Отчёт  
по самостоятельной работе №3

по дисциплине: «ОП.04 Основы алгоритмизации и программирования»

Тема: «Разработка оконного приложения с несколькими формами»

|  |  |  |
| --- | --- | --- |
| Студент гр. 13 ИТ |  | Липинский К.С. |
| Преподаватель |  | Счастливцев А.К. |

### Цель работы.

Обобщить знания по управляющим элементам, научиться различать модальные и немодальные окна, получить практические навыки создания MDI приложений.

### Теоретические сведения.

Рассматривая оконные приложения, следует различать следующие интерфейсы:

* **SDI (Single Document Interface** - однодокументный интерфейс)
* **MDI (Multi Document Interface** - многодокументный интерфейс).

***MDI-приложения*** работают с шаблонным «дублированием» и предназначены для одновременной работы со множеством однотипных документов. При этом все документы располагаются внутри одного контейнера, которым служит, как правило, главная форма. Окна могут быть двух видов:

* **модальные;**
* **немодальные.**

Создаются они одинаково, разница заключается только в способе вывода этих окон на экран. Модальное окно блокирует программу, не даёт с ней работать, пока вы это окно не закроете. Стандартный пример модального окна - окно **"О программе"**, которое присутствует почти в любом приложении. Как правило, такое окно находится в меню **"Справка"**. Пока вы не нажмете **"ОК"**, закрыв это окно, вы не сможете работать с основной программой.

Немодальные окна позволяют переключаться между ними, и программой.

В отличие от однооконного приложения, программы с **многодокументным интерфейсом (MDI)** позволяют отображать несколько документов одновременно, при этом каждый документ отображается в отдельном окне. **Приложения MDI** часто имеют пункт меню «окно» с подменю для переключения между окнами или документами.

**MDI-приложения** обязательно имеют главное окно-контейнер и дочерние окна, создаваемые по шаблону.

С закрытием главного окна закрываются все дочерние, вне зависимости от их количества, а приложение, при этом, завершает свою работу.

### Выполнение работы.

Разберём код программы. Объяснения строк содержатся в комментариях к ним:

public partial class MainForm : Form

{

int lastNum = 1; // номер последнего созданного окна

public MainForm()

{

InitializeComponent();

}

public Daughter(int number)

{

InitializeComponent();

Text = "Doc" + number.ToString(); // заголовок дочернего окна

}

void NewToolStripMenuItemClick(object sender, EventArgs e)

{

Daughter d = new Daughter(lastNum); // вызов конструктора с параметром

lastNum ++; // увеличение номера на единицу

d.MdiParent = this; // "родитель" - главное окно

d.Show();

}

void CascadeToolStripMenuItemClick(object sender, EventArgs e)

{

this.LayoutMdi(MdiLayout.Cascade); // упорядочить дочерние окна каскадом

}

void TileHorizontallyToolStripMenuItemClick(object sender, EventArgs e)

{

this.LayoutMdi(MdiLayout.TileHorizontal); // упорядочить дочерние окна горизонтально

}

void TileVerticallyToolStripMenuItemClick(object sender, EventArgs e)

{

this.LayoutMdi(MdiLayout.TileVertical); // упорядочить дочерние окна вертикально

}

void CloseToolStripMenuItemClick(object sender, EventArgs e)

{

if (this.MdiChildren.Length > 0) // если есть открытые дочерние окна,

this.ActiveMdiChild.Close(); // то закрыть активное дочернее окно

}

void ExitToolStripMenuItemClick(object sender, EventArgs e)

{

this.Close(); // закрыть главное окно и завершить приложение

}

void CloseAllToolStripMenuItemClick(object sender, EventArgs e)

{

foreach (Form f in MdiChildren) // для каждого открытого дочернего окна

f.Close(); // закрыть окно

}

void AboutToolStripMenuItemClick(object sender, EventArgs e)

{

MessageBox.Show("Это модальное окно!!!"); // показать модальное окно с сообщением

}

}

}

Добавим в программу функционал. Модифицируем дочернее окно, добавив в него TextBox, установив ему свойства Multiline на True и Dock на Fill (рис. 1).

A computer screen shot of a white screen

Description automatically generated

Рисунок 1

Добавим меню для открытия текстовых файлов и диалог для их выбора (рис. 2).

A screenshot of a computer

Description automatically generated

Рисунок 2

Реализуем необходимые функции:

// Обработчик нажатия по кнопке открытия файла

private void openToolStripMenuItem\_Click\_1(object sender, EventArgs e)

{

openFileDialog.ShowDialog(); // Вызов диалога выбора файла

}

// Обработчик успешного выбора файла

private void openFileDialog\_FileOk(object sender, CancelEventArgs e)

{

try

{

using (StreamReader sr = new(openFileDialog.FileName))

{

textBox1.Text = sr.ReadToEnd(); // Загрузка текста файла в поле

}

}

catch (Exception ex)

{

MessageBox.Show("Error reading file: " + ex.Message); // Сообщение об ошибке

}

}

Для того, чтобы элементы меню из дочернего окна автоматически не перемещались в родительское, необходимо установить свойство «MergeAction» на «Insert».

Результат (рис. 3):

Изображение выглядит как компьютер, текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 3

### Контрольные вопросы.

1. **MDI (Multiple Document Interface)** — это подход к созданию графических приложений, позволяющий размещать несколько дочерних окон (документов) внутри одного родительского окна (контейнера). Основные характеристики и свойства MDI-приложений включают:

* **Родительская форма (MDI контейнер)**:  
  Родительская форма является основным окном приложения и выступает в роли контейнера для всех дочерних окон. В ней могут располагаться меню, панель инструментов и статусная строка.
* **Дочерние формы (MDI дочерние)**:  
  Эти формы открываются в пределах родительской формы. Они могут быть свернуты, развернуты и закрыты, как и обычные окна, но находятся в границах родительского окна.

**Общие свойства и методы**:

* **IsMdiContainer:** Свойство родительской формы, которое устанавливается в true, чтобы указать, что форма должна быть контейнером для дочерних окон.
* **MdiParent:** Свойство дочерней формы, которое указывает родительскую форму-контейнер.

**Основные характеристики MDI-приложений**:

* Способность работать с несколькими документами одновременно в одном приложении.
* Управление дочерними окнами из единого родительского окна.
* Удобное размещение и организация рабочей области при работе с большим количеством документов.

1. Чтобы сделать родительскую форму контейнером для MDI-приложения, необходимо установить её свойство IsMdiContainer в true. Это можно сделать как в конструкторе формы, так и через дизайнер форм.
2. Чтобы поместить имя открытого текстового файла в заголовок дочернего окна, можно установить свойство Text дочерней формы равным имени файла:

public partial class MainForm : Form

{

public MainForm()

{

InitializeComponent();

this.IsMdiContainer = true;

}

private void OpenFile(string filePath)

{

ChildForm child = new ChildForm

{

MdiParent = this,

Text = Path.GetFileName(filePath) // Устанавливаем заголовок окна

};

// Здесь можно добавить код для загрузки содержимого файла в дочернюю форму

child.Show();

}

}

### Вывод.

Целью данной практической работы было обобщение знаний по управляющим элементам, отличие модальных и немодальных окон, а также получение практических навыков создания MDI приложений. В ходе выполнения работы были изучены и применены на практике основные концепции, связанные с модальными и немодальными окнами. Была создана родительская форма MDI-контейнера и реализованы дочерние окна для работы с несколькими документами в рамках одного приложения. Выполнение практической работы позволило закрепить теоретические знания и развить практические навыки работы с MDI приложениями, что является важным этапом в освоении современных инструментов разработки интерфейсов.