C# .NET Aplikacja kalkulatora z zegarem

1. Wstęp

Celem zadania 1 jest napisanie aplikacji w dowolnym języku programowania dla komputerów w standarcie PC wykorzystując technikę graficznego tworzenia interfejsu użytkownika. Z założenia, interfejs powinien w sposób interaktywny wykorzystywać do komunikacji z operatorem monitor ekranowy, klawiaturę oraz myszkę.

Sugerując się proponowanym tematem w instrukcji zadania 1, postanowiliśmy wykonać kalkulator z zegarem. Do realizacji posłużyliśmy się programem Microsoft *Visual Studio*, *Windows Forms App (.NET Framework)* w języku *C#*, co umożliwia na wykorzystanie techniki graficznego tworzenia interfejsu użytkownika.

2. Założenia

Założyliśmy że aplikacja będzie się składać z:

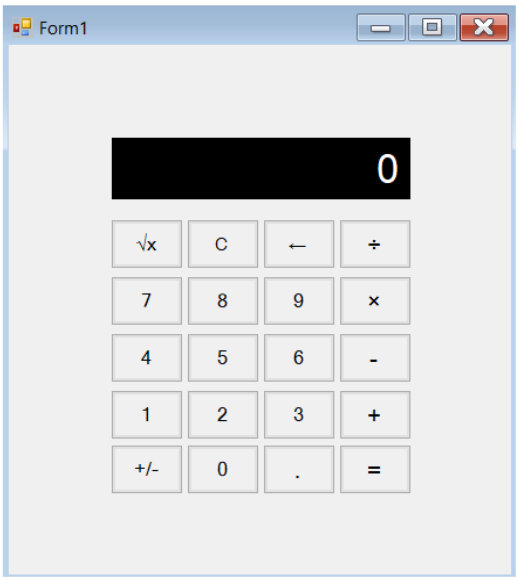
1) prostego kalkulatora, wykonującego podstawowe działania obliczeniowe,
2) wyświetlacza czasu rzeczywistego z możliwością wyboru typu zegara (analogowy lub cyfrowy),
3) przycisków, umożliwiających wybór “skórki” dla kalkulatora:
    - kolory bazowe, uwzględnione na konsoli, oraz możliwość samodzielnego wyboru koloru,
    - możliwość załadowania pobranego na komputer obrazka, jako tła kalkulatora,

Ponadto podjęliśmy decyzję dodać możliwość wpisywania poleceń dla kalkulatora zarówno za pomocą myszki, jak i wykorzystując klawiaturę.

3. Postęp pracy

3.1. Wykonanie kalkulatora

Na początku zbudowaliśmy prosty kalkulator bez dodatkowych funkcji. Do tego posłużyliśmy się rozszerzeniem pliku *.cs* który pozwala na tworzenie aplikacji w sposób graficzny. Dodaliśmy potrzebną ilość przycisków i etykiet. Na etapie początkowym plik *Form1.cs [Design]* wyglądał następująco:



Rys. 1 - graficzny interfejs tworzonego kalkulatora

Aby kalkulator mógł wykonywać funkcje obliczeniowe, dodaliśmy do każdego przycisku kod, określający, jaką funkcję ma wykonywać dany przycisk. Przykładowo: po naciśnięciu przycisku " 1" na wyświetlaczu powinna się pojawiać jedynka (rys. 3), naciskając "+" sygnalizujemy programowi, że powinna wykonać się funkcja dodawania pierwszej liczby, podanej przed chwilą, z drugą, którą za chwilę podamy (rys. 4), a po naciśnięciu "=" na wyświetlaczu powinien pojawić się wynik określonej funkcji (rys. 5).

Do poprawnego działania programu na początku określiliśmy zmienne:

```
float num1, num2, result, negnum;
char operation;
bool dec = false;
bool neg = false;
```

Rys. 2 - określone zmienne do wykonywania funkcji kalkulatora

- *num1* - pierwsza podana liczba,
- *num2* - druga podana liczba,
- *result* - wynik funkcji,
- *neg* - do sprawdzenia, czy podana liczba jest ujemna,
- *negnum* - podana ujemna liczba,
- *operation* - do określenia żądanej operacji (funkcji) liczb przez użytkownika,
- *dec* - do sprawdzenia, czy użytkownik chce wpisać liczbę zmiennoprzecinkową.

Wyświetlanie liczb na wyświetlaczu wykonywano za pomocą funkcji *changeLabel([CYFRA]).* Przykładowo, *num1_Click* odpowiada naciśnięciu "1" na graficznym interfejsie.

```
private void num1_Click(object sender, EventArgs e)
{
    changeLabel(1);
}
```

Rys. 3 - funkcja num1_Click odpowiadająca kliknięciu na przycisk "1"

Aby program zrozumiał, jakie działanie arytmetyczne użytkownik chce wykonać, funkcja naciśniętego przycisku sygnalizuje, jakie działanie na liczbach num1 i num2 powinno zajść (poprzez zapisywanie wartości *operation*). Przykładowo dla funkcji dodania:

```
private void dodawanie_Click(object sender, EventArgs e)
{
    num1 = float.Parse(CalculatorDisplay.Text);
    operation = '+';
    result = result + num1;
    CalculatorDisplay.Text = "";
}
```

Rys. 4 - funkcja dodawanie_Click odpowiadająca kliknięciu na przycisk "+"

- 1 linijka: konwersja stringa num1 do postaci float
- 2 linijka: określenie funkcji żądanej przez użytkownika
- 3 linijka: dodanie do wyniku (result) liczbę num1
- 4 linijka: wyświetlanie na wyświetlaczu "_".

Obliczenia numeryczne kalkulatora są dokonywane po wciśnięciu przycisku "=" - odpowiada to funkcji *rowno_sie_Click*. Rozwiązano to za pomocą switch'ów (rys. 5).

```
private void rowno_sie_Click(object sender, EventArgs e)
{   result = 0;
    //if(CalculatorDisplay.Text.Equals("0") == false)
    {
        switch (operation)
        {
            case '+':
                num2 = float.Parse(CalculatorDisplay.Text);
                result = num1 + num2;
                CalculatorDisplay.Text = result.ToString();
                break;
            case '-':
                num2 = float.Parse(CalculatorDisplay.Text);
                result = num1 - num2;
                CalculatorDisplay.Text = result.ToString();
                break;
            case '*':
                num2 = float.Parse(CalculatorDisplay.Text);
                result = num1 * num2;
                CalculatorDisplay.Text = result.ToString();
                break;
            case '/':
                num2 = float.Parse(CalculatorDisplay.Text);

                if (num2 == 0)
                {
                    CalculatorDisplay.Text = "NaN";
                }
                else
                {
                    result = num1 / num2;
                    CalculatorDisplay.Text = result.ToString();
                }
                break;
            default:
                break;                }           }                }
```

Rys. 5 - funkcja rowno_sie_Click odpowiadająca kliknięciu na przycisk "="

1) wykryto, jaką operację na liczbach chce wykonać użytkownik, na przykład, jeśli *operation = "+"*, wykonywano jest dodawanie (*case '+'*),
2) podczas dodawania:
   - *num2* jest przekonwertowany z postaci *string* do postaci *float,*
   - *result* to wynik dodawania dwóch liczb *num1* i *num2*,
   - na wyświetlaczu wyświetlamy uzyskany wynik *result*, konwertując go z powrotem do postaci *string*,
   - wszystkie operacje są wykonane, następuję wyjście z case'u, kończymy wykonywanie funkcji *rowno_sie_Click.*
3) odejmowanie i mnożenie - algorytm analogiczny do dodawania,
4) podczas dzielenia: zamieszczone jest zabezpieczenie, zapobiegające dzieleniu liczby przez zero (na wyświetlaczu wyświetli się "*NaN*").

Dodatkowo, istnieje możliwość zanegowania liczby wyświetlanej na display'u, do tego służy funkcja *negacja_Click*: funkcja sprawdza, czy wcześniej liczba już była ujemna

(*bool neg*), po czym dokonuje negacji, redukując minus, jeśli wcześniej liczba już była zanegowana (rys. 6)

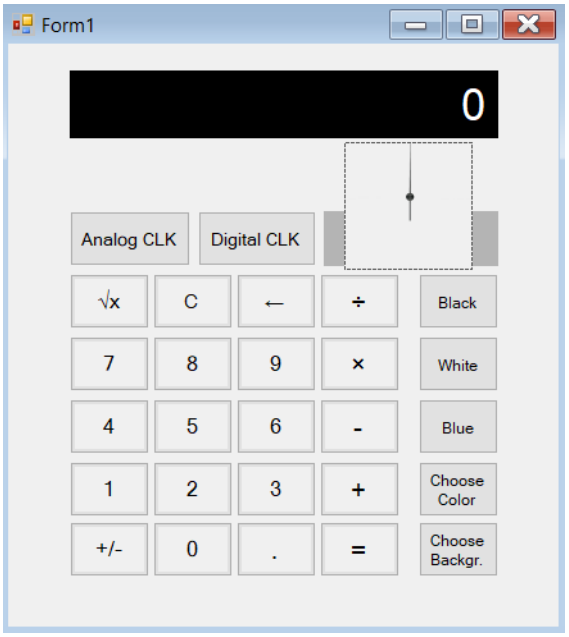
```
private void negacja_Click(object sender, EventArgs e)
{
    if (neg == true || result < 0)
    {
        negnum = float.Parse(CalculatorDisplay.Text);
        negnum = negnum * (-1);
        CalculatorDisplay.Text = negnum.ToString();
        neg = false;
    }
    else
    {
        CalculatorDisplay.Text = "-" + CalculatorDisplay.Text;
        neg = true;
    }
}
```

Rys.5 - funkcja negacja_Click odpowiadająca kliknięciu na przycisk "+/-"

3.2. Wykonanie dodatkowych funkcji aplikacji

Kolejnym krokiem było dodawanie innych przycisków do *Form1.cs [Design],* których naciśnięcie będzie skutkowało wykonywaniem funkcji, określonych w założeniach (rys. 6). Są to przyciski pozwalające na:

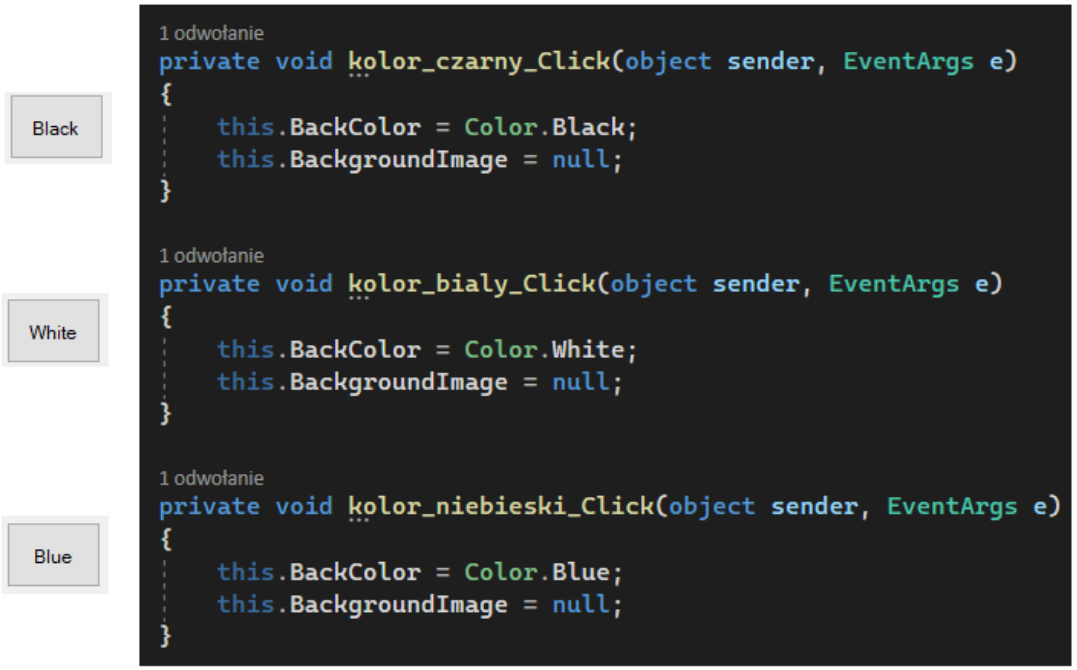
- wyświetlenie zegara analogowego/cyfrowego,
- wyboru "skórki" kalkulatora: czarny, biały, niebieski, dowolny kolory oraz dowolne tło (pobrany z urządzenia obrazek).



Rys.6 - graficzny interfejs kalkulatora po dodaniu dodatkowych przycisków

Aby uzyskać możliwość dopasowania tła dodaliśmy 5 przycisków, dzielących się na trzy rodzaje. Rodzaj pierwszy to kolory bazowe - przyciski, na których widnieją nazwy

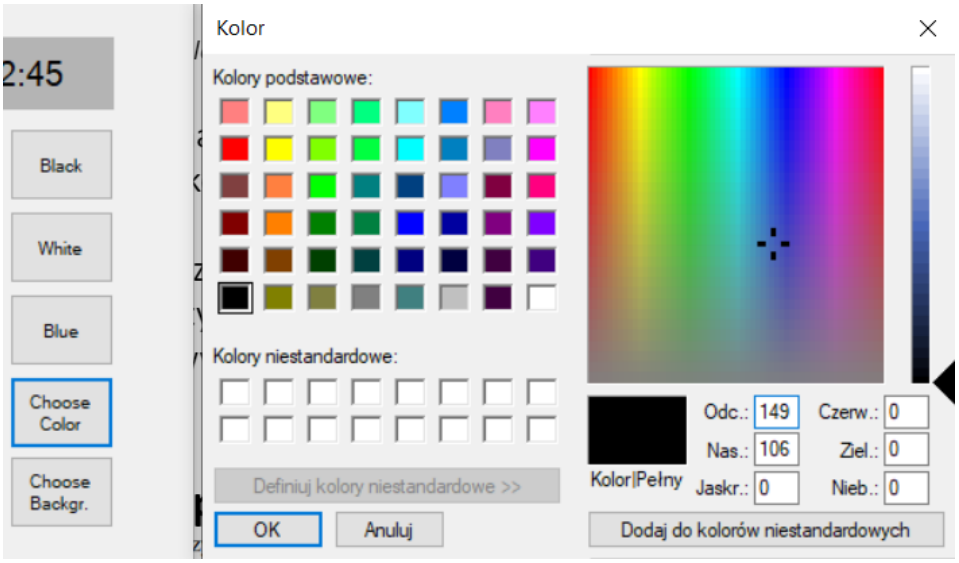
konkretnych kolorów i których przyciśnięcie skutkuje zmianą tła na ten właśnie kolor.



*Rys.7 - funkcje kolor_*nazwa koloru*_Click odpowiadające za zmianę koloru tła na jeden ze standardowych*

Ponieważ *BackgroundImage* wyświetla się na wierzchu *backColoru* każda z tych funkcji zawiera linijkę resetującą *BackgroundImage*.

Drugi rodzaj to przycisk otwierający okno dialogowe pozwalające na dobór konkretnego koloru z listy standardowych kolorów bądź dokładny dobór specyficznego koloru z wykorzystaniem funkcji “Definiuj kolory niestandardowe”. Do uzyskania takiej funkcjonalności przycisku, wykorzystaliśmy element *ColorDialog* z przybornika *Microsoft Forms*.



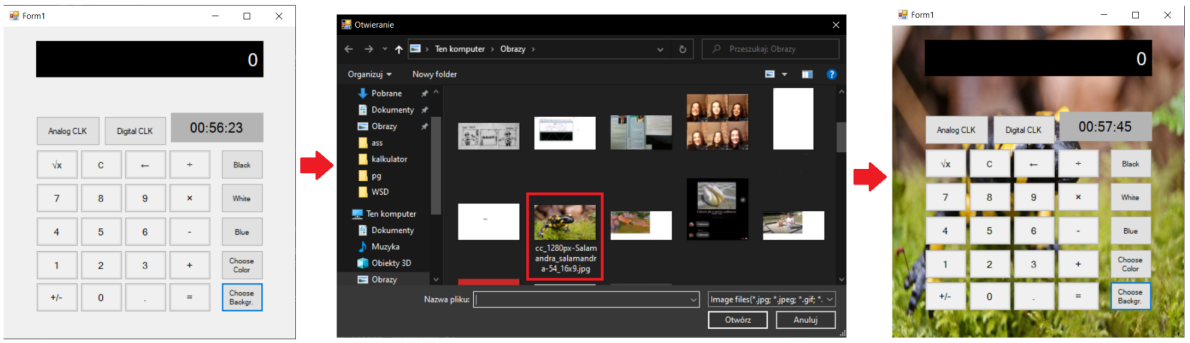
Rys.8 - widok okna dialogowego wyboru koloru po przyciśnięciu guzika “Choose Color”

```
1 odwołanie
private void wybierz_kolor_Click(object sender, EventArgs e)
{
    ColorDialog colorDialog = new ColorDialog();
    if (colorDialog.ShowDialog() == DialogResult.OK)
    {
        this.BackgroundImage = null;
        this.BackColor = colorDialog.Color;
    }
}
```

Rys.9 - funkcja wybierz_kolor_Click pozwalająca na wybór dowolnego koloru tła kalkulatora

Funkcja w pierwszym rzędzie tworzy okno dialogowe, po czym czeka aż kolor zostanie zatwierdzony (colorDialog.ShowDialog() == DialogResult.OK), w którym to przypadku resetuje BackgroundImage i ustawia BackColor na wybrany przez użytkownika.

Trzecim i ostatnim typem guzika do zmiany tła jest guzik "*Choose Backgr.*" otwierający okno dialogowe służące do przeglądania i wczytania pliku graficznego z dysku komputera. Obraz ten zostaje następnie ustawiony na tło kalkulatora.



Rys.10 - demonstracja działania przycisku "Choose Backgr."

```
1 odwołanie
private void wybierz_background_Click(object sender, EventArgs e)
{
    OpenFileDialog openFileDialog = new OpenFileDialog();
    openFileDialog.Filter = "Image files(*.jpg; *.jpeg; *.gif; *.bmp; *.png)|*.jpg; *.jpeg; *.gif; *.bmp; *.png";
    if (openFileDialog.ShowDialog() == DialogResult.OK )
    {
        this.BackgroundImage = new Bitmap(openFileDialog.FileName);
    }
}
```

Rys.11 - kod pozwalający na zmianę tła na obrazek z dysku

Do uzyskania tej funkcjonalności pobrana z przybornika *Microsoft Forms* została funkcja "*OpenFileDialog*". Kod napisany jest analogicznie do kodu funkcji wyboru koloru. Dodatkowo widoczny w kodzie filtr pozwala odfiltrować pliki graficzne od reszty plików aby nie doszło do nieporozumienia wewnątrz programu.

Kalkulator nasz posiada również zegarek o możliwości zmiany trybu między cyfrowym oraz analogowym. Do tego celu wykorzystana została funkcja "*timer*" z przybornika.

```
bool digital = true;
1 odwołanie
private void zegar_analogowy_Click(object sender, EventArgs e)
{
    digital = false;
}

1 odwołanie
private void zegar_cyfrowy_Click(object sender, EventArgs e)
{
    digital = true;
}
```

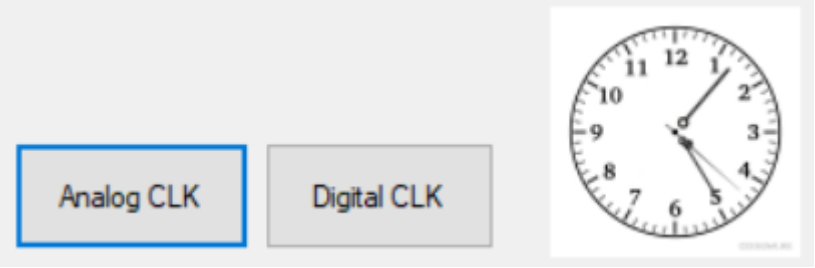
Rys.11 - kod przycisków do wyboru rodzaju zegara

Przyciski do wyboru rodzaju zegara zmieniają tylko zdefiniowaną wcześniej zmienną boolowską, określającą, czy zegar jest cyfrowy. Samo tworzenie zegara odbywa się dopiero w funkcji timera:

```
1 odwołanie
private void timer1_Tick(object sender, EventArgs e)
{
    if (digital == true)
    {
        this.wyswietlZegar.Text = DateTime.Now.ToString("HH:mm:ss");
        this.wyswietlZegar.Show();
        this.pictureBox1.Hide();
        this.pictureBox2.Hide();
        this.pictureBox3.Hide();
        this.pictureBox4.Hide();
    }
    else
    {
        DateTime now = DateTime.Now;
        int Hour = now.Hour;
        int Minute = now.Minute;
        int Second = now.Second;
        this.wyswietlZegar.Hide();
        this.pictureBox1.Show();
        this.pictureBox2.Show();
        this.pictureBox3.Show();
        this.pictureBox4.Show();
        Single AngleS = Second * 6;
        Single AngleM = Minute * 6 + AngleS/60;
        Single AngleH = Hour * 30 + AngleM/12;
        pictureBox1.Image = clock;
        pictureBox1.Controls.Add(pictureBox2);
        pictureBox2.Location = new Point(18, 17);
        pictureBox2.Image = rotateImage(hour, AngleH);
        pictureBox2.Controls.Add(pictureBox3);
        pictureBox3.Location = new Point(-1, 0);
        pictureBox3.Image = rotateImage(minute, AngleM);
        pictureBox3.Controls.Add(pictureBox4);
        pictureBox4.Location = new Point(0, 1);
        pictureBox4.Image = rotateImage(second, AngleS);
    }
}
```

Rys.12 - kod timera odpowiadający za rysowanie i update'owanie zegara

W przypadku, kiedy zegar jest w trybie cyfrowym (*digital == true*), funkcja wyświetla pole zegara, a na nim pobrany z komputera czas dnia z dokładnością co do sekundy oraz aktualizuje go co 0.1 sekundy. Chowa przy tym zegar analogowy.
Gdy zaś zegar jest w trybie analogowym (*digital == false*), to pole cyfrowe zostaje schowane, a zbiór obrazków składowych zegara analogowego zostaje wyświetlony. Te obrazki to tarcza oraz wskazówki - godzinowa, minutowa i sekundowa. Co 0.1 sekundy zegar uaktualnia się dopasowując orientację wskazówek aby wskazywać obecny dla użytkownika czas (rys. 13).



Rys.13 - Widok zegara w trybie analogowym wskazującego godzinę 1:25:22

4. Wyniki pracy

Wyniki pracy można uznać za pomyślne, gdyż aplikacja dobrze pełni swoją funkcję kalkulatora z zegarem, umożliwiając użytkownikowi dowolność przy wyborze “skórki” kalkulatora; interfejs graficzny jest prosty, czytelny i uwzględniający wszystkie funkcje przyjęte w założeniach.

Można uznać, że zaletą tego kalkulatora jest prostota budowy, natomiast wadą jest to, że taki kalkulator ma ograniczone działanie i nie uwzględnia bardziej zaawansowanych funkcji. Aby to rozwiązać, należałoby dodać przyciski, pozwalające, np., na obliczanie funkcji trygonometrycznych lub pierwiastków wyższego stopnia.

Najwięcej trudności doświadczyliśmy podczas budowy zegara analogowego, który wymagał więcej czasu na swoją realizację, niż inny komponent danego kalkulatora. Po przeprowadzeniu analizy oraz poszerzenia wiedzy na temat zegarów i języka C# udało się zrobić w pełni funkcjonalny zegar analogowy.