ASK-Kalkulator - temat 1

Jakub Dobruchowski 188868
Przemysław Piątkiewicz 188823

30.03.2024

1 Zadanie

1.1 Temat

Projektowanie i implementacja kalkulatora

1.2 Cel

Projektowanie i implementacja prostego kalkulatora umożliwiającego wykonywanie podstawowych operacji arytmetycznych na liczbach.

2 Założenia szczegółowe

2.1 Interfejs użytkownika

Kalkulator będzie miał graficzny interfejs użytkownika (GUI) oparty na platformie Windows. Interfejs będzie zawierał przyciski cyfr od 0 do 9, przyciski operacji dodawania (+), odejmowania (-), mnożenia (*), dzielenia (/) oraz przyciski służące do usuwania pamięci kalkulatora (C) oraz do wykonania obliczeń (=). Dodatkowo, kalkulator będzie obsługiwał operacje za pomocą klawiatury numerycznej z zabezpieczeniem przed wprowadzeniem nieobsługiwanych znaków. Wprowadzone znaki będą wyświetlać się w specjalnie przygotowanym okienku.

2.2 Personalizacja interfejsu

Kalkulator będzie zawierał opcję zmiany koloru tła oraz wyświetlania tła na podstawie wczytanego obrazu.

2.3 Wykonywanie operacji

Kalkulator będzie wykonywał podstawowe operacje arytmetyczne (dodawanie, odejmowanie, mnożenie, dzielenie) na liczbach rzeczywistych. Obsługa kolejności wykonywania działań będzie na podstawie kolejności wprowadzonych operacji.

2.4 Pomiar czasu i aktualizacja interfejsu

Interfejs będzie zawierał cyfrowy zegar wskazujący aktualny czas w formacie HH:mm:ss oraz zegar analogowy z wskazówkami godzinową, minutową i sekundową. Cały zegar będzie rysowany dodając obsługę zdarzenia "Paint" dla "PictureBox". Czas będzie aktualizowany co sekundę. Obydwa zegary (cyfrowy i analogowy) można niezależnie ukrywać i wyświetlać.

2.5 Obsługa błędów

Kalkulator będzie obsługiwał podstawowe błędy, takie jak dzielenie przez zero, czy wczytanie jako tło nieistniejącego pliku, informując użytkownika o błędzie za pomocą okna dialogowego. Kalkulator będzie zapamiętywał stan wprowadzonych danych, umożliwiając użytkownikowi kontynuację obliczeń po wystąpieniu błędu.

3 Opis programu

3.1 Obsługa przycisków

Do interfejsu dodano 18 przycisków. Przyciski służące do wpisywania znaków do kalkulatora dzielą się na cyfry i znaki operacji. Klikając jeden z nich zostaje przekazana jego nazwa i zapisana do odpowiedniej listy (lista składników lub lista znaków operacji).

```
private void button4_Click(object sender, EventArgs e)
{
    Button button = (Button)sender;
    textBoxWynik.Text += button.Text;
}
```

Rysunek 1: Kod dla wpisywanych cyfr

```
private void button16_Click(object sender, EventArgs e)
{
    Button button = (Button)sender;
    if (!string.IsNullOrEmpty(textBoxWynik.Text))
    {
        skladniki.Add(double.Parse(textBoxWynik.Text));
        operacje.Add(button.Text);
        textBoxWynik.Clear();
    }
}
```

Rysunek 2: Kod dla wisywanych znaków operacji

W przypadku znaków operacji sprawdzane jest czy wcześniej pojawiła sie jakaś liczba, jeśli znak operacji jest pierwszym wpisanym znakiem, jest on ignorowany.

Po przyciśnięciu znaku "=" program wykonuje algorytm wyliczenia wyniku na podstawie list. Pętla iteruje po elementach listy operacje wykonując je (operacje) na podstawie zawartych w tej liście znaków. Na początku jeśli w textBoxie jest liczba to jest na dodawana do listy składników. Dodatkowo dodano zabezpieczenie przed dzieleniem przez "0", po takiej próbie wyświetli się okienko z odpowiednią informacją. Po zakończeniu wszystkich operacji w elemencie TextBox zostanie wyświetlony wynik, a listy (składników i operacji) zostaną wyzerowane.

```
private void button14_Click(object sender, EventArgs e)
{
    if (!string.IsNullOrEmpty(textBoxWynik.Text))
    {
        skladniki.Add(double.Parse(textBoxWynik.Text));
    }

    double wynik = skladniki[0];
    for (int i = 0; i < operacje.Count; i++)
    {
        switch (operacje[i])
        {
            case "+":
                wynik += skladniki[i + 1];
                break;
            case "-":
                wynik -= skladniki[i + 1];
                break;
            case "*":
                wynik *= skladniki[i + 1];
                break;
            case "/":
                if (skladniki[i + 1] != 0)
                    wynik /= skladniki[i + 1];
                else
                {
                    MessageBox.Show("Nie można dzielić przez zero!");
                    return;
                }
                break;
        }
    }
    textBoxWynik.Text = wynik.ToString();
    skladniki.Clear();
    operacje.Clear();
}
```

Rysunek 3: Kod dla znaku "=".

Dodatkowo utworzono pasek menu, który daje możliwość zamknięcia aplikacji, zmiany koloru tła interfejsu, wczytania obrazu jako tło interfejsu oraz ukrycie/pokazanie zegarów.

```
private void zamknijToolStripMenuItem_Click(object sender, EventArgs e)
{
    this.Close();
}
```

Rysunek 4: Kod do zamknięcia aplikacji

```
private void kolor4ToolStripMenuItem_Click(object sender, EventArgs e)
{
    this.BackColor = Color.Yellow;
    this.BackgroundImage = null;
}
```

Rysunek 5: Kod dla zmiany koloru tła z pozniomu paska mnu

```
1 odwołanie
private void wykresyToolStripMenuItem_Click(object sender, EventArgs e)
{
    OpenFileDialog openFileDialog = new OpenFileDialog();
    openFileDialog.Filter = "Pliki obrazów|*.bmp;*.jpg;*.jpeg;*.gif;*.png";
    if (openFileDialog.ShowDialog() == DialogResult.OK)
    {
        try
        {
            // Wczytanie obrazu z pliku
            Image image = Image.FromFile(openFileDialog.FileName);
            // Ustawienie obrazu jako tła formularza
            this.BackgroundImage = image;
            // Skalowanie tła, aby pasowało do rozmiaru formularza
            this.BackgroundImageLayout = ImageLayout.Stretch;
        }
        catch (Exception ex)
        {
            MessageBox.Show("Błąd wczytywania pliku: " + ex.Message);
        }
    }
}
```

Rysunek 6: Kod umożliwiający ustawienie wczytanego obrazu jako tło kalkulatora

```
1 odwołanie
private void cyfrowyToolStripMenuItem_Click(object sender, EventArgs e)
{
    // Wyświetlenie zegara cyfrowego, ukrycie analogowego

    labelDigitalClock.Visible = !labelDigitalClock.Visible;
    if (labelDigitalClock.Visible)
    {
        cyfrowyToolStripMenuItem.Text = "Ukryj cyfrowy";
    }
    else
    {
        cyfrowyToolStripMenuItem.Text = "Pokaż cyfrowy";
    }
    if (pictureBox1.Visible && labelDigitalClock.Visible)
    {
        cyfrowyIAnalogowyToolStripMenuItem.Text = "Ukryj analogowy i cyfrowy";
    }
    else
    {
        cyfrowyIAnalogowyToolStripMenuItem.Text = "Pokaż analogowy i cyfrowy";
    }
    // Rozpoczęcie aktualizacji cyfrowego czasu
    timerDigitalClock.Start();
    // Aktualizacja cyfrowego czasu
    UpdateDigitalClock();
}
```

Rysunek 7: Kod do wyświetlania/ukrywania zegarów (na przykładzie cyfrowego)

3.2 Obsługa klawiatury

W programie została zaimplementowana funkcja ”Form1KeyPress”, która przechwytuje obsługiwane w aplikacji znaki, rozpoznaje je i wykonuje odpowiednie operacje przycisku o nazwie takiej jak wprowadzony znak.

```
private void Form1_KeyPress(object sender, KeyPressEventArgs e)
{
    if (Char.IsDigit(e.KeyChar) || e.KeyChar == '+' || e.KeyChar == '-' || e.KeyChar
    {
        if (e.KeyChar == '=')
        {
            button14.PerformClick();
        }
        else if (e.KeyChar == 'c')
        {
            button15.PerformClick();
        }
        else
        {
            // Obsługa wprowadzania cyfr i znaków operacji
            foreach (Control control in this.Controls)
            {
                if (control is Button && control.Text == e.KeyChar.ToString())
                {
                    ((Button)control).PerformClick();
                    break;
                }
            }
        }
    }
    else
    {
        e.Handled = true; // Ignoruje wprowadzenie innych znaków
    }
}
```

Rysunek 8: Kod obsługi klawiatury

3.3 Obsługa zegarów

Utworzono funkcjce "UpdateDigitalClock()" i "timerDigitalClockTick" odpowiednio do aktualizacji tekstu zegara cyfrowego i aktualizacji cyfrowego czasu co sekundę oraz do aktualizacji obrazu zegara analogowego.

```
private void UpdateDigitalClock()
{
    // Aktualizacja tekstu cyfrowego zegara
    labelDigitalClock.Text = DateTime.Now.ToString("HH:mm:ss");
}

1 odwołanie
private void timerDigitalClock_Tick(object sender, EventArgs e)
{
    // Aktualizacja cyfrowego czasu co sekundę
    UpdateDigitalClock();
    pictureBox1.Invalidate();
}
```

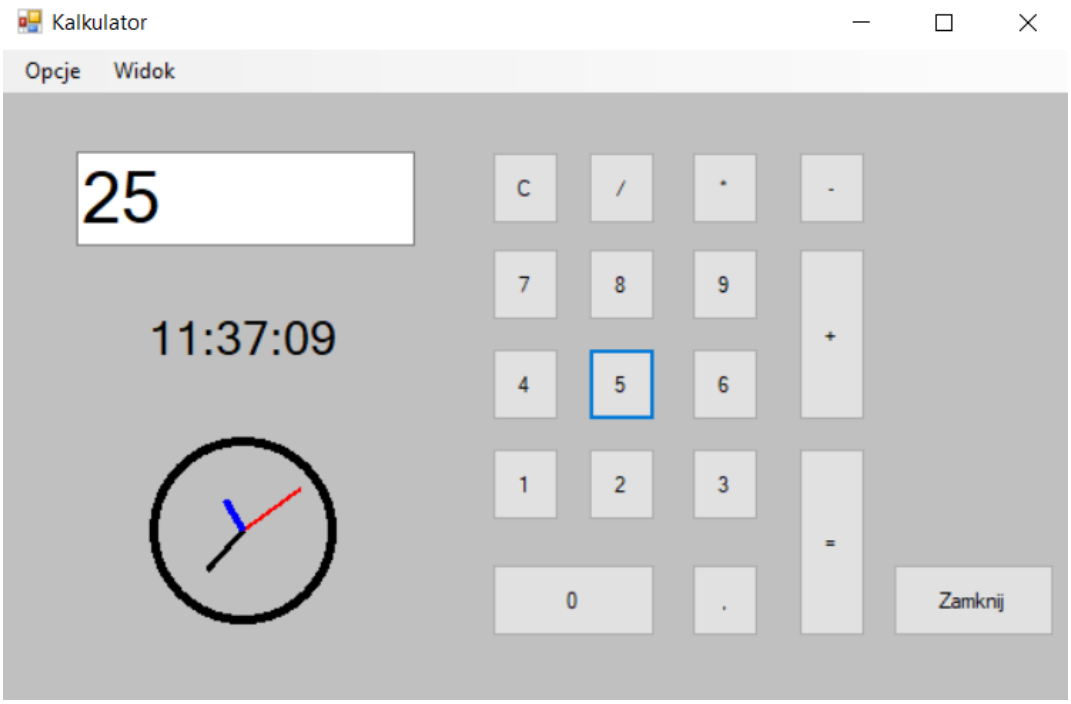
Rysunek 9: Kod aktualizacja zegarów

Funkcja "Form1Paint" rysuje tarczę oraz wskazówki zegara analogowego wykorzystując bibliotekę "PaintEventArgs" i wzory na rysowanie odpowiednich wskazówek.

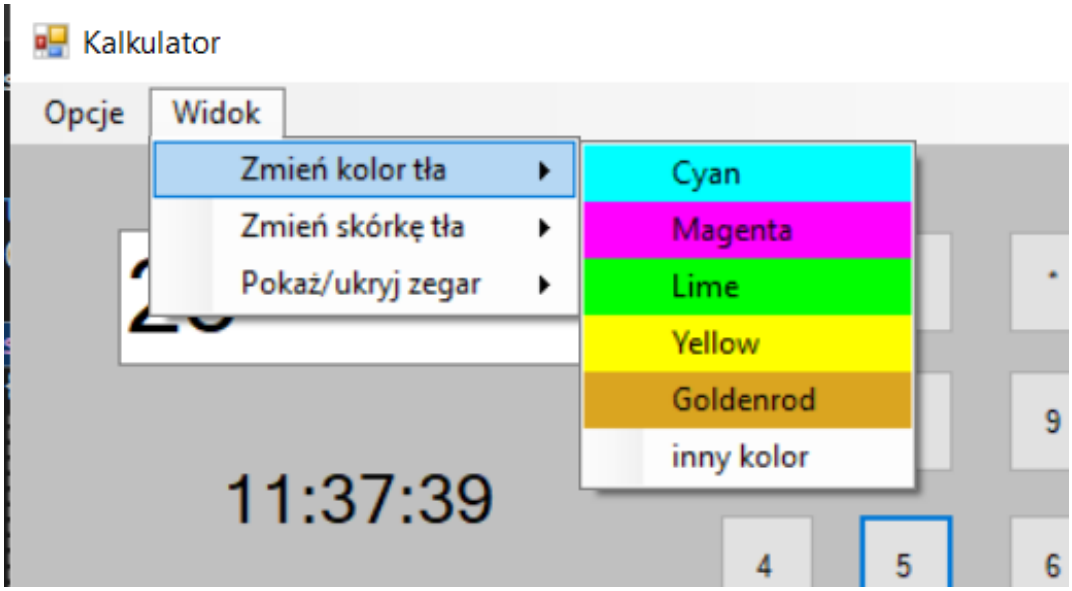
```
private void Form1_Paint(object sender, PaintEventArgs e)
{
    Graphics g = e.Graphics;
    int centerX = this.pictureBox1.Width / 2;
    int centerY = this.pictureBox1.Height / 2;
    DateTime now = DateTime.Now;
    Pen tarcza = new Pen(Color.Black, 5);
    Pen godz = new Pen(Color.Blue, 4);
    Pen min = new Pen(Color.Black, 3);
    Pen sec = new Pen(Color.Red, 2);
    g.DrawEllipse(tarcza, centerX - 50, centerY - 50, 100, 100); // Rysuje tarczę zegara

    // Sekundnik
    g.DrawLine(sec, centerX, centerY, centerX + 40 * (float)Math.Sin(Math.PI * now.Second / 30)

    // Minutowa
    g.DrawLine(min, centerX, centerY, centerX + 30 * (float)Math.Sin(Math.PI * now.Minute / 30)

    // Godzinowa
    g.DrawLine(godz, centerX, centerY, centerX + 20 * (float)Math.Sin(Math.PI * now.Hour / 6),

    tarcza.Dispose();
    godz.Dispose();
    min.Dispose();
    sec.Dispose();
}
```

Rysunek 10: Kod rysowania zegara analogowego (operacje arytmetyczne zostały ucięte)

3.4 Widok



Rysunek 11: Ogólny widok interfejsu



Rysunek 12: Opcja zmiany tła