Отчёт  
по практической работе №2

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

Тема: «Создание проекта с использованием кнопочных компонентов»

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

### Цель работы.

Сформировать навыки разработки приложений с использованием кнопочных компонентов в среде программирования Visual Studio (или SharpDevelop), изучить особенности их использования.

### Теоретические сведения.

**Визуальное проектирование** — размещение на форме компонентов и задание их свойств с помощью окна свойств.

**Библиотека Windows Forms** — платформа для разработки графических приложений на базе .NET Framework. Данная библиотека содержит набор простых в использовании и расширяемых классов, позволяющих создавать приложения с развитым интерфейсом.

**Windows Forms** — технология для платформы .NET Framework в форме набора библиотек, упрощающих выполнение типичных задач приложений (чтение и запись в файловую систему, вывод информации, ввод данных пользователем и т.п.).

**Label** - обычно используются для предоставления описательного текста для элемента управления. Например, можно использовать Label, чтобы добавить описательный текст для TextBox элемента управления, чтобы информировать пользователя о типе данных, ожидаемых в элементе управления. Чтобы задать отображаемый текст метки, надо установить свойство Text элемента.

**TextBox** – служит для ввода и редактирования текста. Так же, как и у элемента Label, текст элемента TextBox можно установить или получить с помощью свойства Text. По умолчанию, при переносе элемента с панели инструментов создается однострочное текстовое поле. Для отображения больших объемов информации в текстовом поле нужно использовать его свойства Multiline и ScrollBars. При установке для свойства Multiline значения true, все избыточные символы, которые выходят за границы поля, будут переноситься на новую строку.

**Button** – является наиболее часто используемым элементом управления. Обрабатывая событие нажатия кнопки, мы может производить те или иные действия. Как и для многих элементов управления, для кнопки можно задавать текст, изображение, размер, цвет и другие свойства. Enabled — разрешает или запрещает доступ к элементу. Size — позволяет задать размер кнопке. Работает аналогично свойство Size у формы. Text — позволяет задать надпись на кнопке. TextAlign — позволяет задать выравнивание текста, например, по центу, или в левом верхнем углу кнопки. Visible — позволят скрыть кнопку, если установлено в false. Для отображения кнопки снова, нужно установить этому свойству значение true. 9 Взаимодействие пользователя с кнопкой вполне может ограничиваться одним действием – нажатием. Все, что необходимо сделать при работе с кнопкой, — это поместить ее в нужном месте формы и назначить ей соответствующий обработчик. Основной обработчик назначается для события Click данной кнопки. Инициировать нажатие кнопки можно при помощи метода PerformClick(). Для корректной работы данного метода требует установки свойства для формы KeyPreview значения, равным True. Другие элементы управления в данной работе не используются.

### Выполнение работы.

Разберём предоставленный код. Объяснение строк содержится в комментариях к ним:

void Button1Click(object sender, EventArgs e)

{

// Устанавливаем текст метки label1 равного тексту кнопки, вызвавшей событие

label1.Text = (sender as Button).Text;

}

void Button5Click(object sender, EventArgs e)

{

double x = 0, // Объявление переменной для результата

x1, // Объявление переменной для первого числа

x2; // Объявление переменной для второго числа

// Проверка, являются ли вводимые значения в текстовые поля допустимыми числами

if (!double.TryParse(textBox1.Text, out x1) || !double.TryParse(textBox2.Text, out x2))

{

// Установка текста метки label2 на "ERROR!!!" при недопустимом вводе

label2.Text = "= ERROR!!!";

return; // Завершение работы процедуры обработчика

}

// Выполнение арифметической операции в зависимости от текста метки label1

switch(label1.Text[0]) // Проверяем первый символ текста метки label1

{

case '+':

x = x1 + x2; // Сложение чисел

break;

case '-':

x = x1 - x2; // Вычитание чисел

break;

case '/':

x = x1 / x2; // Деление чисел

break;

case '\*':

x = x1 \* x2; // Умножение чисел

break;

default:

label2.Text = "= Invalid Operation"; // Вывод сообщения об ошибке при недопустимой операции

return;

}

// Установка текста метки label2 для отображения результата

label2.Text = "= " + x;

}

Перейдём к расширению функционала. Для начала включим обработку событий клавиатуры (рис. 1).

A screenshot of a computer

Description automatically generated

Рисунок 1

Сделаем программу более модульной, добавив отдельный метод по обработке подсчётов, вместе с обработкой ошибок:

private void Calculate()

{

// Проверка на запятую в текстовых полях и их замена на точки

if (textBox1.Text.Contains(','))

{

textBox1.Text = textBox1.Text.Replace(',', '.');

}

else if (textBox2.Text.Contains(','))

{

textBox2.Text = textBox2.Text.Replace(',', '.');

}

double x1, x2, x;

// Проверка для конвертации значений из textBox1 и textBox2

if (!double.TryParse(textBox1.Text, out x1) || !double.TryParse(textBox2.Text, out x2))

{

label2.Text = "= ERROR!";

return;

}

// Проверка, что в label1 есть хотя бы 1 символ

if (string.IsNullOrEmpty(label1.Text))

{

label2.Text = "= ERROR!";

return;

}

// Выполнение операции в зависимости от знака в label1

switch (label1.Text[0])

{

case '+':

x = x1 + x2;

break;

case '-':

x = x1 - x2;

break;

case '/':

if (x2 == 0)

{

label2.Text = "= ERROR!";

return;

}

x = x1 / x2;

break;

case '\*':

x = x1 \* x2;

break;

default:

label2.Text = "= ERROR!";

return;

}

// Запись результата в label2

label2.Text = "= " + x;

}

В обработчик5 кликов button5 заменим код на вызов функции подсчёта:

private void button5\_Click(object sender, EventArgs e)

{

Calculate();

}

Добавим метод для установки арифметической операции по нажатию:

private void Calculator\_KeyDown(object sender, KeyEventArgs e)

{

switch (e.KeyCode)

{

case Keys.Add:

case Keys.Oemplus: // Обработка нажатия клавиши "+" на основной клавиатуре

label1.Text = "+"; // Предотвращение ввода текста в TextBox

e.Handled = true; // Подавление дальнейшей обработки клавиши

e.SuppressKeyPress = true;

break;

case Keys.Subtract:

case Keys.OemMinus: // Обработка нажатия клавиши "-" на основной клавиатуре

label1.Text = "-";

e.Handled = true;

e.SuppressKeyPress = true;

break;

case Keys.D8 when e.Shift:

case Keys.Oem8: // Обработка нажатия клавиши "\*" на основной клавиатуре

label1.Text = "\*";

e.Handled = true;

e.SuppressKeyPress = true;

break;

case Keys.Divide:

case Keys.OemQuestion: // Некоторые клавиатуры используют "/" на основной клавиатуре

label1.Text = "/";

e.Handled = true;

e.SuppressKeyPress = true;

break;

case Keys.Enter: // Обработка нажатия клавиши "Enter" на основной и цифровой клавиатуре

Calculate();

e.Handled = true;

e.SuppressKeyPress = true;

break;

}

}

}

}

### Контрольные вопросы.

1. **События в программировании** — это механизм, позволяющий объектам сообщать другим объектам о произошедших действиях. События могут быть вызваны, когда пользователь взаимодействует с приложением (например, нажимает кнопку) или когда происходит изменение состояния объекта.

**Процедура обработки событий:**

1. Определение события: объявляется событие в классе.
2. Подписка на событие: другие объекты (например, методы) подписываются на это событие.
3. Вызов события: когда событие происходит, все подписанные обработчики вызываются.

**Применение:**

События широко используются в разработке графического интерфейса пользователя (GUI), при обработке пользовательских действий (нажатие кнопок, ввод текста и т. д.) и в других областях, где требуется реакция на определённые действия.

1. **Button (Кнопка):**

**Назначение:** Элемент управления, используемый для выполнения команд, когда пользователь нажимает на кнопку.

**Свойства:**

Text: Текст, отображаемый на кнопке.

Enabled: определяет, активна кнопка или нет.

Visible: определяет, видима кнопка или нет.

BackColor, ForeColor: Цвет фона и текста кнопки.

**События:**

Click: Событие, возникающее при нажатии кнопки.

**Методы:**

PerformClick(): Программное нажатие кнопки.

**TextBox (Текстовое поле):**

**Назначение:** Элемент управления, используемый для ввода текста от пользователя.

**Свойства:**

Text: Текст, отображаемый в текстовом поле.

Multiline: определяет, можно ли вводить несколько строк текста.

ReadOnly: устанавливает, можно ли редактировать текст.

**События:**

TextChanged: Событие, возникающее при изменении текста.

**Методы:**

Clear(): Очищает текстовое поле.

**Label (Метка):**

Назначение: Элемент управления, используемый для отображения текста пользователю.

**Свойства:**

Text: Текст, отображаемый на метке.

AutoSize: определяет, изменяет ли метка размер автоматически под текст.

ForeColor, BackColor: Цвет текста и фона метки.

**События:**

Метка не имеет значительных событий, так как она используется только для отображения информации.

**Методы:** Метка не имеет значимых методов, за исключением тех, которые наследуются от Control.

1. Да, размеры кнопки можно изменить программно. Пример кода:

using System;

using System.Windows.Forms;

namespace ExampleApp

{

public partial class MainForm : Form

{

private Button myButton;

public MainForm()

{

InitializeComponent();

Load += MainForm\_Load;

}

private void MainForm\_Load(object sender, EventArgs e)

{

// Создание кнопки и добавление ее на форму

myButton = new Button();

myButton.Text = "Click Me";

myButton.Location = new Point(50, 50);

// Изменение размеров кнопки програмно

myButton.Size = new Size(150, 50);

// Добавляем кнопку на форму

Controls.Add(myButton);

}

}

}

### Вывод.

Целью практической работы было сформировать навыки разработки приложений с использованием кнопочных компонентов в среде программирования Visual Studio (или SharpDevelop) и изучить особенности их использования. В ходе выполнения работы были изучены основные свойства, события и методы элементов управления Button, TextBox и Label. Реализованы примеры обработки событий кнопок, изменения их размеров и динамического изменения элементов управления. Работа позволила закрепить теоретические знания и получить практические навыки работы с кнопочными компонентами в среде разработки.