**Лабораторная работа № 7. Разработка приложений**

**с многодокументным интерфейсом**

**Цель работы:** изучить принципы работы с MDI и получить практические навыки разработки многооконных (многодокументных) интерфейсов.

**Краткая теория**

**Multiple document interface** (или **MDI**) — представляет собой [окна](http://ru.wikipedia.org/wiki/%D0%9E%D0%BA%D0%BE%D0%BD%D0%BD%D1%8B%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81), расположенные под одним общим окном (как правило, за исключением [модальных окон](http://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%BE%D0%BA%D0%BD%D0%BE)), в отличие от окон, расположенных отдельно друг от друга ([SDI](http://ru.wikipedia.org/wiki/Single_document_interface)). Другими словами MDI-приложения позволяют отображать одновременно несколько документов. При этом каждый документ будет отображаться в своём собственном окне. Как правило MDI-приложения имеют в основном меню подпункты для переключения между окнами и документами.

Механизм работы с MDI-приложениями немного сложнее, чем обычных приложений, которые основываются на диалогах. Основным окном MDI-приложения является **родительская форма**. Она может содержать несколько **дочерних окон**. Только одно из дочерних окон может быть активно в один момент времени.

**Создание родительской формы**

***Создание главного окна MDI-приложения***

Главное окно приложения MDI и отдельные MDI-окна связывают родительские отношения. При этом главное окно MDI-приложения является родительским для дочерних MDI-окон. В этом разделе мы рассмотрим процедуру создания главного окна приложения MDI средствами Microsoft Visual Studio .NET на языке C#.

Для демонстрации способов создания приложений MDI мы создадим программу MDI.

Вначале необходимо создать новый проект приложения Windows Application Project, как обычно. Далее откройте окно редактирования свойств главного окна (формы) приложения и установите значение свойства IsMDIContainer, равным True (см. рисунок 1).

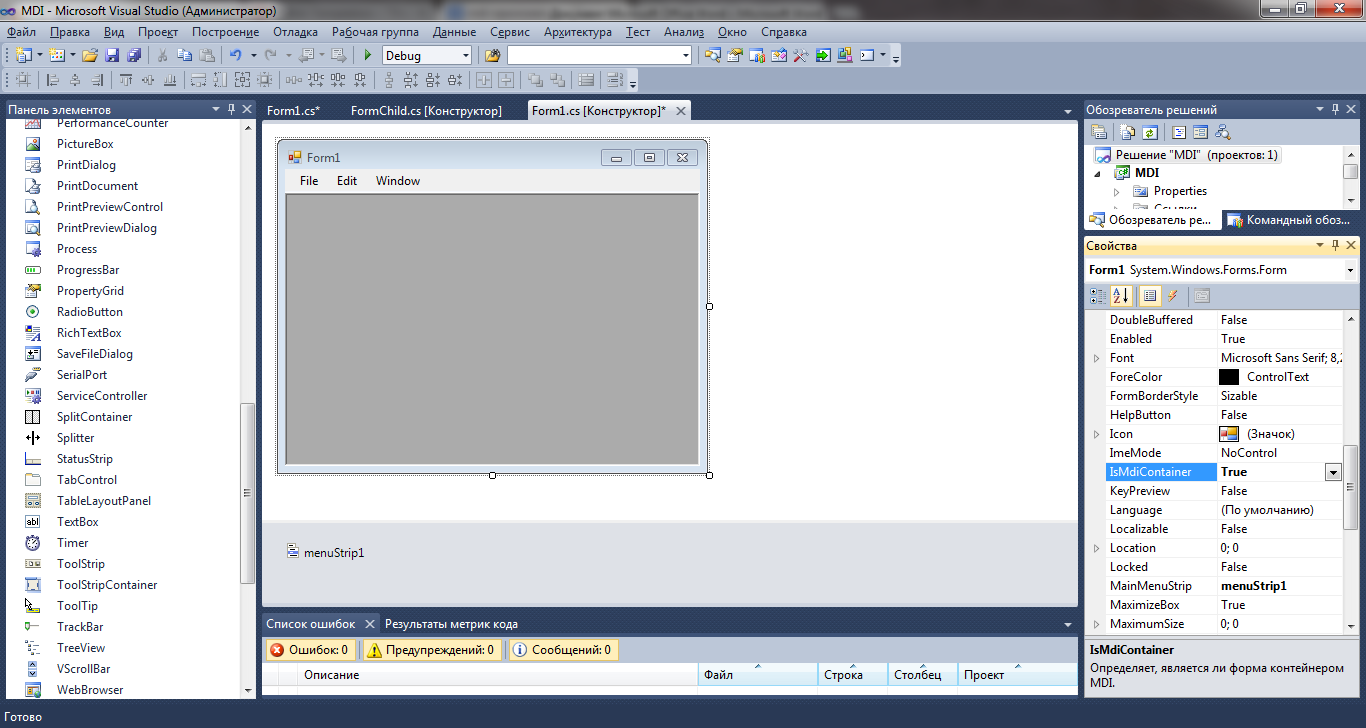


Рисунок 1 – Установка свойства IsMDIContainer

В результате главное окно приложения превратится в контейнер для дочерних MDI-окон. Назовём его ParentForm.

Далее, добавьте в приложение главное меню. Для этого перетащите из панели Toolbox системы Microsoft Visual Studio.NET значок главного меню MenuStrip.

Создайте меню File со строками New, Close и Exit, как это показано на рисунке 2.

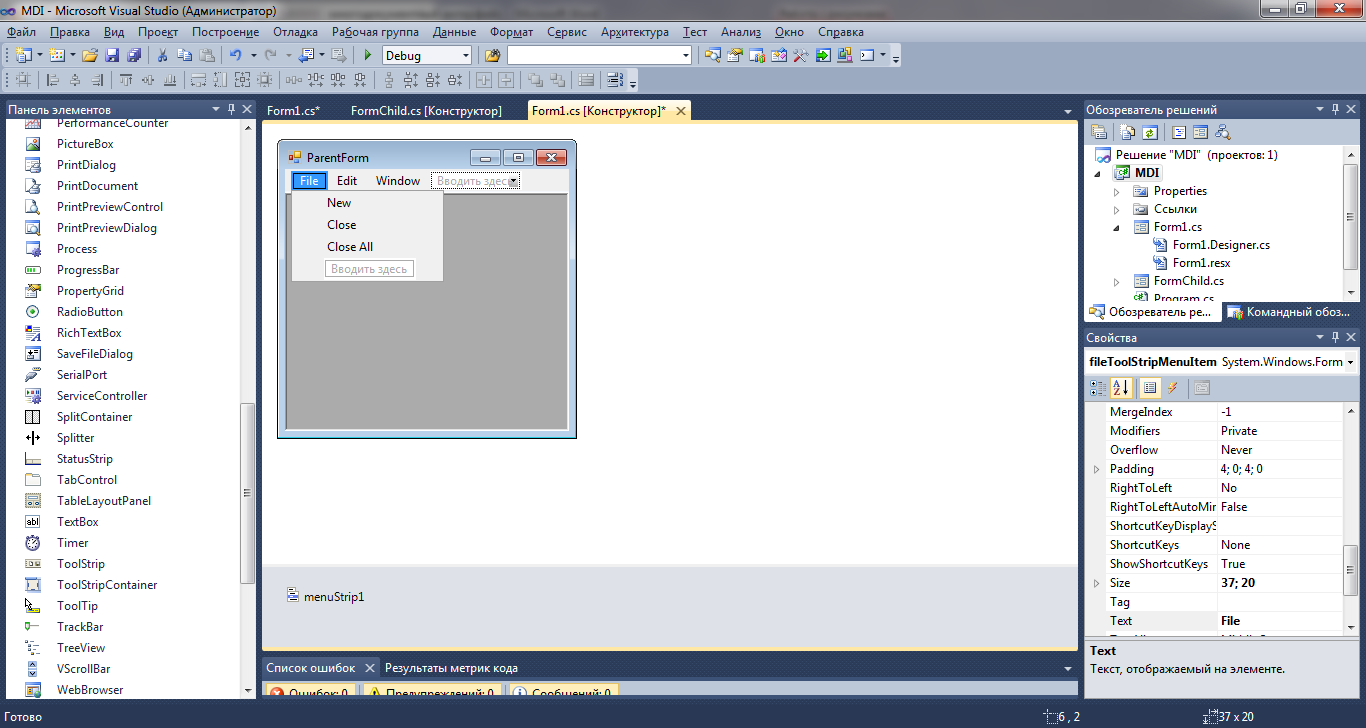


Рисунок 2 – Меню File

С помощью строки New этого меню мы будем создавать новые MDI-окна с редактором текста на базе элемента управления RichTextBox. Строка Close предназначена для закрытия MDI-окон, а строка Exit — для завершения работы приложения.

Создайте также меню Edit со строками Copy и Paste (см. рисунок 3), а также меню Window,  предназначенное для управления MDI-окнами и показанное на рисунке 4.

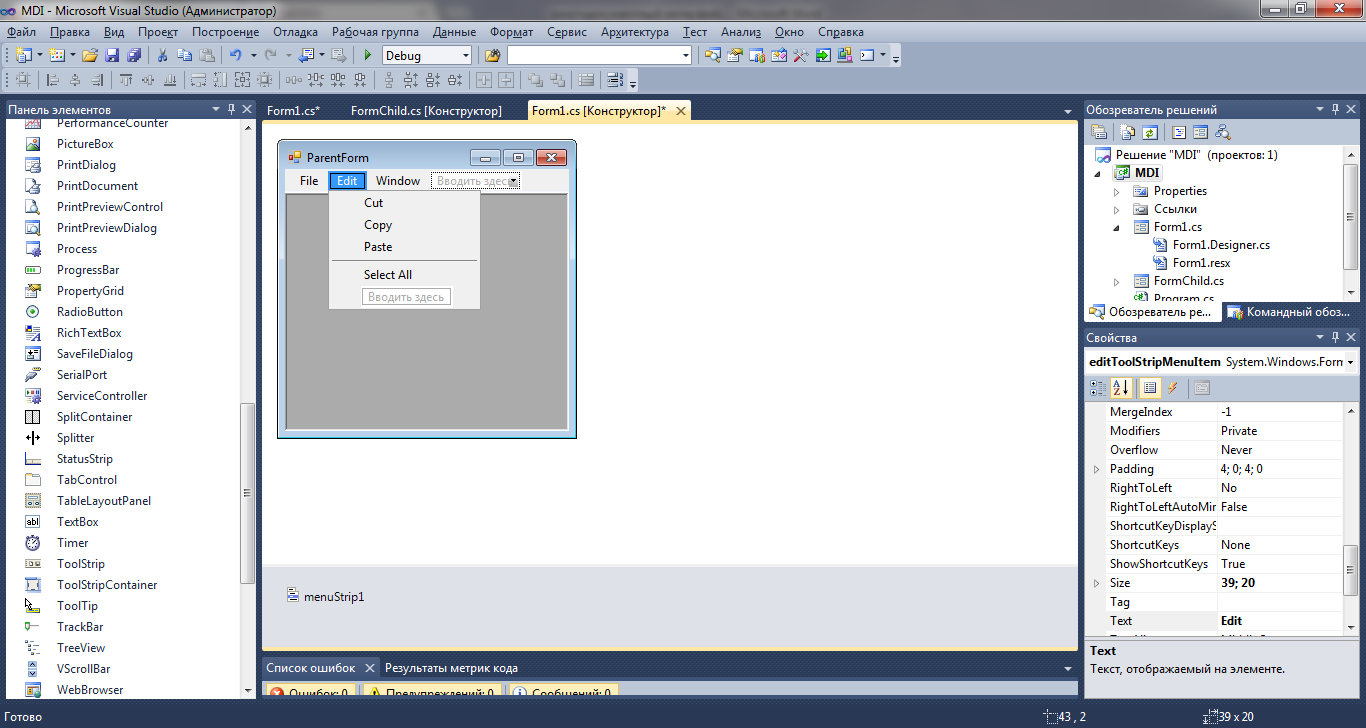


Рисунок 3 – Меню Edit.

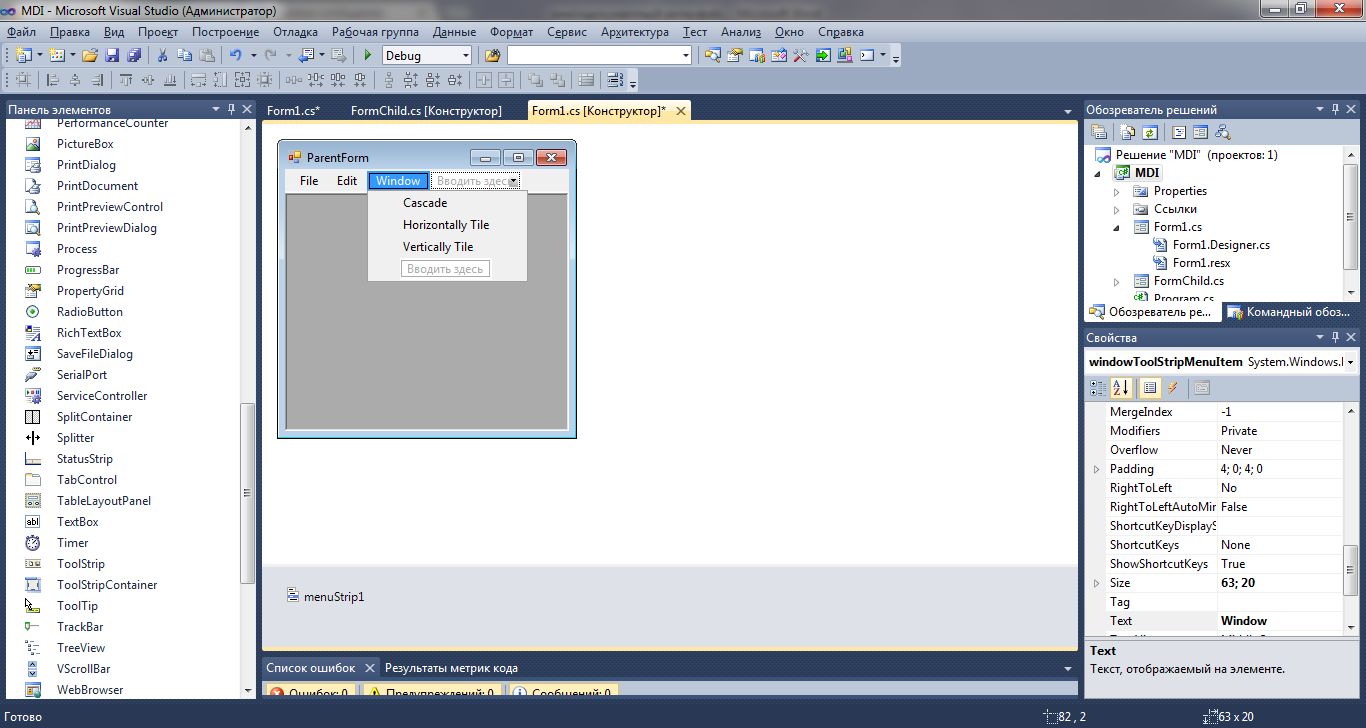


Рисунок 4 – Меню Window

Строка Cascade предназначена для каскадного расположения открытых MDI-окон, а строки Horizontally Tile и Vertically Tile — для размещения окон рядом друг с другом в горизонтальном и вертикальном направлении, соответственно.

***Создание MDI-окон***

MDI-окна обычно используются для отображения различных документов или различных представлений одного и того же документа. В нашем приложении MDI MDI-окна будут содержать внутри себя элемент управления RichTextBox, т.е. редактор текста.

***Шаблон MDI-окна***

Прежде всего, нужно добавить в проект приложения новую форму, которая будет играть роль шаблона для создания дочерних MDI-окон. Для добавления формы щелкните правой клавишей мыши в окне Solution Explorer (обозреватель решений) название проекта MDI и выберите в контекстном меню Add (Добавить) строку Add Windows Form (добавить форму Windows).

После этого на экране появится диалоговое окно Add New Item,  в котором нужно выбрать значок Windows Form и затем нажать кнопку Open. В результате в окне дизайнера форм появится новая форма Form2. Для удобства переименуем её в FormChild.

Так как это окно будет предназначено для редактирования текста, перетащите в него из панели Toolbox значок элемента управления RichTextBox. Чтобы окно редактора текста RichTextBox заполнило собой клиентскую часть MDI-окна, установите значение свойства Dock равным Fill. Кроме того, проследите, чтобы свойство Anchor было равно Top, Left. Установка значений этого свойства выполняется при помощи редактора, окно которого показано на рисунке 5.

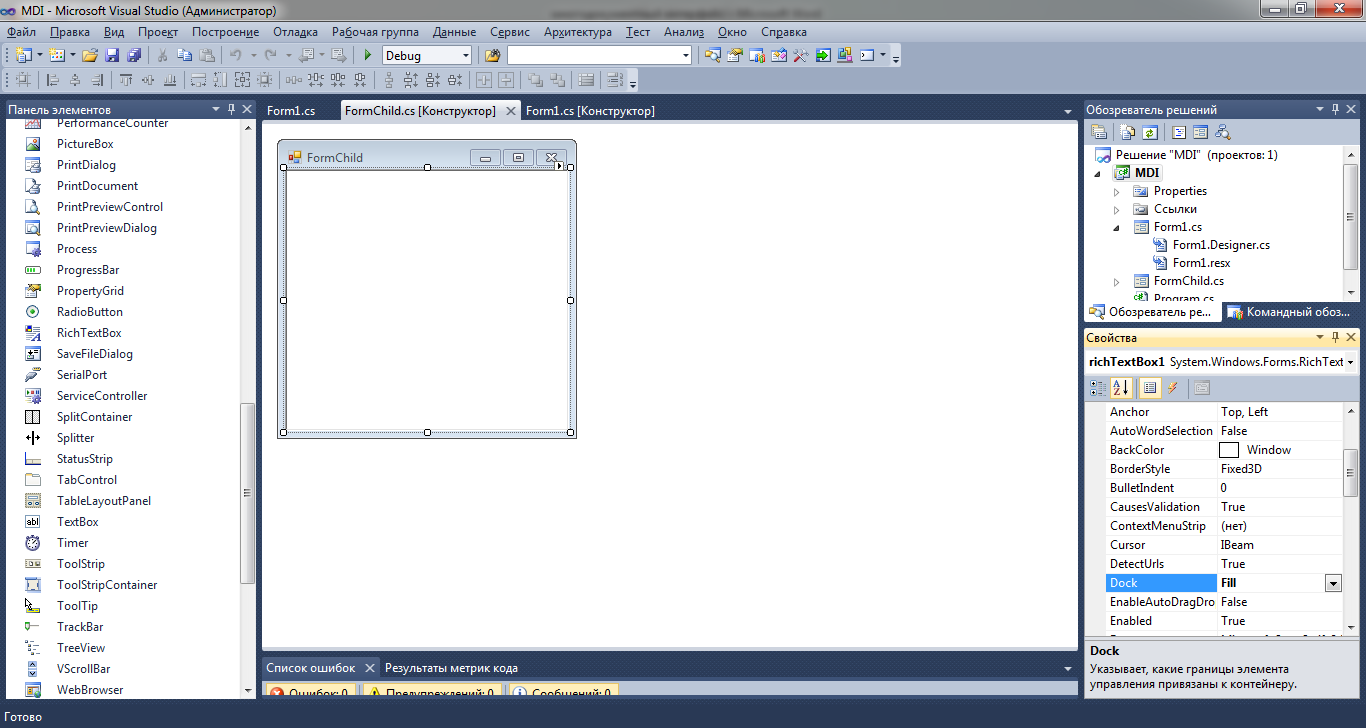


Рисунок 5 – Установка свойства Dock

Если щелкнуть левую и верхнюю кнопку окна редактирования этого свойства, а также установить правильным образом значение свойства Dock, то будет получен необходимый нам результат. А именно, окно элемента управления RichTextBox будет заполнять клиентскую область MDI-окна при любом изменении размеров последнего.

**Программный код для создания MDI-окна**

Теперь нам нужно написать программный код, создающий MDI-окна. Этот код должен получать управление, когда пользователь выбирает строку New из меню **File**.

Добавьте следующий обработчик события Click для упомянутой выше строки New меню File:

private void newToolStripMenuItem\_Click(object sender, EventArgs e)

{

FormChild fc = new FormChild();

fc.MdiParent = this;

fc.Show();

}

Здесь мы сначала создаем новую форму как объект класса Form2, а затем сохраняем ссылку на эту форму в переменной fc. Свойство MdiParent этого окна должно содержать ссылку на родительское окно приложения MDI, поэтому мы записываем в него ссылку на объект класса Form1, используя ключевое свойство this.

Для того чтобы MDI-окно появилось на экране, его необходимо отобразить явным образом при помощи метода Show.

На рисунке 6 мы показали главное окно нашего приложения, в котором было создано три MDI-окна.

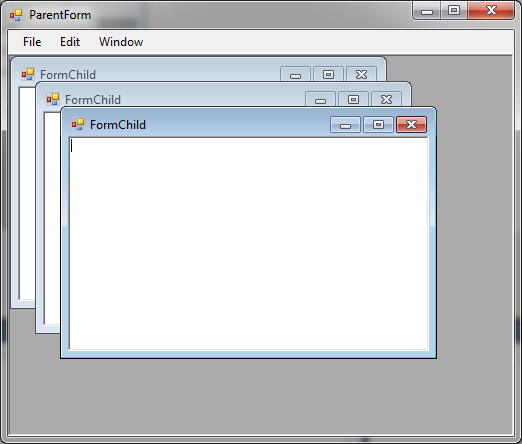


Рисунок 6 – Создано три MDI-окна

**Закрытие активного MDI-окна.**

Далее нам необходимо написать программный код, закрывающий активное MDI-окно. Этот код должен получать управление, когда пользователь выбирает строку Close из меню **File**.

Добавьте следующий обработчик события Click для строки Close меню File:

private void closeToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (ActiveMdiChild != null)

{

ActiveMdiChild.Close();

}

}

**Закрытие всех MDI-окон.**

Следующим шагом нам требуется написать программный код, создающий MDI-окна. Этот код должен получать управление, когда пользователь выбирает строку Close All из меню **File**.

Добавьте следующий обработчик события Click для строки Close All меню File:

private void closeAllToolStripMenuItem\_Click(object sender, EventArgs e)

{

foreach (Form form in this.MdiChildren)

{

form.Close();

}

}

**Упорядочивание MDI-окон**

Средства управления дочерними MDI-окнами позволяют легко упорядочить эти окна одним из трех способов.

Прежде всего, окна могут быть упорядочены с перекрытием (каскадным образом), как  это показано на рисунке 7. Кроме этого, возможно расположение окон рядом по горизонтали (см. рисунок 8) и по вертикали (см. рисунок 9).

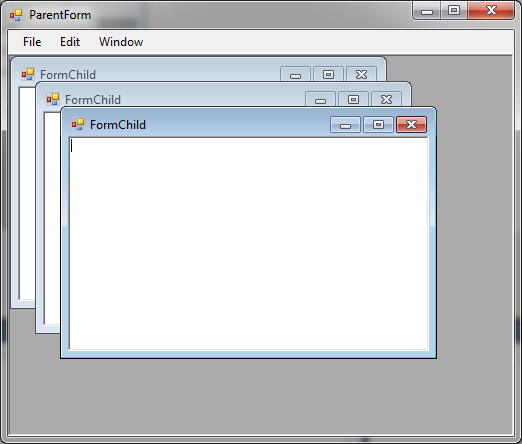


Рисунок 7 – Упорядочивание каскадом (типа Cascade)

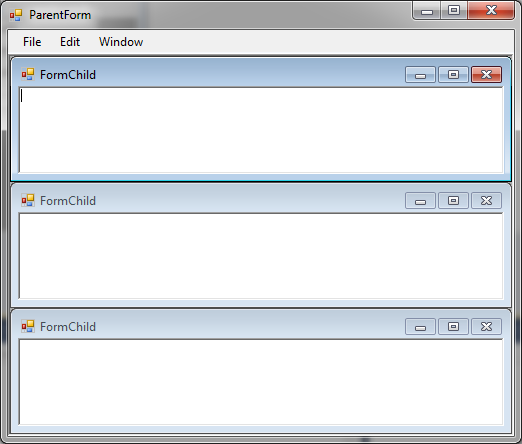
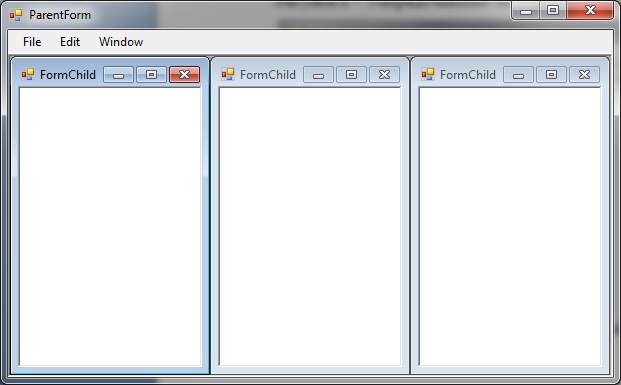


Рисунок 8 – Упорядочивание по горизонтали (типа Horizontally Tile)

Рисунок 9 – Упорядочивание по вертикали (типа Vertically Tile)

Чтобы пользователь мог упорядочивать дочерние MDI-окна, мы предусмотрели в меню Window строки Cascade,Horizontally Tile и Vertically Tile. Вот обработчики событий от этих строк меню, которые Вам необходимо добавить в исходный текст приложения:

private void cascadeToolStripMenuItem\_Click(object sender, EventArgs e)

{

this.LayoutMdi(MdiLayout.Cascade);

}

private void horizontallyToolStripMenuItem\_Click(object sender, EventArgs e)

{

this.LayoutMdi(MdiLayout.TileHorizontal);

}

private void verticallyTitleToolStripMenuItem\_Click(object sender, EventArgs e)

{

this.LayoutMdi(MdiLayout.TileVertical);

}

Как видите, все они вызывают метод LayoutMdi родительской формы, передавая ему в качестве параметра одну из констант, определяющих нужный тип упорядочивания. Это константы MdiLayout.Cascade, MdiLayout.TileHorizontal и MdiLayout.TileVertical.

В большинстве случаев для работы с документами, отображаемыми в MDI-окнах, используется одно меню — главное меню приложений. Например, с помощью строк New и Close меню File пользователь нашего приложения MDI может, соответственно, создавать и уничтожать MDI-окна.

Теперь мы реализуем операцию обмена данных через универсальный буфер обмена Clipboard, для которого в нашем приложении предусмотрено меню Edit. Строка Copy этого меню должна копировать данные из активного MDI-окна в буфер Clipboard, а строка Paste — вставлять данные из буфера Clipboard в активное MDI-окно. Активным окном здесь называетсяMDI-окно, имеющее фокус ввода.

**Меню Edit.**

Для реализации обработчиков событий Cut, Copy, Paste, Select All меню Edit необходимо создать метод GetRichTextBox.

private RichTextBox GetRichTextBox()

{

Form activeChild = this.ActiveMdiChild;

RichTextBox none = null;

if (activeChild != null) {

try

{

RichTextBox theBrox = (RichTextBox)activeChild.ActiveControl;

if (theBrox != null)

{

return theBrox;

}

}

catch {

MessageBox.Show("Что-то пошло не так");

}

}

return none;

}

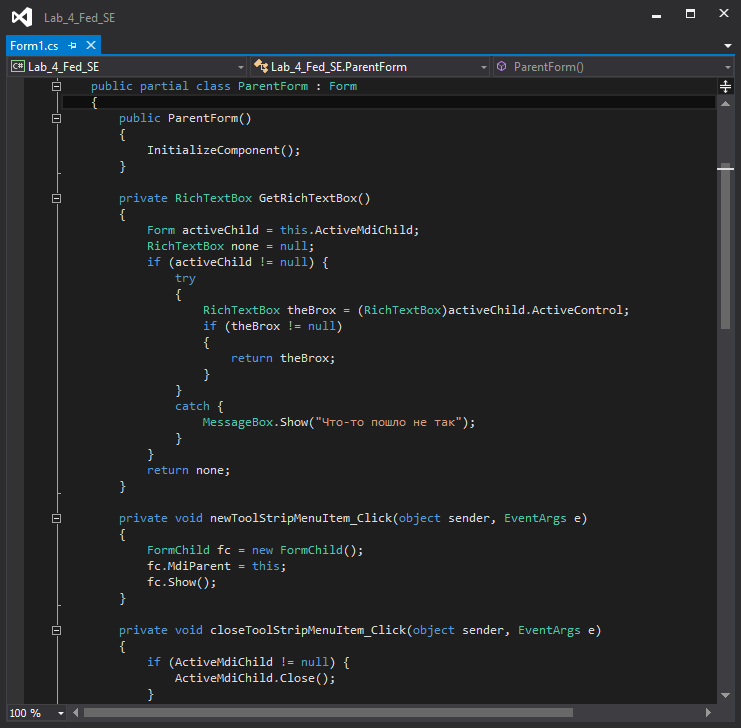


Рисунок 10 – Вид главного окна редактора

**Вырезать выделенные данные.**

Вначале мы реализуем операцию вырезания выделенного фрагмента текста из активного MDI-окна. Для этого добавьте следующий обработчик событий к строке Cut меню Edit:

private void cutToolStripMenuItem\_Click(object sender, EventArgs e)

{

GetRichTextBox().Cut();

}

**Копирование данных.**

Далее реализуем операцию копирования выделенного фрагмента текста из активного MDI-окна. Для этого добавьте следующий обработчик событий к строке Copy меню Edit:

private void copyToolStripMenuItem\_Click(object sender, EventArgs e)

{

GetRichTextBox().Copy();

}

**Вставка данных.**

Теперь дополним приложение кодом, необходимым для вставки данных из универсального буфера обмена Clipboard в активное MDI-окно.

Добавьте следующий обработчик события для строки Paste меню Edit:

private void pasteToolStripMenuItem\_Click(object sender, EventArgs e)

{

GetRichTextBox().Paste();

}

**Выделение данных.**

Следующим шагом реализуем выделение всего текста активного MDI-окна. Для этого необходимо добавить следующий обработчик событий к строке Select All в  меню Edit:

private void selectAllToolStripMenuItem\_Click(object sender, EventArgs e)

{

GetRichTextBox().SelectAll();

}

**4. Варианты заданий для самостоятельного решения**

В каждом варианте в главном окне необходимо создать меню и панель инструментов (компонент ToolStrip) с кнопками, дублирующими пункты меню.

1. Дан список фамилий (загружается из файла при создании дочернего окна). Организовать отбор во всех списках (во всех дочерних окнах) по первой букве фамилий. Результаты отображать в виде списка в отдельном окне (не дочернем).

2. Создать простейший текстовый редактор с возможностью открытия и сохранения файлов.

3. Создайте приложение «Секундомер». При создании дочернего окна должен запускаться секундомер. В окне должна быть кнопка останова, паузы и нового запуска.

4. Создать простейшее приложение для просмотра графических файлов с возможностью открытия и сохранения изображений.

5. Создать следующее приложение. В каждом дочернем окне – текстовая метка или поле ввода, которые работают как числовой счетчик. Скорость увеличения задается таймером. Скорость таймера можно менять одновременно для всех дочерних окон в отдельном диалоговом окне (не дочернем).

6. Создать приложение с двумя видами шаблонов дочерних окон. В первом случае должен осуществляться контроль ввода только цифровых данных, во втором – только текстовых. Заголовок должен содержать слово «Цифры» или «Символы».

7. Создайте приложение «Конверт». Шаблон дочернего окна должен представлять собой образец заполнения полей на почтовом конверте: в левом верхнем углу адрес отправителя, внизу справа адрес получателя. В заголовке окна должен отображаться получатель.

8. Создайте приложение «Статистика текста». При наборе текста с клавиатуры должен проводиться подсчет и осуществляться вывод количества знаков препинания, букв и цифр.

9. Создать приложение, где в каждом дочернем окне изображается какой-то графический объект (круг, прямоугольник и т.п.). Предусмотреть возможность изменения цвета линии этого объекта.

10. Создать приложение, где в каждом дочернем окне изображается какой-то графический объект (круг, прямоугольник и т.п.). Предусмотреть возможность изменения цвета линии этого объекта с помощью задания интенсивности красного, зеленого и синего цветов (модель RGB) через компоненты **TrackBar**.

11. Создать приложение, где в каждом дочернем окне изображается какой-то графический объект (круг, прямоугольник и т.п.). Предусмотреть возможность изменения стиля линии этого объекта (для этого у пера нужно изменять свойство **DashStyle**).

12. Создать приложение, где в каждом дочернем окне изображается какой-то графический объект (круг, прямоугольник и т.п.). Предусмотреть возможность изменения толщины линии этого объекта.

13. Создать приложение, где в каждом дочернем окне изображается какой-то графический объект (круг, прямоугольник и т.п.). Предусмотреть возможность изменения цвета заливки этого объекта.

14. Создать приложение, где в каждом дочернем окне изображается какой-то графический объект (круг, прямоугольник и т.п.). Предусмотреть возможность изменения цвета заливки этого объекта с помощью задания интенсивности красного, зеленого и синего цветов (модель RGB) через компоненты **TrackBar**.

15. Создать приложение, где в каждом дочернем окне изображается какой-то графический объект (круг, прямоугольник и т.п.). Предусмотреть возможность изменения стиля заливки этого объекта.

16. Создать приложение, где в каждом дочернем окне изображается движущийся графический объект (круг, прямоугольник и т.п.).

17. Разработать приложение для ввода результатов сессии. Организовать табличный ввод, используя компонент **DataGridView**. Предусмотреть запись результатов в текстовый файл.

18. Разработать приложение «Табулирование функций». Программа позволяет получить значения аргумента и функции в заданном интервале с заданным шагом. Вид функции можно задать выбором в компоненте **ListBox** или в компонентах **RadioButton**.

19. Разработать приложение, где в каждом дочернем окне должен вводиться пароль. Если пароль введен верно, то отображается какое-нибудь графическое изображение.

20. Разработать следующее приложение. В каждом дочернем окне – компонент-счетчик. В главном меню есть пункт, по нажатию которого открывается диалоговое окно, где отображается сумма чисел из счетчиков во всех открытых дочерних окнах.

21. Создать приложение, где в каждом дочернем окне изображается какой-то графический объект (круг, прямоугольник и т.п.). Предусмотреть возможность изменения координат этого объекта.

22. Создать приложение, где в каждом дочернем окне изображается какой-то графический объект (круг, прямоугольник и т.п.). Предусмотреть возможность изменения размера этого объекта.

23. Создать приложение, где в каждом дочернем окне изображается какой-то графический объект (круг, прямоугольник и т.п.). Предусмотреть возможность поворота этого объекта (влево на 90°, вправо на 90°, на 180°).

24. Создать простейший текстовый редактор с возможностью изменения шрифта и цвета текста.

25. Разработать следующее приложение. В каждом дочернем окне выбирается дата с помощью компонента **DateTimePicker**. В главном меню есть пункт, по нажатию которого открывается диалоговое окно, где выводится самая ранняя из всех дат, заданных в открытых дочерних окнах.

26. Разработать следующее приложение. В диалоговом окне (не дочернем) вводится текстовая строка. Программа должна разбить эту строку на слова и каждое слово передать в отдельное дочернее окно, например, в компонент **Label** или **TextBox** (должно открыться столько окон, сколько слов в строке).

27. Разработать следующее приложение. В диалоговом окне (не дочернем) вводится матрица чисел (в компонент **DataGridView**). Программа должна разбить эту матрицу на строки и каждую строку отобразить в отдельном дочернем окне (должно открыться столько окон, сколько строк в матрице).

28. Разработать следующее приложение. В диалоговом окне (не дочернем) вводится матрица чисел (в компонент **DataGridView**). Программа должна разбить эту матрицу на столбцы и каждый столбец отобразить в отдельном дочернем окне (должно открыться столько окон, сколько столбцов в матрице).

29. Разработать следующее приложение. В каждом дочернем окне вводится одно и то же количество чисел (например, в компонент **DataGridView** или **ListBox**). Необходимо все введенные числа из каждого дочернего окна объединить по строкам в матрицу, матрицу отобразить в отдельном диалоговом окне (не дочернем).

30. Разработать следующее приложение. В каждом дочернем окне вводится одно и то же количество чисел (например, в компонент **DataGridView** или **ListBox**). Необходимо все введенные числа из каждого дочернего окна объединить по столбцам в матрицу, матрицу отобразить в отдельном диалоговом окне (не дочернем).