представлены элементами типа **TextBox**, для выбора секции – **ComboBox**, для отображения фотографии – **PictureBox**.

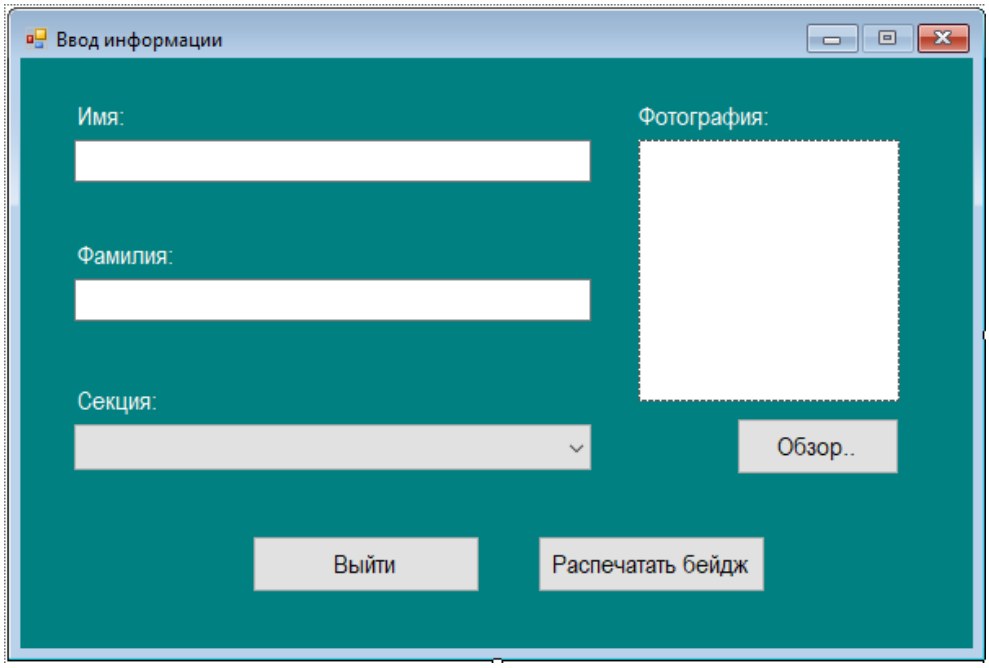


Рисунок 1. Примерный вид формы ввода

 

6. Выполните настройку свойств размещенных компонентов:

| Компонент | Свойство | Значение | Описание |
|---|---|---|---|
| ComboBox | Items | Искусственный интеллект<br>Компьютерная графика<br>Разработка игр | Список значений |
| | Sorted | True | Сортировка списка значений |
| | DropDownStyle | DropDownList | Можно выбрать одно из значений списка, но нельзя ввести другое |
| PictureBox | BackColor | Любой светлый | Цвет заливки компонента |
| | SizeMode | StretchImage | Растягивание изображения по размеру компонента |

7. В полях для редактирования имени и фамилии участника запретите ввод любых символов, кроме русских букв верхнего и нижнего регистра:
   1) выделите компоненты **textBox** на форме;
   2) перейдите на вкладку **Свойства – События**;
   3) для события **KeyPress** создайте обработчик со следующим программным кодом:

```
//если введен символ вне диапазона от  «А» до «я»
//или клавиша не является клавишей Backspace
if ((e.KeyChar < 'А' || e.KeyChar > 'я')&&(e.KeyChar!=8))
   e.Handled=true; //предотвращаем обработку символа
```

8. Запустите программу и проверьте правильность обработки ввода.
9. Обеспечьте выбор значения по умолчанию из списка **Секция**:
   1) выделите форму;
   2) на вкладке **Свойства – События** создайте обработчик события **Load** со следующей инструкцией:

```
comboBox1.SelectedIndex = 0; //установка 1го значения из списка как выбранного
```

10. Разместите на форме невизуальный компонент типа **OpenFileDialog**, предназначенный для стандартного диалога выбора файла.
11. Настройте свойства диалога:

| Свойство | Значение | Описание |
|---|---|---|
| FileName | (пустая строка) | Имя выбранного в диалоге файла |
| Filter | Image Files(*.BMP;*.JPG;*.GIF)\|*.BMP;*.JPG;*.GIF\|All files (*.*)\|*.* | Расширения файлов, которые можно увидеть в диалоговом окне |

12. При нажатии на кнопку **«Обзор»** (в событии **Click**) выполните выбор фотографии и ее отображение на форме:

```
//если в диалоге выбора пользователь нажал на кнопку OK
if (openFileDialog1.ShowDialog() == DialogResult.OK)
{
   pictureBox1.Load(openFileDialog1.FileName); //загружаем изображение в компонент
}
```

 

13. Укажите действия, выполняемы при нажатии на кнопку **«Выход»**:

```
this.Close(); //закрытие текущей формы
```

14. Добавьте в проект новую форму:
    1) выполните команду **Проект – Добавить форму Windows**;
    2) в поле имя укажите **PreviewForm**;
    3) нажмите кнопку **OK**.
15. При нажатии на кнопку **«Печать бейджа»** откройте форму с предварительным просмотром:

```
//если все поля и имя файла изображения не являются пустыми значениями
if ((textBox1.Text != "")&&(textBox1.Text != "")&&(openFileDialog1.FileName != ""))
{
   //создаем форму предварительного просмотра
   PreviewForm previewFrm = new PreviewForm();
   //показываем эту форму в диалоговом режиме
   previewFrm.ShowDialog();
}
```

16. Запустите программу и проверьте правильность работы элементов управления.
17. Интерфейс **PreviewForm** сформируйте по аналогии с рисунком 2, используя для отображения текста компонент типа **Label**, а для отображения фотографии – **PaintBox**.

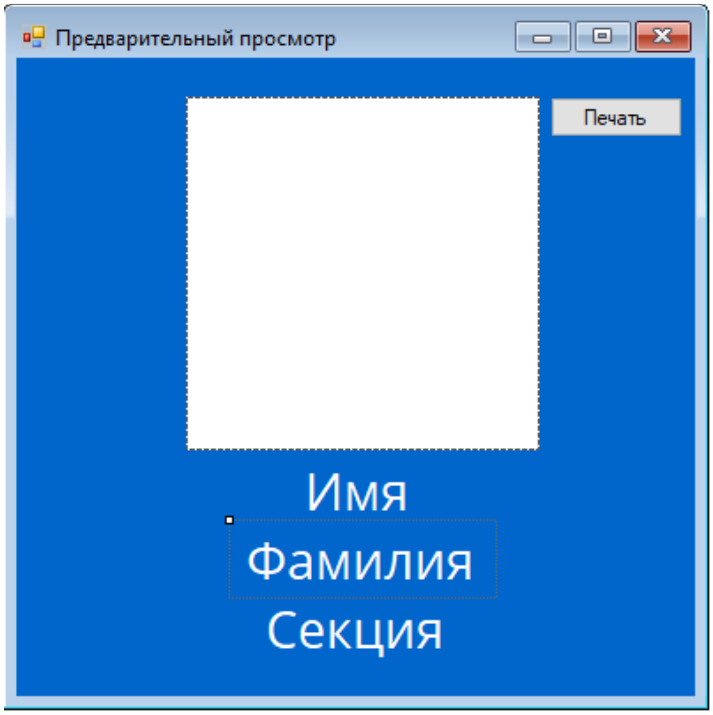


Рисунок 2. Интерфейс окна предварительного просмотра

18. Дополните программный код формы **PreviewForm** конструктором, принимающим значения полей в качестве параметров и отображающим их в соответствующие компоненты:

```
public PeviewForm(string name, string surname, string section, string filename)
{
   InitializeComponent();
   label1.Text = name.ToUpper();
   label2.Text = surname.ToUpper();
   label3.Text = section.ToUpper();
   pictureBox1.Load(filename);
}
```

19. При нажатии на кнопку **«Распечатать бейдж»** выполните показ новой формы:

```
//создаем форму предварительного просмотра
PreviewForm previewFrm = new PreviewForm(textBox1.Text, textBox2.Text,
   comboBox1.Text, openFileDialog1.FileName);
//показываем эту форму в диалоговом режиме
previewFrm.ShowDialog();
```

20. Запустите программу и проверьте правильность ее работы.
21. **Самостоятельно** выполните настройку компонентов окна **«Предварительный просмотр»** так, чтобы данные отображались корректно (не выходили за пределы формы и т.п.).
22. В классе формы **PreviewForm** объявите следующее поле:

```
private Bitmap memoryImage; //буфер для хранения изображения
```

23. Там же опишите метод **CaptureScreen()**, выполняющий копирование изображения формы в буфер:

```
private void CaptureScreen()
{
   Graphics myGraphics = this.CreateGraphics();
   Size s = this.ClientSize;
   memoryImage = new Bitmap(s.Width, s.Height, myGraphics);
   Graphics memoryGraphics = Graphics.FromImage(memoryImage);
   memoryGraphics.CopyFromScreen(this.Location.X, this.Location.Y, 0, 0, s);
}
```

24. Разместите на форме компонент типа **PrintDocument**, предназначенный для вывода документов на печать.
25. Создайте обработчик события **PrintPage** компонента **PrintDocument**:

```
e.Graphics.DrawImage(memoryImage, 0, 0); //печать изображения из буфера
```

26. Опишите программный код кнопки **«Печать»**:

```
 CaptureScreen();          //получение изображения формы
 printDocument1.Print();   //печать документа
```

27. Запустите программу и проверьте правильность ее работы.
28. **Самостоятельно** изучите основные приемы работы с компонентом **ToolTip** и дополните программу всплывающими подсказками.

## Часть II. Графический интерфейс пользователя

**Задание 2:** Разработайте приложение для компании по продаже керамической плитки, обеспечивающее расчет необходимого количества плитки и формирование предварительного просмотра сочетания плитки и декора.

1. Добавьте в решение **Лабораторная работа 3** новый проект **Windows Forms**.

Стилевое оформление всех окон приложения должно соответствовать единому формату. В Visual Studio существует возможность создания шаблонов форм, которые могут включать в себя как элементы интерфейса, так и их методы.

2. Создайте шаблон формы:
   1) через **Обозреватель решений** переименуйте класс формы **Form1** в **TemplateForm**;
   2) выполните настройку следующих свойств формы:

| Свойство | Значение | Описание |
|---|---|---|
| `Size` | `600;400` | Размер окна (ширина; длина) |
| `StartPosition` | `CenterScreen` | Расположение окна при открытии по центру экрана |
| `FormBorderStyle` | `None` | Отключение рамки окна |
| `BackColor` | `(на выбор)` | Цвет фона |
| `ForeColor` | `Черный` | Цвет надписей окна |
| `Font` | `Name = Verdana`<br>`Size = 14` | Шрифт (название и размер) |

   3) для создания строки заголовка окна разместите на форме компонент типа **Panel** и выполните настройку его свойств:

| Свойство | Значение | Описание |
|---|---|---|
| `Name` | `pnlBorder` | Имя объекта |
| `Dock` | `Top` | Привязка к верхнему краю формы |
| `Width` | `30` | Размер компонента по ширине |
| `BackColor` | `(на выбор)` | Цвет фона компонента |

   4) разместите на компоненте **pnlBorder** две кнопки (для закрытия и сворачивания окна) и последовательно назначьте для них следующие значения свойств:

| Свойство | Значение | Описание |
|---|---|---|
| `Name` | `1) btnClose`<br>`2) btnMinimize` | Кнопка «Закрыть окно»<br>Кнопка «Свернуть окно» |
| `Dock` | `Right` | Привязка к правому краю панели |
| `Size` | `30;30` | Размер компонента |
| `FlatStyle` | `Popup` | Элемент управления выглядит плоским, пока на него не наведен курсор мыши, когда на него наведен курсор мыши, элемент управления выглядит трехмерным. |
| `Text` | `1) X`<br>`2) _` | Отображаемый компонентом текст |

5) разместите на панели компонент типа **PictureBox** для отображения логотипа:

| Свойство | Значение | Описание |
|---|---|---|
| Name | imgLogo | Имя компонента |
| Dock | Left | Привязка к левому краю панели |
| Size | 30;30 | Размер компонента |
| Image | Logo.jpg | Отображаемое изображение из файла |
| SizeMode | Stretch | Изменение размера изображения в соответствии с размером компонента |

6) для визуализации заголовка на панель перенесите компонент типа **Label**:

| Свойство | Значение | Описание |
|---|---|---|
| Name | lblTitle | Имя компонента |
| Dock | Fill | Заполнение свободного пространства панели |
| AutoSize | False | Отключение автоматического изменения размера компонента по размеру текста |
| TextAlign | MiddleLeft | Выравнивание текста по центру и слева |
| Font - Size | 16 | Размер шрифта в пунктах |
| Text | Заголовок | Отображаемый компонентом текст |

7) при нажатии на кнопку **«Закрыть окно»** выполните соответствующее действие:

```
this.Close();
```

8) при нажатии на кнопку **«Свернуть окно»** измените размер окна до минимального:

```
this.WindowState = FormWindowState.Minimized;
```

3. Сравните полученный результат с примером на рисунке 3.

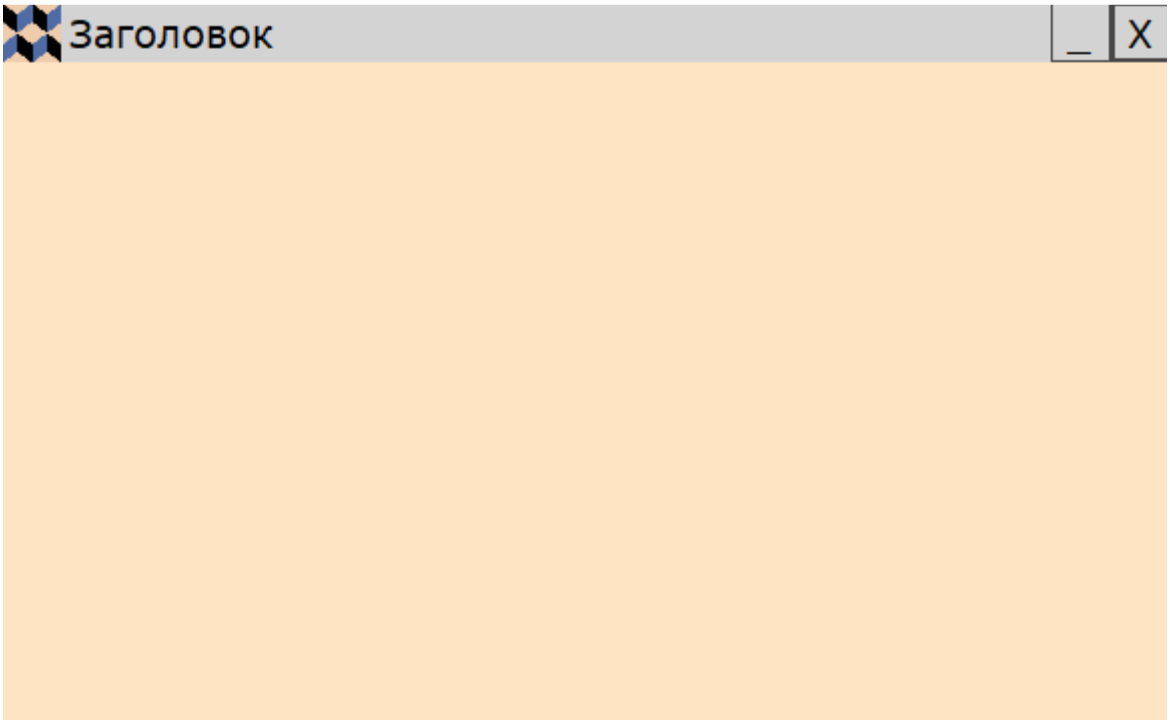


Рисунок 3. Пример интерфейса шаблона формы

4. Выполните экспорт шаблона формы с помощью **Мастера**:
   1) откройте пункт **Файл – Экспорт шаблона…**;
   2) выберите тип **Шаблон элемента**;
   3) выберите **Элемент для экспорта TemplateForm.cs**;
   4) на странице **Параметры шаблона** в качестве имени можете указать название проекта;
   5) остальные параметры оставьте без изменения;
   6) запомните каталог сохранения шаблона и скопируйте файл в личную папку.
5. Спроектируйте интерфейс главной формы:
   1) через **Обозреватель решений** переименуйте текущую форму в **MainForm**;
   2) измените заголовок окна на **«Главная форма»** в соответствующем компоненте;
   3) разместите на форме две кнопки: **«Конструктор»** и **«Калькулятор»**;
   4) установите в качестве фона для кнопок серый цвет.
6. Запустите программу и проверьте правильность отображения элементов управления на форме.

Окно **«Калькулятор»** предназначено для расчета количества плитки, необходимой для покрытия стены указанного размера. Существует несколько форматов плитки и коробки с различным количеством штук.

7. Создайте форму **«Калькулятор»**:
   1) выполните команду: **Проект – Добавить форму Windows…**;
   2) в качестве шаблона выберите созданный вами в п. 4 шаблон формы;
   3) укажите имя новой формы: **CalculateForm**;
   4) на главной форме при нажатии на кнопку «Калькулятор» обеспечьте создание и отображение соответствующей формы.
8. Спроектируйте каркас новой формы:
   1) разместите на форме компонент типа **TableLayoutPanel**;
   2) в свойстве **Columns** компонента создайте **4 столбца с шириной по 25%;**
   3) в свойстве **Rows** компонента создайте **5 строк с высотой по 20%;**
   4) с помощью свойства **Dock** растяните компонент по размеру формы.
9. Разместите элементы управления на форме:
   1) в верхнюю левую ячейку получившегося каркаса поместите текстовую метку **Label**;
   2) у метки установите следующие значения свойств:

| Свойство | Значение | Описание |
|---|---|---|
| `AutoSize` | `False` | Отключение автоматического изменения размера компонента по размеру текста |
| `Dock` | `Bottom` | Стыковка компонента к нижнему краю ячейки каркаса |
| `Column` | `0` | Номер столбца |
| `ColumnSpan` | `2` | Количество занимаемых столбцов |

| | | |
|---|---|---|
| TextAlign | MiddleCenter | Выравнивание текста по центру |
| Text | Параметры поверхности | Отображаемый компонентом текст |

3) аналогично разместите остальные элементы управления в соответствии с рисунком 4.

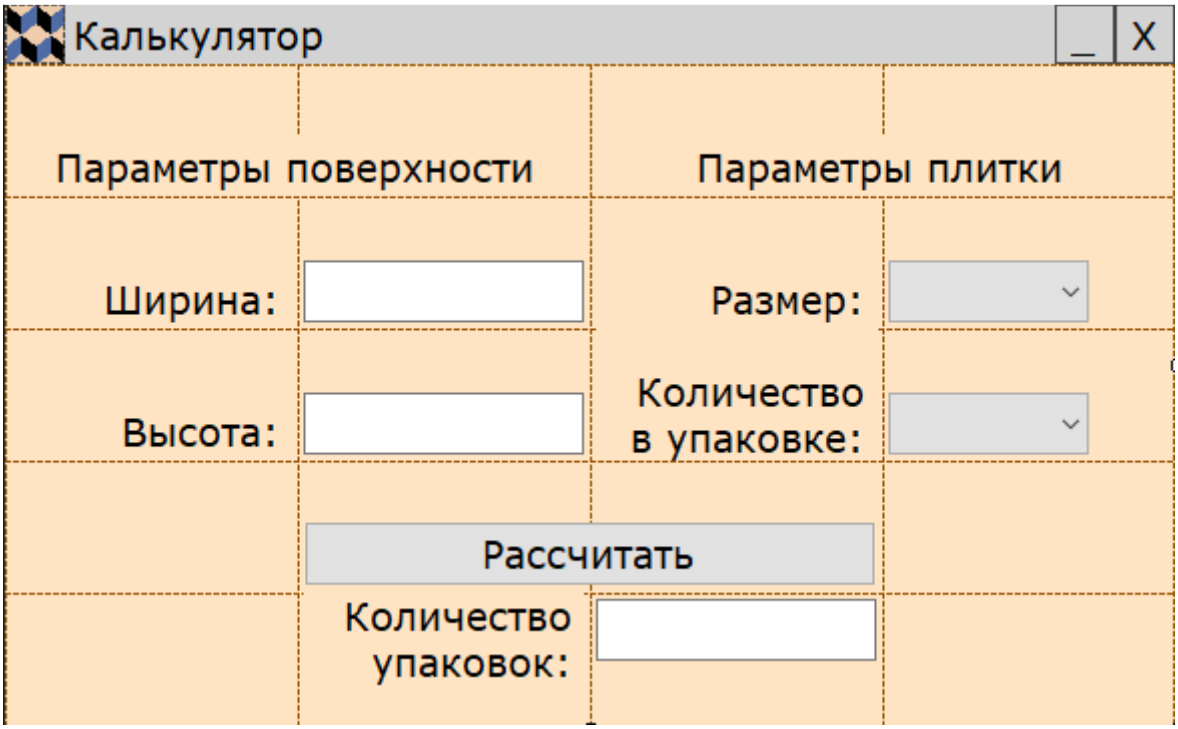


Рисунок 4. Пример размещения элементов управления окна "Калькулятор"

10. Для полей ввода **«Ширина»** и **«Высота»** определите ввод только цифровых символов.
11. Поле ввода **«Количество упаковок»** сделайте доступным только для чтения.
12. Для элементов-списков **«Размер»** и **«Количество в упаковке»** определите возможность выбора значения только из списка.
13. Заполните значения списков **«Размер»** и **«Количество в упаковке»:**
    1) в классе формы **CalculateForm** объявите и проинициализируйте переменные:

```
private static short n = 4;                  //количество значений
private short[,] tileSize = new short[n, 2]; //список возможных размеров плитки
private short[] tileCount = new short[n];    //список для выбора кол-ва плитки
```

    2) в конструкторе этой формы занесите в массивы возможные значения:

```
//заполнение списка "размер плитки" (ширина и высота)
tileSize[0, 0] = 10; tileSize[0, 1] = 10;

tileSize[1, 0] = 15; tileSize[1, 1] = 15;

tileSize[2, 0] = 10; tileSize[2, 1] = 15;

tileSize[3, 0] = 20; tileSize[3, 1] = 20;


//заполение списка "количество"
tileCount[0] = 8;
tileCount[1] = 10;
tileCount[2] = 12;
tileCount[3] = 20;
```

3) там же выполните заполнение компонентов-списков:

```
for (int i = 0; i < n; i++) //заполняем список "размеры плитки"
comboBox1.Items.Add(String.Format("{0}x{1}", tileSize[i, 0], tileSize[i, 1]));
for (int i = 0; i < n; i++) //заполняем список "количество"
comboBox2.Items.Add(String.Format("{0}", tileCount[i]));
```

4) запустите программу и проверьте правильность отображения и поведения элементов управления.

14. **Самостоятельно** при нажатии на кнопку **«Рассчитать»** выполните расчет требуемого количества плитки с округлением в большую сторону до целого числа.

Окно **«Конструктор»** предназначено для предварительного просмотра сочетания плитки и декора. В качестве фона может быть выбран один из вариантов плитки, количество размещенных элементов декора не ограничено. Образцы плитки и декора отображаются в списках, откуда они перетаскиваются на панель предварительного просмотра с помощью мыши.

15. Создайте новую форму **ConstructForm** на основе шаблона.
16. Спроектируйте интерфейс окна в соответствии с рисунком 5.

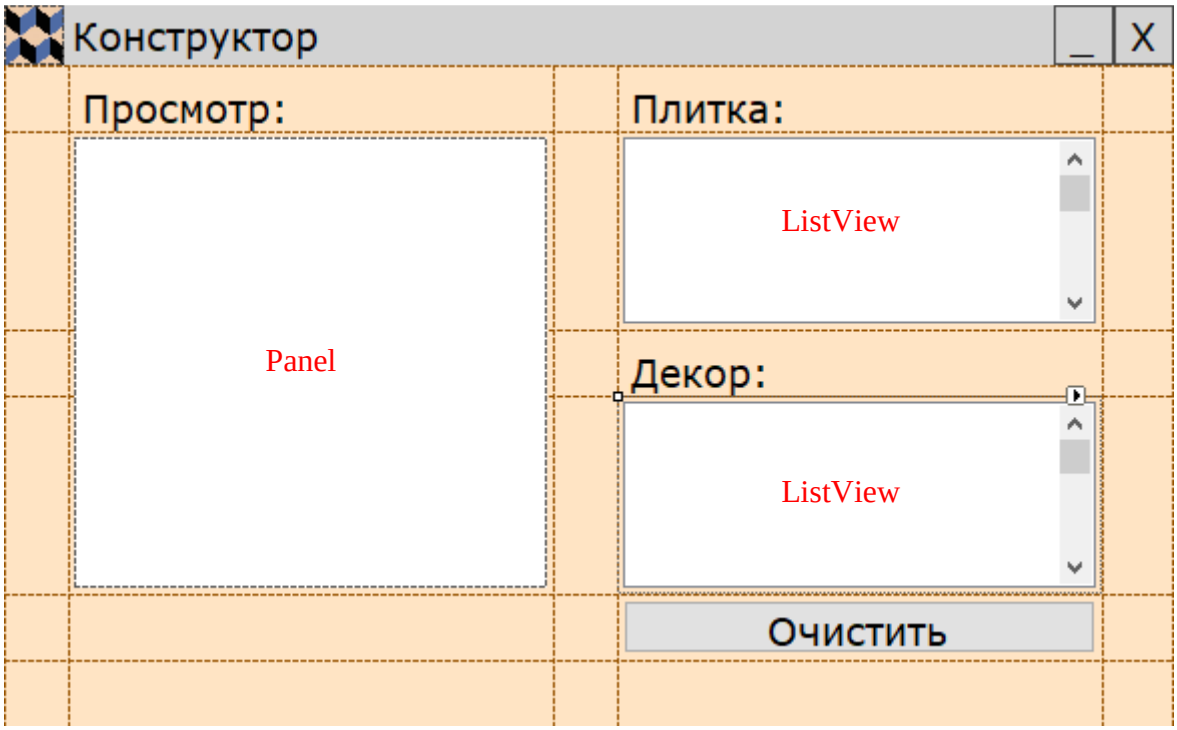


Рисунок 5. Примерный вид окна "Конструктор"

17. Добавьте на форму два невизуальных компонента типа **ImageList**: **imgListFlat** и **imgListDecor**.
18. В коллекцию изображений **imgListFlat** загрузите образцы плитки из каталога **Flat**:
    1) раскройте свойство **Images** компонента;
    2) нажмите кнопку **«Добавить»**;
    3) в каталоге **Flat** выберите все находящиеся там изображения.
    4) установите размер изображений в свойстве **ImageSize** равным 50×50 пикселей.
19. Аналогично загрузите образцы декора в компонент **imgListDecor** из каталога **Decor**.

20. Выполните настройку компонента типа **ListView**, предназначенного для отображения образцов плитки:
    1) в свойстве **LargeImageList** выберите имя **imgListFlat**;
    2) раскройте свойство **Items**;
    3) с помощью кнопки **«Добавить»** создайте элементы списка по количеству изображений плитки в соответствующем каталоге;
    4) для каждого элемента в свойстве **ImageIndex** выберите номер отображаемого изображения по порядку.
21. Аналогично настройте компонент, предназначенный для отображения образцов декора.
22. Для разрешения выполнения операции перетаскивания в компонент типа **Panel** установите у него свойство **AllowDrop** равным **True**.
23. В классе текущей формы объявите следующие переменные:

```
private int indexOfFlat;   //индекс переносимого пункта из списка Плитка
private bool flat = false; //флаг переноса изображения плитки
```

24. При щелчке мышью по списку с образцами плитки (в обработчике события **MouseDown**) выполните:

```
flat = true; //устанавливаем флаг переноса плитки

// Получаем индекс выбранного элемента
indexOfFlat = FlatList.Items.IndexOf(FlatList.GetItemAt(e.X, e.Y));

if (indexOfFlat != -1) //если удалось определить индекс
{
   //начинаем операцию перетаскивания элемента
   FlatList.DoDragDrop(FlatList.Items[indexOfFlat].ImageIndex,
   DragDropEffects.Copy);
}
```

25. Для компонента типа **Panel** определите событие **DragEnter**, возникающее при перемещении в его область другого элемента:

```
e.Effect = DragDropEffects.Copy; //устанавливаем режим перетаскивания
```

26. Выполните обработку завершения операции перетаскивания с помощью события **DragDrop** компонента **Panel**:

```
if (flat) //если выбрана операция перетаскивания изображения плитки
{
   //устанавливаем в качестве фонового изображения панели перетаскиваемое из списка
   pnlView.BackgroundImage = imgListFlat.Images[indexOfFlat];
}
```

27. Запустите программу и проверьте правильность выполнения операций.
28. Дополните класс текущей формы следующими переменными:

```
private int indexOfDecor;        //индекс переносимого пункта из списка Декор
private int sizeOfDecor = 50;    //размер изображения декора

//список размещенных элементов декора
private List<PictureBox> decors= new List <PictureBox>();
```

29. При щелчке мышью по списку с образцами декора выполните следующие действия:

```
flat = false; //сбрасываем флаг переноса плитки

//определяем индекс выбранного в списке элемента
indexOfDecor = DecorList.Items.IndexOf(DecorList.GetItemAt(e.X, e.Y));
//если удалось определить индекс
if (indexOfDecor != -1)
{
   //начинаем операцию перетаскивания элемента
   DecorList.DoDragDrop(DecorList.Items[indexOfDecor].ImageIndex,
   DragDropEffects.Copy);
}
```

30. Дополните событие **DragDrop** компонента **Panel** следующим программным кодом:

```
else      //если выбрана операция перетаскивания изображения декора
{
    decors.Add(new PictureBox()); //добавляем в список новый элемент декора

    //помещаем новый компонент в панель
    pnlView.Controls.Add(decors[decors.Count - 1]);

    //устанавливаем размер компонента
    decors[decors.Count - 1].Width = decors[decors.Count - 1].Height = sizeOfDecor;
    //устанавливаем изображение элемента
    decors[decors.Count - 1].Image = imgListDecor.Images[indexOfDecor];
    //подписываем компонент на событие отпускания клавиши мыши
    decors[decors.Count - 1].MouseUp +=
     new System.Windows.Forms.MouseEventHandler(this.pictureBox_MouseUp);
}
```

31. Добавьте в класс формы обработчик события, возникающего при перетаскивании декора в пределах панели:

```
private void pictureBox_MouseUp(object sender, MouseEventArgs e)
{
   //если была нажата левая кнопка мыши
   if (e.Button == MouseButtons.Left)
   {
         //создаем копию перетаскиваемого объекта
         PictureBox img = sender as PictureBox;

         //изменяем его положение на панели
         img.Left = img.Left + e.X - img.Width / 2;
         img.Top = img.Top + e.Y - img.Height / 2;
   }
}
```

32. Запустите программу и проверьте правильность ее работы.

33. **Самостоятельно** реализуйте:

1) перетаскивание окон с помощью мыши;
2) ограничение области перемещения декора по панели предварительного просмотра;
3) удаление одного из образцов декора, размещенного на панели предварительного просмотра;
4) очистку панели предварительного просмотра.

## Часть III. Задания для самостоятельной работы

**Задание:** Разработайте приложение с графическим интерфейсом пользователя в среде MS Visual Studio на языке C# для решения задачи согласно своему варианту.

**В программе должны быть предусмотрены:**

1) главная форма с кнопками для перехода на формы добавления и просмотра информации;
2) добавление информации в экранной форме;
3) просмотр информации в многострочном текстовом поле;
4) проверка корректности ввода данных;
5) обработка возможных исключительных ситуаций;
6) комментарии к основным блокам и переменным.

**Вариант 1:** В двумерном массиве хранятся данные о товарах на мебельном складе: название, тип, производитель. Реализуйте следующий функционал: добавление новой позиции на склад, вывод полного списка товаров, вывод на экран и подсчет количества позиций выбранного пользователем типа мебели, поиск и отображение сведений о товаре по введенному пользователем названию.

Входные данные:

| Поле | Значение | Пример значения |
|---|---|---|
| Название | Уникальное | Комфорт |
| Тип | Может повторяться | Диван |
| Производитель | Может повторяться | ОАО «Диваны и кресла» |

**Вариант 2:** В двумерном массиве хранятся данные об абитуриентах колледжа: количество баллов по русскому языку, математике и код специальности. Обеспечьте следующий функционал: добавление новой записи, вывод полного списка с указанием суммы баллов по двум предметам, поиск и отображение списка отличников (имеющий по 100 баллов за каждый предмет), вывод среднего балла абитуриентов по выбранной пользователем специальности.

Входные данные:

| Поле | Значение | Пример значения |
|---|---|---|
| Русский язык | 0 – 100, может повторяться | 75 |
| Математика | 0 – 100, может повторяться | 65 |
| Код специальности | 1 – 10, может повторяться | 2 |

**Вариант 3:** В двумерном массиве хранится информация о сотрудниках компании: ФИО, специальность и подразделение. Обеспечьте следующий функционал: добавление новой записи, вывод полного списка сотрудников, поиск и отображение информации о сотруднике по фамилии, вывод и подсчет количества сотрудников выбранного пользователем подразделения.

 

Входные данные:

| Поле | Значение | Пример значения |
|---|---|---|
| ФИО | Если повторяется, добавить номер | Иванов Иван Иванович 2 |
| Должность | Может повторяться | Продавец |
| Подразделение | Может повторяться | Отдел продаж |

**Вариант 4:** В двумерном массиве хранится информация о книгах домашней библиотеки: название, автор, жанр. Обеспечьте следующий функционал: добавление новой записи, вывод полного списка книг, поиск по названию и автору (одновременно), вывод на экран и подсчет количества книг выбранного пользователем жанра.

Входные данные:

| Поле | Значение | Пример значения |
|---|---|---|
| Название | Может повторяться | Портрет Дориана Грея |
| Автор | Может повторяться | Оскар Уайльд |
| Жанр | Может повторяться | Роман |

**Вариант 5:** В двумерном массиве хранится информация о поставках товаров на овощную базу: номер склада, количество килограммов моркови, количество килограммов картофеля. Обеспечьте следующий функционал: добавление новой записи, вывод полного списка; вывод на экран позиций, находящихся на выбранном пользователем складе; вывод на экран номеров складов, на которых общее количество товара меньше введенного пользователем значения.

Входные данные:

| Поле | Значение | Пример значения |
|---|---|---|
| Код склада | 1-5, может повторяться | 3 |
| Количество моркови | Целое число (кг) | 100 |
| Количество картофеля | Целое число (кг) | 52 |

**Вариант 6:** В двумерном массиве хранятся результаты олимпиады по информатике: код участника, количество баллов за тест, количество баллов за практическое задание. Обеспечьте следующий функционал: добавление новой записи; вывод полного списка участников; поиск и вывод кодов участника(-ов), набравших наибольшее количество баллов; подсчет и вывод количества участников, не справившихся с тестом, практическим заданием или обоими заданиями.

Входные данные:

| Поле | Значение | Пример значения |
|---|---|---|
| Код участника | Целое уникальное число | 6 |
| Баллы за тест | 0 – 100, может повторяться | 68 |
| Баллы за практическое задание | 0 – 10, может повторяться | 8 |

**Вариант 7:** В двумерном массиве хранятся данные об участниках IT-конференции: ФИО, город, название секции. Обеспечьте следующий функционал: добавление новой записи; вывод полного списка участников; вывод списка участников из выбранного пользователем города; подсчет количества участников в секции, указанной пользователем.

Входные данные:

| Поле | Значение | Пример значения |
|---|---|---|
| ФИО | Если повторяется, добавить номер | Петров Петр Петрович 3 |
| Город | Может повторяться | Новосибирск |
| Секция | Может повторяться | Облачные технологии |

**Вариант 8:** В двумерном массиве хранятся данные о пользователях информационной системы. Обеспечьте следующий функционал: добавление новой записи, вывод полного списка пользователей, поиск и вывод информации о пользователе по значению логина, вывод на экран общего количества пользователей указанного пользователем типа учетной записи.

Входные данные:

| Поле | Значение | Пример значения |
|---|---|---|
| ФИО | Может повторяться | Васильев Василий Васильевич |
| Логин | Уникальное значение | Vasya |
| Пароль | 6 символов, может повторяться | 123456 |
| Тип учетной записи | Может повторяться | Администратор |