**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФГАОУ ВО «Пермский государственный национальный исследовательский университет»**

Физический факультет

Кафедра радиоэлектроники и защиты информации

Компонентное программирование

Лабораторная работа №3 на тему   
**«Разработка элементов управления Windows Forms»**

Выполнил:

студент 4 курса физического факультета

группа ФЗ/О ФЗ-11-2020 НБ

направление «01.03.02 Прикладная математика и информатика»

профиль «Инженерия программного обеспечения»

Кириллов Данил Алексеевич

Пермь, 2023 г.

Оглавление

[1. Цель работы 3](#_Toc148473021)

[2. Постановка задач 3](#_Toc148473022)

[**Задача 1** 3](#_Toc148473023)

[**Задача 2.** 3](#_Toc148473024)

[3. Реализация 4](#_Toc148473025)

[**Задача 1** 4](#_Toc148473026)

[**Задача 2** 9](#_Toc148473027)

[4. Список литературы 9](#_Toc148473028)

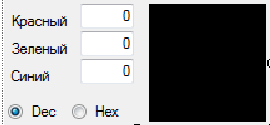
1. Цель работы

Познакомиться с разработкой элементов управления.

1. Постановка задач

Задача 1.

Необходимо разработать элемент управления, позволяющий задавать цвет в системе RGB указанием интенсивности его составляющих: красного, зеленого и синего цветов.



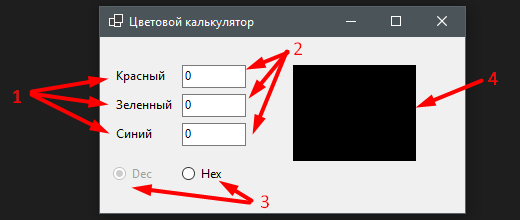
Задача 2.

Необходимо разработать элемент управления, отображающий текущее время с помощью часовой и минутной стрелок (аналоговые часы). При разработке использовать фреймворк WPF. Должна быть использована только векторная графика (не использовать растровый фон). Компонент должен поддерживать масштабирование.



1. Реализация

## **Задача 1**

В ходе выполнения был разработан элемент управления «Цветовой калькулятор».  


Элемент управления состоит из:

1. Подписи полей ввода (Label);
2. Поля ввода (TexBoxCustom), разработанное на основе базового поля ввода (TextBox);
3. Радиокнопки (RadioButton) для определения формата ввода числа;
4. Окно для просмотра цвета (PictureBox);

Для выполнения требования **«**Интенсивность цвета может быть задана как в десятичной системе исчисления, так и в шестнадцатеричной» – были реализованы радиокнопки Dec и Hex. Проверяя состояния радиокнопок, можно определить в какой системе счисления задавать насыщенность цвета.

|  |
| --- |
| txtBoxRed.Text = DexBtn.Checked ? value.R.ToString() : value.R.ToString("X"); |
| *Пример 1* – Проверка свойства радиокнопки для определения в какую систему счисления преобразовывать число. |

Для выполнения требования «Просмотр цвета должен осуществляться в интерактивном режиме, непосредственно при изменении значений в текстовых полях.» – были реализованы обработчики, срабатывающие при изменении текста в поле ввода.

|  |
| --- |
| private void txtBoxRed\_TextChanged(object sender, EventArgs e)  {  if (txtBoxRed.Text != "" && txtBoxGreen.Text != "" && txtBoxBlue.Text != "")  {  ChangeColor();  }  } |
| *Пример 2* – При изменении текста в поля ввода для красного цвета автоматически вызывается метод для изменения цвета в элементе управления. |

Для выполнения требования «Для ввода параметров должен использоваться специально разработанный элемент управления, наследующий от стандартного TextBoх.» – был разработан элемент управления TextBoxCustom.

|  |
| --- |
|  |
| *Пример 3* – Структура класса TextBoxCustom. |

Для выполнения требования «При вводе значений должен осуществляться контроль вводимых символов. В десятичном режиме допустимы только цифровые символы, в шестнадцатеричном режиме допустимы цифровые символы и символы A-F без учета регистра» – были разработаны методы (например IsHexDigit) и прописаны различные условия.

|  |
| --- |
| private bool IsHexDigit(char c)  {  return (c >= 'a' && c <= 'f') || (c >= 'A' && c <= 'F');  } |
| *Пример 4* – Метод класса TextBoxCustom, проверяющий, что введенный символ допускается в записи шестнадцатеричного числа. |

Для выполнения требования «При вводе значений должен осуществляться контроль попадания значения в диапазон от 0 до 255 (FF). При выходе числа за пределы должно подставиться максимально близкое допустимое число (0 или 255 (FF))» – запрограммирована логика, приведенная в *пример 5*.

|  |
| --- |
| protected override void OnKeyPress(KeyPressEventArgs e)  {  if (DecOrHex == "Dec")  {  if (e.KeyChar == '\b' || char.IsDigit(e.KeyChar))  {  if (e.KeyChar == '\b')  base.OnKeyPress(e);  else if (СonvertDecToInt(e.KeyChar) <= 255 && СonvertDecToInt(e.KeyChar) > 0) // Число можно интерпритировать как насыщенность цвет  base.OnKeyPress(e);  else if (СonvertDecToInt(e.KeyChar) > 255)  {  this.Text = "255"; // Устанавливаем максимально допустимую насыщенность  e.Handled = true; // Имитируем отработку события  }  else if (СonvertDecToInt(e.KeyChar) < 0) // Число меньше допустимой насыщенности  {  this.Text = "0"; // Устанавливаем минимально допустимую насыщенность  e.Handled = true; // Имитируем отработку события  }  else  e.Handled = true; // Имитируем отработку события  }  else  {  e.Handled = true; // Имитируем отработку события  }  }  else if (DecOrHex == "Hex")  {  }  } |
| *Пример 5* – Логика для реализации фильтрации текста в поле ввода (если символ некорректен – то в поле ввода ничего вводится не будет, если выходит за пределы числа, допустимого к преобразованию в насыщенность цвета – заменяем на ближайшее допустимое. Если число корректно – отрабатывает стандартное поведение при вводе числа) |

Для выполнения требования «При смене режима текущее значение должно автоматически преобразоваться.» – были реализованы конвертеры для преобразования одного цвета в другой и обработчики нажатия радиокнопок.

|  |
| --- |
| private void HexBtn\_Click(object sender, EventArgs e)  {  HexBtn.Enabled = false;  ConvertDecToHex();  txtBoxRed.DecOrHex = "Hex";  txtBoxGreen.DecOrHex = "Hex";  txtBoxBlue.DecOrHex = "Hex";  DexBtn.Enabled = true;  } |
| *Пример 6* – Обработчик нажатия на радиокнопку выбора значения в Hex |

«Элемент должен иметь свойство типа Color для установки/получения текущего цвета. После установки свойства изменения должны отобразиться (в полях ввода и в поле просмотра цвета)» – выполненное требование приведено в примере 7.

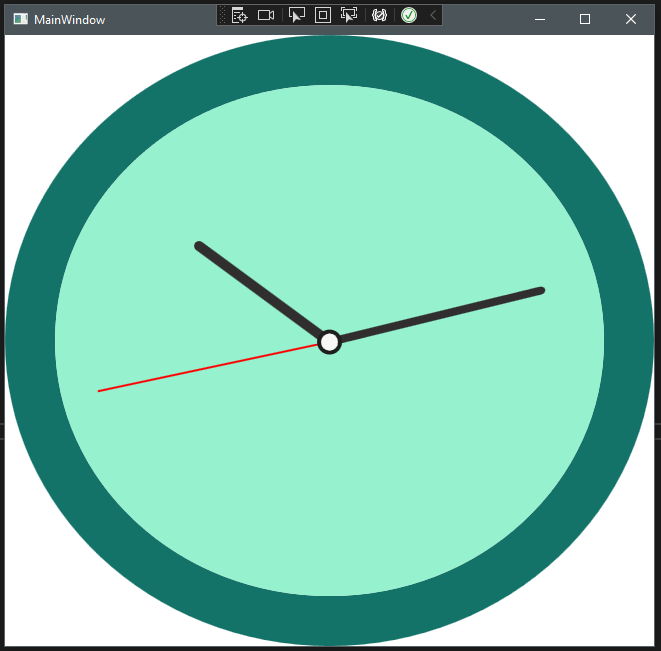
|  |
| --- |
| private Color \_color = Color.FromArgb(0, 0, 0);  public Color ColorWidget  {  get  {  return pictureBox1.BackColor;  }  set  {  \_color = value;  pictureBox1.BackColor = value;  txtBoxRed.Text = DexBtn.Checked?value.R.ToString():value.R.ToString("X");  txtBoxGreen.Text= DexBtn.Checked?value.G.ToString():value.G.ToString("X");  txtBoxBlue.Text = DexBtn.Checked?value.B.ToString():value.B.ToString("X");  }  } |
| *Пример 7* – Приватное поле \_color и свойства get set |

«Элемент должен поддерживать вставку данных из буфера обмена» – для реализации требования, было переопределено событие OnKeyDown для элемента управления TextBoxCustom.

|  |
| --- |
| protected override void OnKeyDown(KeyEventArgs e)  {  // Проверяем, были ли нажаты клавиши Ctrl и V одновременно  if (e.Control && e.KeyCode == Keys.V)  {  // Вставляем текст из буфера обмена  if (Clipboard.ContainsText())  {  string str = Clipboard.GetText();  int res;  try  {  res= DecOrHex=="Dec"? Convert.ToInt32(str):Convert.ToInt32(str, 16);  }  catch  {  res = 0;  }  if (res <= 255 && res >= 0)  str = res.ToString();  else if (res < 0)  str = "0";  else  str = "255";  Text = str;  }  }  // Вызываем базовую реализацию OnKeyDown  base.OnKeyDown(e);  } |
| *Пример 8* – Реализация сочетания клавиш Ctrl + V |

## **Задача 2**

В ходе выполнения работы был разработан элемент управления «Часы».



|  |
| --- |
| public partial class MainWindow : Window  {  const double angleSecMin = 6;  const double angleHour = 30;  public MainWindow()  {  InitializeComponent();  StartClock();  }  void StartClock()  {  this.Hide();  var timer = new DispatcherTimer();  timer.Interval = TimeSpan.FromMilliseconds(100);  timer.Tick += Timer\_Tick;  timer.Start();  }  private void Timer\_Tick(object sender, EventArgs e)  {  var rotateSec = new RotateTransform();  var rotateMin = new RotateTransform();  var rotateHour = new RotateTransform();  int sec = DateTime.Now.Second;  int min = DateTime.Now.Minute;  int hour = DateTime.Now.Hour;  rotateSec.Angle = angleSecMin \* sec;  SecondArrow.RenderTransform = rotateSec;  rotateMin.Angle = (min \* angleSecMin) + (rotateSec.Angle / 60.0);  MinuteArrow.RenderTransform = rotateMin;  rotateHour.Angle = (hour - 12) \* angleHour + rotateMin.Angle / 12;  HourArrow.RenderTransform = rotateHour;  this.Show();  }  } |
| *Пример 9* – Логика работы часов |

1. Заключение

В ходе выполнения лабораторной работы был получен опыт реализация элементов управления.

С подробным кодом можно познакомиться в репозитории <https://github.com/danon21/ComponentProgramming>.