**Krok 1: Instalacja i Konfiguracja Visual Studio**

**1.1 Pobranie Visual Studio**

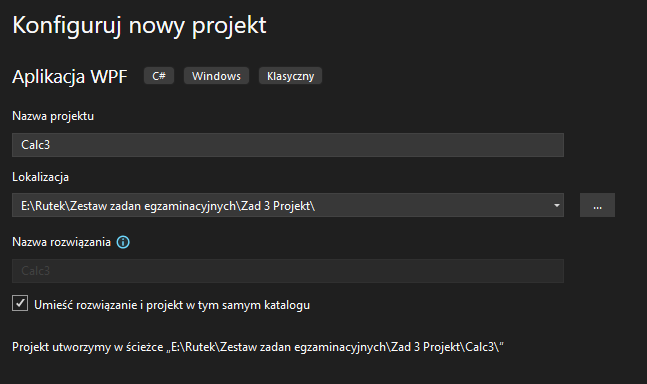
1. Otwórz przeglądarkę internetową (np. Microsoft Edge, Firefox lub Chrome).
2. Przejdź na oficjalną stronę Visual Studio pod adresem: <https://visualstudio.microsoft.com/pl/downloads/>.
3. Znajdź sekcję „Visual Studio Community” – jest to darmowa wersja oprogramowania przeznaczona dla indywidualnych programistów.
4. Kliknij przycisk „Pobierz Visual Studio Community” (lub podobny, w zależności od wersji strony).
5. Po pobraniu pliku instalacyjnego (np. vs\_community.exe) kliknij go dwukrotnie, aby rozpocząć instalację.

**1.2 Instalacja Visual Studio**

1. Po uruchomieniu instalatora pojawi się okno wyboru składników (workloads).
2. Zaznacz opcję „Programowanie aplikacji klasycznych w platformie .NET” (ang. „.NET desktop development”). Ta opcja zawiera narzędzia potrzebne do tworzenia aplikacji WPF.
3. Kliknij przycisk „Zainstaluj” w prawym dolnym rogu okna.
4. Poczekaj, aż instalacja się zakończy – może to potrwać kilka minut w zależności od szybkości Twojego komputera i połączenia internetowego.
5. Po zakończeniu instalacji kliknij „Uruchom”, aby otworzyć Visual Studio.

**1.3 Tworzenie Nowego Projektu**

1. Po uruchomieniu Visual Studio pojawi się ekran powitalny.
2. Kliknij przycisk „Utwórz nowy projekt” (znajduje się po prawej stronie ekranu).
3. W oknie „Utwórz nowy projekt” wpisz w polu wyszukiwania (u góry) frazę „WPF”.
4. Wybierz szablon „Aplikacja WPF (.NET)” z listy wyników. Upewnij się, że w opisie szablonu widnieje „C#” jako język programowania.
5. Kliknij przycisk „Dalej”.
6. W kolejnym oknie:
   * W polu „Nazwa projektu” wpisz „ProstyKalkulator”.
   * Wybierz lokalizację na dysku, gdzie projekt zostanie zapisany (np. „C:\Users\TwojaNazwa\Projekty”).
   * Pozostaw domyślne ustawienia w polu „Framework” (np. .NET 6.0 lub najnowsza dostępna wersja).
7. Kliknij przycisk „Utwórz”.

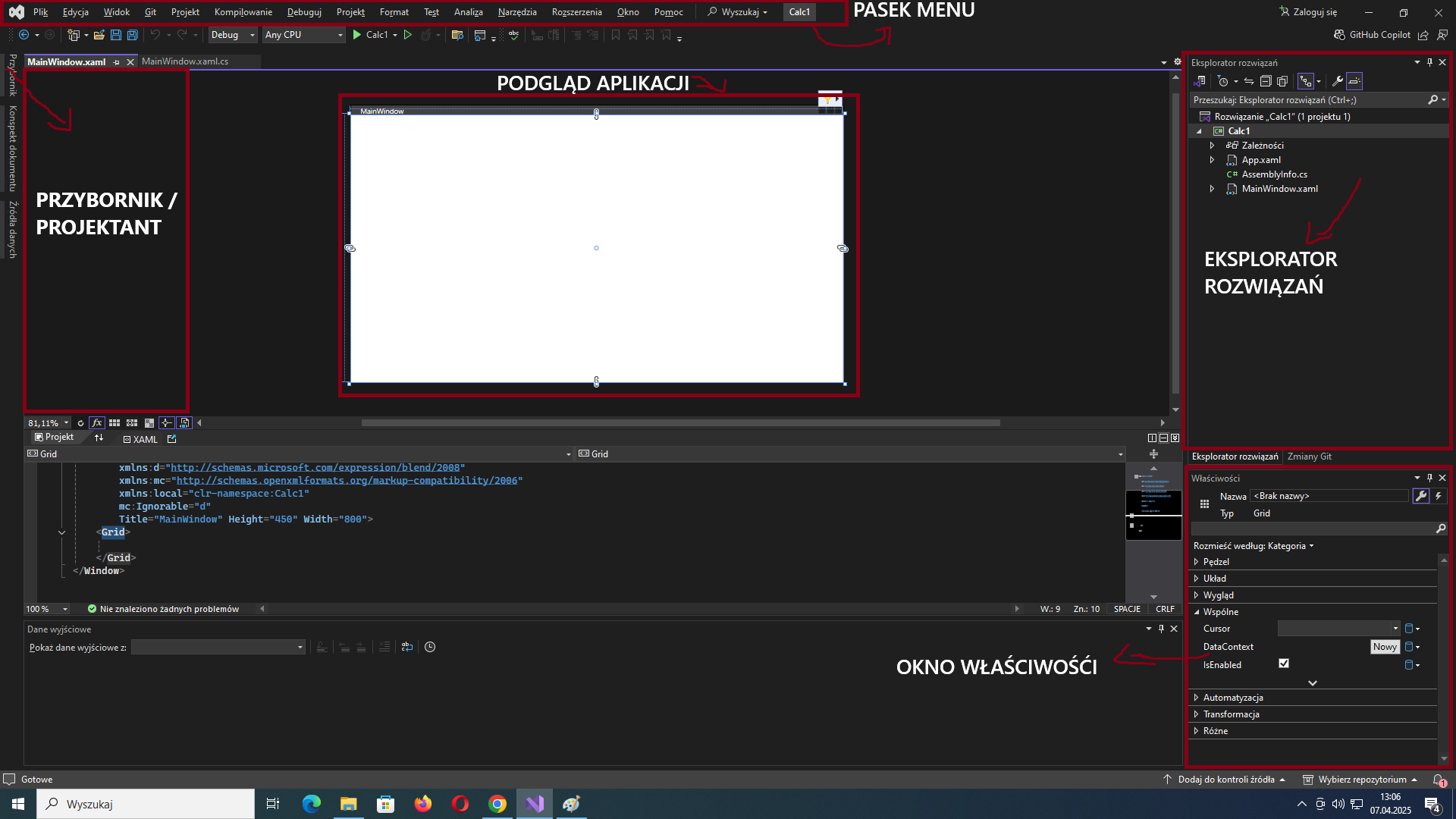


Po wykonaniu tych kroków Visual Studio utworzy nowy projekt i otworzy interfejs programistyczny.

**Krok 2: Zrozumienie Interfejsu Visual Studio**

Po utworzeniu projektu zobaczysz główne okno Visual Studio. Poniżej opisano podstawowe elementy interfejsu, które będą używane w tym tutorialu:

1. **Pasek Menu i Narzędzi**:
   * Znajduje się u góry ekranu. Zawiera opcje takie jak „Plik”, „Edycja”, „Kompiluj” czy „Debuguj”.
   * Poniżej paska menu znajduje się pasek narzędzi z ikonami, np. zielona strzałka („Uruchom”) i ikonka zapisu (dyskietka).
2. **Eksplorator Rozwiązań (Solution Explorer)**:
   * Panel po prawej stronie ekranu. Wyświetla strukturę projektu, w tym pliki takie jak App.xaml, MainWindow.xaml i MainWindow.xaml.cs.
   * Plik MainWindow.xaml odpowiada za interfejs graficzny, a MainWindow.xaml.cs za logikę programu.
3. **Projektant Interfejsu i Edytor Kodu**:
   * W centralnej części ekranu znajduje się okno podzielone na dwie części:
     + Górna część: Projektant graficzny (Design View), gdzie możesz przeciągać elementy interfejsu.
     + Dolna część: Edytor kodu XAML, gdzie definiujesz interfejs w języku znaczników.
4. **Okno Właściwości (Properties Window)**:
   * Znajduje się po prawej stronie, poniżej Eksploratora Rozwiązań. Służy do zmiany ustawień wybranych elementów interfejsu (np. rozmiaru, tekstu).

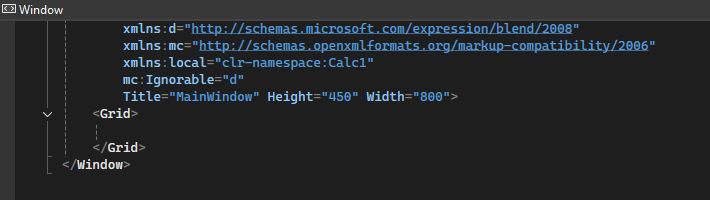


**Krok 3: Projektowanie Interfejsu Użytkownika**

W tym kroku zaprojektujemy interfejs graficzny kalkulatora w pliku MainWindow.xaml.

**3.1 Otwarcie Pliku MainWindow.xaml**

1. W Eksploratorze Rozwiązań (po prawej stronie) znajdź plik MainWindow.xaml.
2. Kliknij dwukrotnie na MainWindow.xaml, aby otworzyć go w centralnym oknie Visual Studio.
3. Zobaczysz projektant graficzny (u góry) i edytor kodu XAML (u dołu).



(Edytor kodu XAML)

**3.2 Zmiana Tytułu Okna Aplikacji**

1. W edytorze kodu XAML (dolna część okna) znajdź linię zaczynającą się od <Window.
2. Zmień atrybut Title z domyślnej wartości „MainWindow” na „Prosty Kalkulator – Imie Nazwisko”. Linia powinna wyglądać następująco:

<Window x:Class="ProstyKalkulator.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

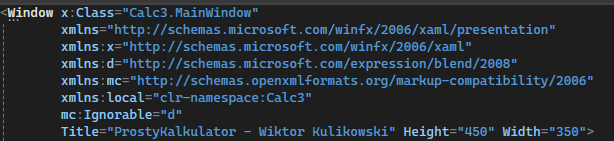
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

mc:Ignorable="d"

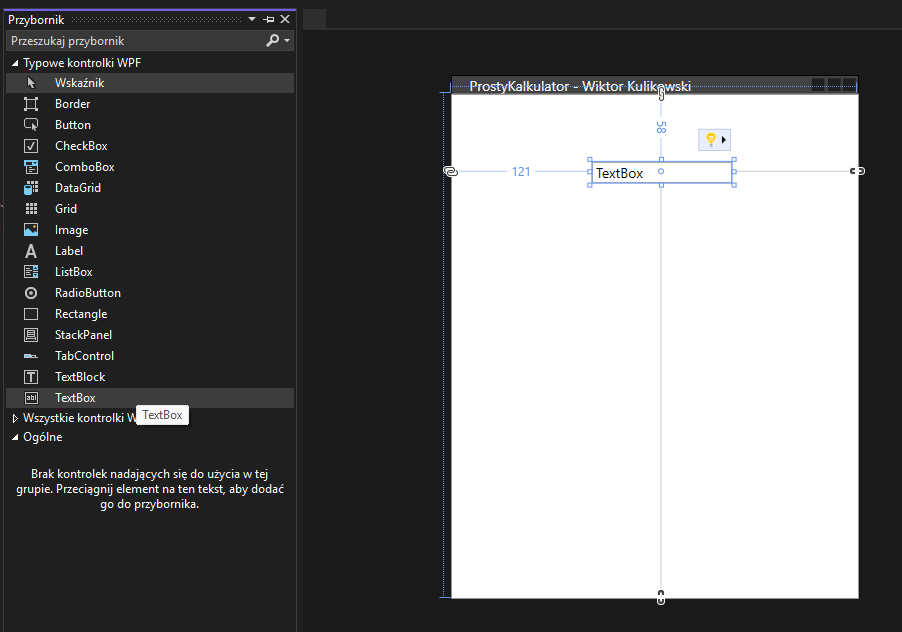
Title="Prosty Kalkulator – Imie Nazwisko" Height="450" Width="350">

1. **Zapisz zmiany, klikając ikonę dyskietki lub naciskając Ctrl+S.**

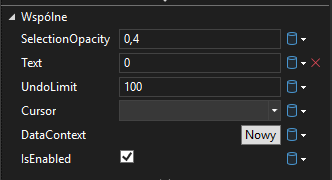


**3.2 Dodanie Pola Wyniku**

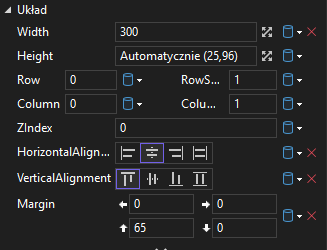
1. W panelu „Przybornik” (Toolbox), który zwykle znajduje się po lewej stronie (jeśli nie jest widoczny, kliknij „Widok” > „Przybornik” w pasku menu), znajdź kontrolkę „TextBox”.
2. Przeciągnij kontrolkę „TextBox” na projektant graficzny i upuść ją w górnej części okna.



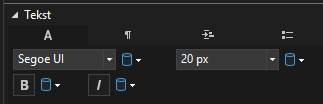
1. W oknie Właściwości (po prawej):
   * Znajdź pole „Text” i wpisz „0”.



* + Ustaw „HorizontalAlignment” na „Center”.
  + Ustaw „VerticalAlignment” na „Top”.



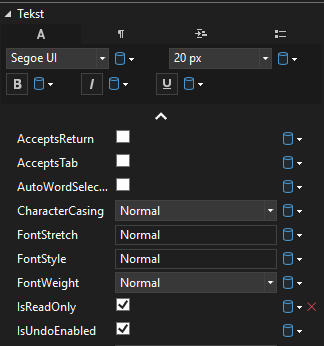
* + Ustaw „FontSize” na „20”.



* + Ustaw „Margin” na „0,65,0,0” (odstęp od górnej krawędzi).



Znajdź pole “IsReadOnly” i je zaznacz.



Ustaw pole Nazwa na “ResultBox”.



Do kodu XAML dodaj właściwość TextAlignment i ustaw ją na “Right”.



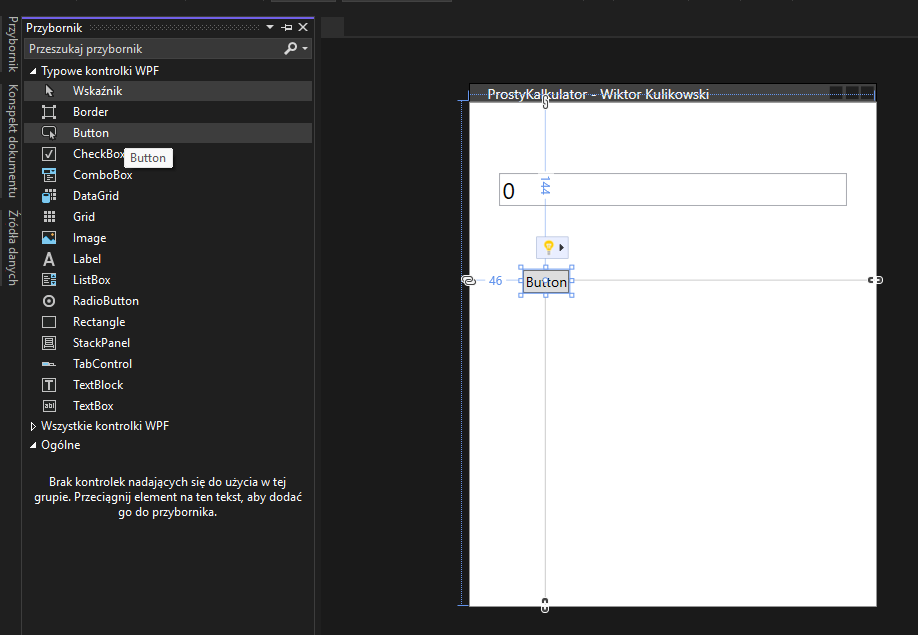
**3.3 Dodanie Przycisków**

**UWAGA:** Przycisk dodajesz zawsze tak samo:

* Przeciągasz z **Przybornika** (Toolboxa) kontrolkę **Button**.
* Ustawiasz mu parametry w oknie **Właściwości** (Properties).

1. Dodanie przycisku sqrt(x):

* Tak jak opisano wyżej, otwórz przybornik (Toolbox) i przeciągnij z niego element Button na swoją aplikację



1. W oknie Właściwości:

- Ustaw szerokość (Width) na “55”.

- Ustaw wysokość (Height) na “45”.

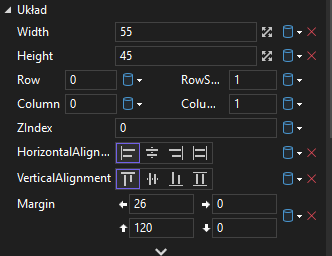
- Ustaw VerticalAlignment na “Top”.

- Ustaw HorizontalAlignment na “Left”.

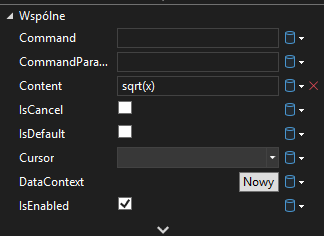
- Ustaw marginesy (Margin) na “26,120,0,0”.

- Ustaw pole Nazwa na “sqrt”.





- Ustaw właściwość Content na “sqrt(x)”



3. Dodanie przycisku pow(x):

- Skopiuj przycisk sqrt(x) używając CTRL+C i CTRL+V.

- Zmień marginesy (Margin) na “91,120,0,0”.

- Zmien właściwość Content na “pow(x)”.

- Zmień pole Nazwa na “pow”.

4. Dodanie przycisku C (wyczyszczenia):

- Skopiuj przycisk pow(x) używając CTRL+C i CTRL+V.

- Zmień marginesy (Margin) na “156,120,0,0”.

- Zmien właściwość Content na “C”.

- Zmień pole Nazwa na “Clear”.

5. Dodanie przycisku dzielenia:

- Skopiuj przycisk C używając CTRL+C i CTRL+V.

- Zmień marginesy (Margin) na “270,120,0,0”.

- Zmien właściwość Content na “/”.

- Zmień pole Nazwa na “divide”.

6. Dodanie przycisku numerycznego 7:

- Skopiuj przycisk sqrt(x) używając CTRL+C i CTRL+V.

- Zmień marginesy (Margin) na “26,176,0,0”.

- Zmien właściwość Content na “7”.

- Zmień pole Nazwa na “num7”.

7. Dodanie przycisku numerycznego 8:

- Skopiuj przycisk 7 używając CTRL+C i CTRL+V.

- Zmień marginesy (Margin) na “91,176,0,0”.

- Zmien właściwość Content na “8”.

- Zmień pole Nazwa na “num8”.

8. Dodanie przycisku numerycznego 9:

- Skopiuj przycisk 8 używając CTRL+C i CTRL+V.

- Zmień marginesy (Margin) na “156,176,0,0”.

- Zmien właściwość Content na “9”.

-Zmień pole Nazwa na “num7”.

9. Dodanie przycisku mnożenia:

- Skopiuj przycisk 9 używając CTRL+C i CTRL+V.

- Zmień marginesy (Margin) na “270,176,0,0”.

- Zmien właściwość Content na “x”.

- Zmień pole Nazwa na “multiply”.

10. Dodanie przycisku numerycznego 4:

- Skopiuj przycisk 7 używając CTRL+C i CTRL+V.

- Zmień marginesy (Margin) na “26,231,0,0”.

- Zmien właściwość Content na “4”.

-Zmień pole Nazwa na “num4”.

11. Dodanie przycisku numerycznego 5:

- Skopiuj przycisk 4 używając CTRL+C i CTRL+V.

- Zmień marginesy (Margin) na “91,231,0,0”.

- Zmien właściwość Content na “5”.

- Zmień pole Nazwa na “num5”.

12. Dodanie przycisku numerycznego 6:

- Skopiuj przycisk 5 używając CTRL+C i CTRL+V.

- Zmień marginesy (Margin) na “156,231,0,0”.

- Zmien właściwość Content na “6”.

- Zmień pole Nazwa na “num6”.

13. Dodanie przycisku odejmowania:

- Skopiuj przycisk 6 używając CTRL+C i CTRL+V.

- Zmień marginesy (Margin) na “270,231,0,0”.

- Zmien właściwość Content na “-”.

- Zmień pole Nazwa na “subtract”.

14. Dodanie przycisku numerycznego 1:

- Skopiuj przycisk 4 używając CTRL+C i CTRL+V.

- Zmień marginesy (Margin) na “26,286,0,0”.

- Zmien właściwość Content na “1”.

- Zmień pole Nazwa na “num1”.

15. Dodanie przycisku numerycznego 2:

- Skopiuj przycisk 1 używając CTRL+C i CTRL+V.

- Zmień marginesy (Margin) na “91,286,0,0”.

- Zmien właściwość Content na “2”.

- Zmień pole Nazwa na “num2”.

16. Dodanie przycisku numerycznego 3:

- Skopiuj przycisk 2 używając CTRL+C i CTRL+V.

- Zmień marginesy (Margin) na “270,286,0,0”.

- Zmien właściwość Content na “x”.

- Zmień pole Nazwa na “num3”.

17. Dodanie przycisku dodawania:

- Skopiuj przycisk 3 używając CTRL+C i CTRL+V.

- Zmień marginesy (Margin) na “270,286,0,0”.

- Zmien właściwość Content na “x”.

- Zmień pole Nazwa na “add”.

18. Dodanie przycisku numerycznego 0:

- Skopiuj przycisk 1 używając CTRL+C i CTRL+V.

- Zmień marginesy (Margin) na “26,341,0,0”.

- Zmien właściwość Content na “0”.

- Zmień pole Nazwa na “num0”.

- Zmień szerokość (Width) na “120”.

19. Dodanie przycisku kropki:

- Skopiuj przycisk 3 używając CTRL+C i CTRL+V.

- Zmień marginesy (Margin) na “156,341,0,0”.

- Zmien właściwość Content na “.”.

- Zmień pole Nazwa na “dot”.

20. Dodanie przycisku równa się (=):

- Skopiuj przycisk kropki (.) używając CTRL+C i CTRL+V.

- Zmień marginesy (Margin) na “270,341,0,0”.

- Zmien właściwość Content na “=”.

- Zmień pole Nazwa na “equals”.

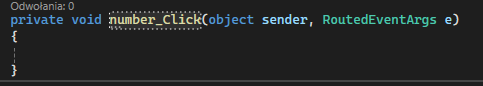
**3.4 Pisanie kodu odpowiedzialnego za logikę działania aplikacji:**

- Kliknij podwójnie na przyciski: sqrt(x), pow(x), =, C

- Spowoduje to automatyczne utworzenie metod występujących po kliknięciu

- Do przycisków numerycznych przypisz manualnie metodę number\_Click:

- Utwórz ją w pliku C#



- Przypisz ją ręcznie do przycisków numerycznych poprzez dodanie Click=”number\_Click” w pliku XAML

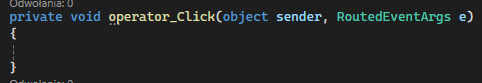




- Postępuj tak z każdym przyciskiem numerycznym 0-9

- Do przycisków operacji (+, -, x, /) przypisz manulanie metodę operator\_Click:

- Utwórz ją w pliku C#



- Przypisz ją ręcznie do przycisków operacji poprzez dodanie Click=”operator\_Click” w pliku XAML





- Postępuj tak z każdym przyciskiem operacji: +, -, x, /

1. Dodanie logiki po kliknięciu przycisku numerycznego:

private void number\_Click(object sender, RoutedEventArgs e)

{

// rzutujemy klikniety przez nas obiekt na button po to zebysmy mogli np. odczytac jego tekst

var przycisk = (Button)sender;

// po kliknieciu kropki sprawdzamy czy w liczbie jest juz kropka

if (przycisk.Content.ToString() == ".") {

// jesli brakuje kropki w polu wynikowym to ja dodajemy

if (!ResultBox.Text.Contains(".")) {

ResultBox.Text += ".";

}

return;

}

// logika powodujaca wyczyszczenie naszego textboxa wyniku jesli aktualny tekst jest rowny 0 lub zakonczylismy dzialanie poprzez klikniecie =

if (ResultBox.Text == "0" || operacja == "=") {

ResultBox.Text = "";

}

// dodajemy kliknieta cyfre do pola tekstowego

ResultBox.Text += przycisk.Content.ToString();

}

Wyjaśnienie:

* Rzutowania używany w celu odczytanie tekstu z przycisku np “7” z przycisku numerycznego 7 po to, żebyśmy mogli wykonywać na nim działania np. wyświetlenia go w polu wyniku

1. Dodanie logiki pobrania pierwszej liczby:

private void operator\_Click(object sender, RoutedEventArgs e)

{

// rzutujemy klikniety przez nas obiekt na button po to zebysmy mogli np. odczytac jego tekst

var przycisk = (Button)sender;

// przypisujemy do zmiennej pierwsza kliknieta przez nas liczbe przy okazji konwertujac ja na double i umozliwiajac wprowadzanie liczb z przecinkami

if (!double.TryParse(ResultBox.Text, System.Globalization.NumberStyles.Any, System.Globalization.CultureInfo.InvariantCulture, out pierwszaLiczba))

{

MessageBox.Show("Błędny format liczby!");

return;

}

// przypisujemy do zmiennej klikniety przez nas operator a w przypadku gdyby Content naszego przycisku bylby null to zamiast rzucic wyjatku przypisze pustego stringa

operacja = przycisk.Content.ToString() ?? "";

// w polu wyniku będziemy wyświetlac również operację aby działanie było sekwencyjne

ResultBox.Text += " " + operacja + " ";

}

Wyjaśnienie:

System.Globalization.NumberStyles.Any - jest to swego rodzaju “wyliczenie” wystepujące w przestrzeni nazw System.Globalization, które pozwala na uwzględnienie wszystkich stylów formatowania, czyli umożliwi nam obsługę kropki jako separatora dziesiętnego

System.Globalization.CultureInfo.InvariantCulture - CultureInfo jest obiektem w przestrzeni naz System.Globalization, natomiast InvariantCulture jest neutralną kulturą nie związaną z żadnym krajem używaną do standardowego formatu. Dzięki niej kropka będzie traktowana jako separator dziesiętny niezależnie od ustawień lokalnych.

Działa ona od momentu kliknięcia operatora, wtedy pobiera liczbę aktualnie znajdującą się w polu wyniku i przypisuje ją do zmiennej pierwszaLiczba.

1. Dodanie logiki przycisku C (Clear, czyszczenia):

private void Clear\_Click(object sender, RoutedEventArgs e)

{

// czyscimy wszystko do stanu poczatkowego

ResultBox.Text = "0";

pierwszaLiczba = 0;

drugaLiczba = 0;

operacja = "";

}

1. Dodanie logiki potęgowania:

private void pow\_Click(object sender, RoutedEventArgs e)

{

// przypisujemy do zmiennej pierwsza kliknieta przez nas liczbe przy okazji konwertujac ja na double i umozliwiajac wprowadzanie liczb z przecinkami

if (!double.TryParse(ResultBox.Text, System.Globalization.NumberStyles.Any, System.Globalization.CultureInfo.InvariantCulture, out pierwszaLiczba))

{

MessageBox.Show("Błędny format liczby!");

return;

}

// ustawiamy operacje na potega aby uzyc jej pozniej w metodzie equals\_Click, po to aby zczytac z niej druga liczbe ktora bedzie wykladnikiem potegi

operacja = "potega";

// czyscimy pole by wprowadzic druga liczbe (wykladnik) potegi

ResultBox.Text = "0";

}

Wyjaśnienie:

* Zamiast liczyć potęgi w tej samej metodzie przypisujemy operacje na “potega” gdyż w logice przycisku równa się występuje już pobieranie drugiej liczby, która będzie nam też potrzebna jako wykładnik potęgi.

1. Dodanie logiki pierwiastkowania:

private void sqrt\_Click(object sender, RoutedEventArgs e)

{

// tworzymy zmienna przechowujaca liczbe do pierwiastkowania i bazowo ustawiamy jej wartosc 0;

double liczbaPierwiastkowana = 0;

// przypisujemy do zmiennej liczbe kliknieta przez nas ktora chcemy zpierwiastkowac przy okazji konwertujac ja na double i umozliwiajac wprowadzanie liczb z przecinkami

if (!double.TryParse(ResultBox.Text, System.Globalization.NumberStyles.Any, System.Globalization.CultureInfo.InvariantCulture, out liczbaPierwiastkowana))

{

MessageBox.Show("Błędny format liczby!");

return;

}

// sprawdzenie czy pierwiastkowana liczba nie jest mniejsza od 0, a jezeli jest to wyswietlimy komunkat, pole wyniku zostanie zresetowane a metoda zakonczona

if (liczbaPierwiastkowana < 0) {

MessageBox.Show("Nie mozesz pierwiastkowac liczby mniejszej od 0!");

ResultBox.Text = "0";

return;

}

// obliczamy pierwiastek przy uzyciu wbudowanej biblioteki Math

liczbaPierwiastkowana = Math.Sqrt(liczbaPierwiastkowana);

// wyswietlamy wynik po pierwiastkowaniu

ResultBox.Text = liczbaPierwiastkowana.ToString();

}

Wyjaśnienie:

* Jeśli wprowadzimy liczbę mniejszą wyświetli się komunikat:

"Nie mozesz pierwiastkowac liczby mniejszej od 0!") i zakończy nam działanie metody.

* W odróżnieniu od potęgowania pierwiastkowanie może odbyć się w tej samej metodzie gdyż bazuje ono tylko na jednej liczbie - pierwszej podanej przez użytkownika.

1. Dodanie logiki liczenie wyniku:

private void equals\_Click(object sender, RoutedEventArgs e)

{

// przypisujemy do zmiennej druga kliknieta przez nas liczbe przy okazji konwertujac ja na double i umozliwiajac wprowadzanie liczb z przecinkami

// liczba która będzie konwertowana zostanie pobrana poprzez podzielenie tekstu z pola ResultBox uzywajac spacji jako separatora a [2] to ostatni element

// naszej tabliczy czyli druga liczba

if (!double.TryParse(ResultBox.Text.Split(' ')[2], System.Globalization.NumberStyles.Any, System.Globalization.CultureInfo.InvariantCulture, out drugaLiczba))

{

MessageBox.Show("Błędny format liczby!");

return;

}

// tworzymy zmienna przechowujaca wynik dzialania

double wynik = 0;

// tworzymy warunek wielokrotnego wyboru bazujacego na wartosci zmiennej operacja

switch (operacja)

{

case "+":

wynik = pierwszaLiczba + drugaLiczba;

break;

case "/":

// sprawdzenie czy nie dzielimy przez 0 a jesli tak to wyswietli sie komunikat, pole wyniku zostanie zresetowane a dzialanie metody zakonczone

if (drugaLiczba == 0) {

MessageBox.Show("Nie mozesz dzielic przez 0!");

ResultBox.Text = "0";

return;

}

wynik = pierwszaLiczba / drugaLiczba;

break;

case "x":

wynik = pierwszaLiczba \* drugaLiczba;

break;

case "-":

wynik = pierwszaLiczba - drugaLiczba;

break;

case "potega":

wynik = Math.Pow(pierwszaLiczba, drugaLiczba);

break;

}

// wyswietlamy wynik

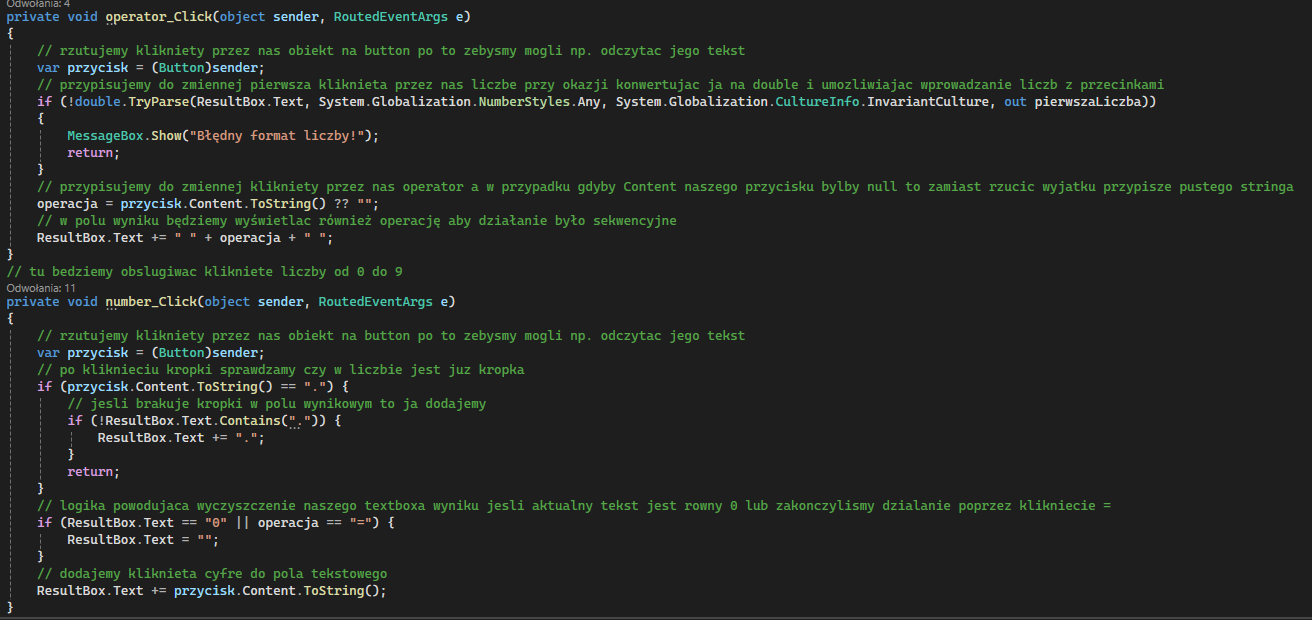
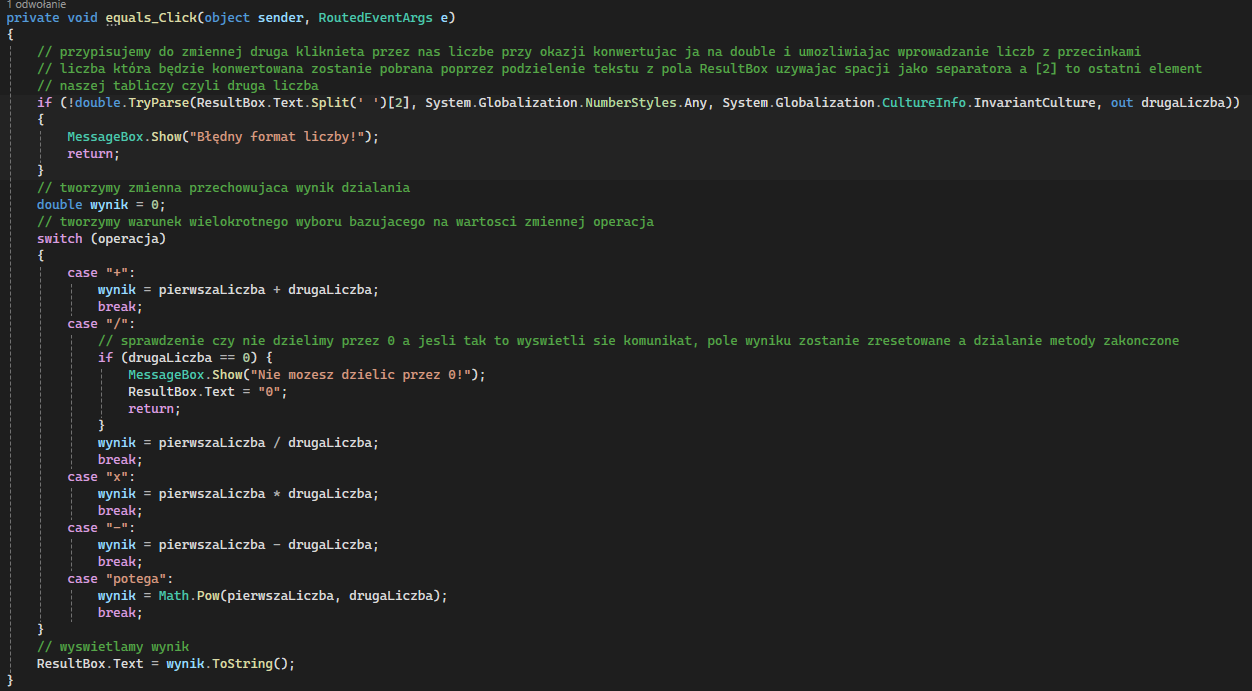
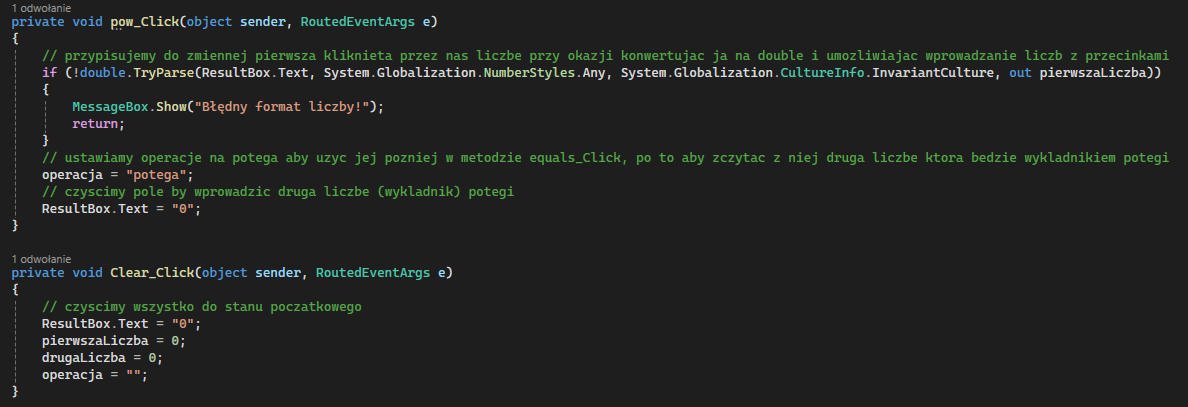
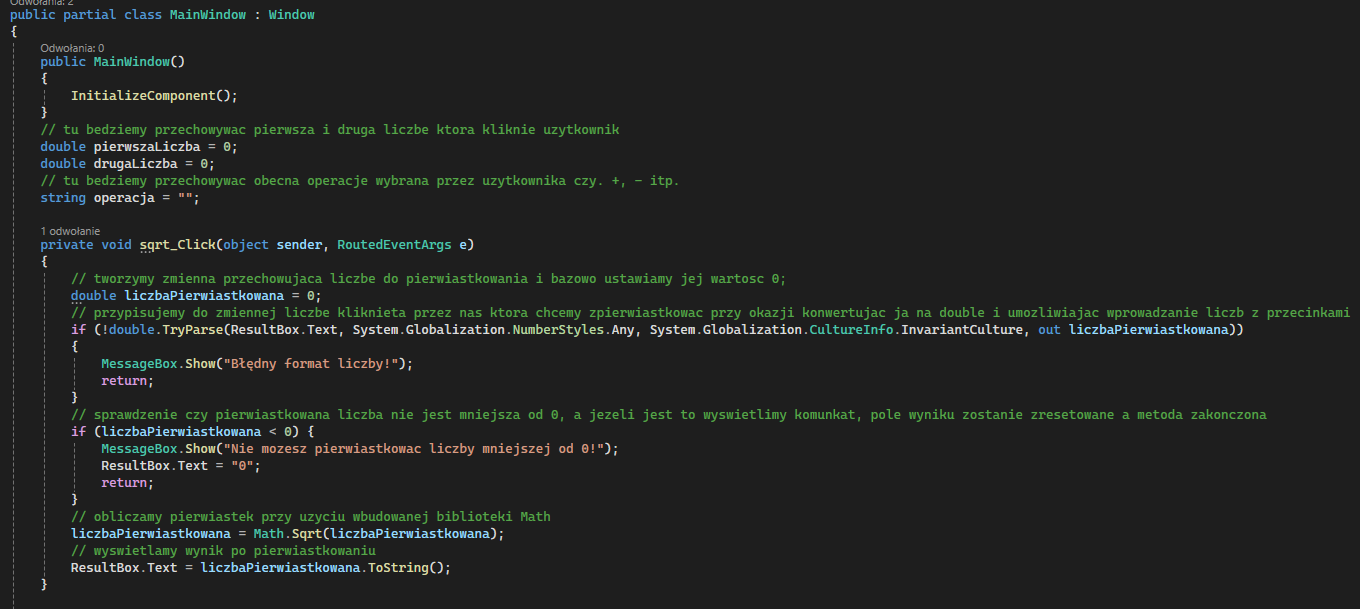
ResultBox.Text = wynik.ToString();

}

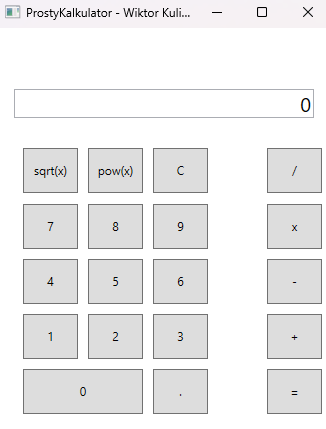
Wyjaśnienie:

* Switch-case to sposób na wybór jednego z bloków kodu do wykonania w zależnośći od wartości pewnej zmiennej lub wyrażenia (w naszym przypadku jest to zmienna operacja).
* Switch rozpoczyna instrukcję i określa zmienna lub wyrażenie, którego wartość będzie porównywana
* Case definiuje poszczególne przypadki (wartości)
* Jeśli wartość zmiennej switch zostanie dopasowana do jednego z przypadków to wykonają się instrukcje określone w tym przypadku
* Na końcu każdego bloku kodu używamy break aby przerwać wykonanie switcha
* Jeśli wprowadzimy 0 jako drugą liczbę (tą przez którą będziemy dzielić) wyświetli się komunikat:"Nie mozesz dzielic przez 0!") i zakończy nam działanie metody.
* Metoda Split dzieli nam ciąg tekstowy na tablicę, w naszym przypadku robi to po spacji gdyż tak zdefiniowaliśmy, a więc przy działaniu np. 5 + 4 utworzy nam tablicę [“5”,”+”,”4”], indexy w tablicy zaczynają się od 0 więc 5 = [0], + = [1], 4 = [2]. Dlatego używamy indexu [2] aby pobrać drugą liczbę

OSTATECZNY WYGLĄD KODU (ZAWIERA ON 6 METOD!):



OSTATECZNY WYGLĄD APLIKACJI:



PRZYKŁADOWE DZIAŁANIE:

