Informatyka II

Laboratorium – Aplikacja okienkowa

Założenia

Program będzie obliczał obwód oraz pole trójkąta na podstawie podanych zmiennych. Użytkownik będzie poproszony o podanie długości boków trójkąta. W tym celu potrzebne będą trzy zmienne, w których przechowane będą wartości długości boków. W polu tekstowym należy napisać co program będzie robił i jak go używać. W programie będzie potrzebnych około 4-5 pól tekstowych.

Przyda się również jakiś element graficzny, by użytkownik wiedział, którego boku długość podaje aktualnie. Wystarczy prosty szkic trójkąta prostokątnego. Potrzebny będzie także element, który wywoła obliczenia i pokaże wynik. W tym celu należy użyć elementu np. Przycisk. Zatem do stworzenia programu będą potrzebne następujące elementy:

- 5 pól tekstowych
- 1 przycisk
- 1 obrazek

Tworzenie programu

uruchamiamy Visual Studio i przystępujemy do prac. Na początku wybieramy "*New Project...*", a następnie *Visual C# -> Windows Form Application*. Na dole nadajemy jej nazwę i zatwierdzamy. Czekamy chwilkę, aż program stworzy projekt.

	New	Project	an foundation Server	⊗	What's New What's new in Visual 3 What's new in JNET for Getting Started Getting started with Vi Cetting started with Vi	uda maxek ud Studi	
New Project	-	-	- See 3 Mindows Appendix	and a	Real Property lies	-	2 ×
Recent		.NET Fr	amework 4.5 * Sort by: Defaul	t	- II 🗉	Searc	ch Installed Templates (Ctrl+E) 🧳 🖉 -
Installed Templates		Ľ	Windows Forms Application		Visual C#	Typ An	ne: Visual C# project for creating an application with a
✓ Visual C# Windows			WPF Application		Visual C#	Wir	ndows Forms user Interface
Web D Office		<u>c1</u>	Console Application		Visual C#		<
Cloud Reporting		51	ASP.NET Web Forms Application		Visual C#		
SharePoint Silverlight			Class Library		Visual C#		
Test WCF			Portable Class Library		Visual C#		
Workflow LightSwitch		51	ASP.NET MVC 3 Web Application		Visual C#		
 Other Languag Other Project T 	es ypes	51	ASP.NET MVC 4 Web Application		Visual C#		
Samples		Ŷ	Silverlight Application		Visual C#		
		J.	Silverlight Class Library		Visual C#		
		S.	Silverlight Business Application		Visual C#		
		, MÎ	WCF RIA Services Class Library		Visual C#	Ŧ	
Name:	Program_do_ob	liczania_ti	rojkata			_	
Location:					-	Bro	wse
Solution name:	Program_do_ob	liczania_ti	rojkata			Cre Cre	ate directory for solution
					/	Add	a to source control
	_				/		OK Cancel
		Micros	oft Visual Studio		7		
	>	Creatin	g project 'Program_do_obliczania_tro	jkata'			
	1	Visual	Studio troubleshooting and support				

Po chwili powinniśmy ujrzeć taki widok:

t.cs (Design) 9 ×	+ Properties
	Form1 System.Windows.Forms.Form
Formi D K	3 1 2 4 8
	Defense finance
	Backgroundinage (rone)
	Cancel Button (none)
	Caused/alidation True
	ContextMenuStrip (none)
	ControlBox True
P	Cursor Default
	DoubleBuffered False
	Enabled True
	E Font Microsoft Sans Sarif; 8,25
	ForeColor ControlText
	FormBorderStyle Sizeble
	HelpButton False
	🖽 kon 🔤 (izon)
	ImeMode NoControl
	IsMdiContainer False
	KeyPreview False
	Language (Default)
	Localizable False
	E Location 0; 0
	Locked False
	MainMenuStrip (none)
	MaximizeBox True
	E MaximumSiza 0;0
	MinimizeBox True
	Lti MinemumSize 0; 0
	Opacky 100%
	ta vedarng 0; 0; 0
	RightToLeft No
	Fight due to a final
	Showcon The
	TE Cite 200, 200
	Garden See
	Studiostyn Windowsflatautil oration
	Tan
	Test Form1
	TOL FORM
	Text
	The second

By móc dodawać elementy programu musimy dostać się do przybornika (*Toolbox*). Najeżdżamy więc po lewej stronie na nazwę *Toolbox*, klikamy ją, a następnie przypinamy nas przybornik klikając na pinezkę. Możemy przystąpić do pracy.

				A.A.M.B.L.V.	Juli - Juli -	nennà . I he for le la la la la la regione en
	Form1 of (Decion		Dat	Toolbox	- 4 × F	orm1.cs [Design] 😕 🗙
ta	roma.cs (Desigi		S	Search Toolbox	ρ-	12
l li			5	All Windows Forms		Form1
8	E Form1	Toolbox 👻 📼 🗙	0	Common Controls		
1		Search Toolbox		Pointer		
ě.		All Windows Forms		Button	_	
 2		Common Controls		CheckBox		
		🗧 📐 Pointer		E CheckedListBox		
		🖉 💷 Button		ComboBox		
		CheckBox		DateTimePicker		P
		E CheckedListBox		A Label		
	\ \	E ComboBox		A linklabel		
		DateTimePicker		E ListBox		
		A Label		ListView		
		▲ LinkLabel		() MackedTextRev		
		E Lindon		Masked Textbox		
				Nexif Jaco		L
				E Notivicon		
				NumericupDown		
				PictureBox		
				ProgressBar		
				RadioButton		

Na początku zmieńmy nazwę naszego okienka. W tym celu w prawej zakładce *Properties* szukamy opcji *Text*, gdzie zmieniamy nazwę Form1 na nasza własną.



Elementy programu

Dodajemy kolejno elementy naszego programu wybierając je z przybornika (*Toolbox*) i przeciągając na nasze okienko. Potrzebujemy:

- 5 x Label
- 3 x numericUpDown
- 1 x Button
- 1 x PictureBox

Oczywiście nie musimy dodawać wszystkiego naraz. Możemy to robić po kolei. By lepiej pogrupować elementy możemy nadać im własne nazwy. W tym celu klikamy LPM raz na danym elemencie i w prawej zakładce *Properties* znajdujemy gałąź *DataBindings*. Rozwijamy ja i w polu *Name* wpisujemy własną nazwę.

romittos (besign) 🕞 🔨		Properties	•	^
(label1 System.Windows.Fo	rms.Label	*
🖳 Program do obliczania trójkąta protokątnego		8 🛛 🗲 🖉		
	label1			
	Lumming.			
		(Name)	label1	
		AccessibleDescription		
		AccessibleName		
		AccessibleRole	Default	
		AllowDrop	False	
		Anchor	Top, Left	
		AutoEllipsis	False	
		AutoSize	True	
		BackColor	Control	
		BorderStyle	None	
		CausesValidation	True	
		ContextMenuStrip	(none)	
		Cursor	Default	

Dla zachowania porządku nazwy w przykładzie to:

- *label1 ->* Opis
- *label2* ->Działanie
- numericUpDown1 -> dl_boku_A
- *numericUpDown2* -> dl_boku_B
- numericUpDown3 -> dl_boku_C
- *button1* -> przycisk_Oblicz
- *picturebox* -> trojkat

Ustawiamy wszystko. Wszelkich zmian w parametrach elementów dokonujemy w prawej zakładce *Properties*. W polu *Text* możemy zmienić domyślny tekst pól tekstowych czy przycisków.

	TabIndex	0	
	Tag		
	Text	label1	-
Program służy do oblic -pole trójkąta -obwód trojkąta	zania następujących j	parametrów trójkąta prostokątneg	0:

Natomiast w przypadku elementu *PictureBox,* możemy tam załadować obraz klikając na przycisk przy parametrze Image.

OK Cancel	System.Drawing.Bitmap
-----------	-----------------------

Gdy umieścimy wszystkie elementy programu naciskamy klawisz *F5* lub zielona strzałkę w pasku narzędziowym. Jak widzimy nasz program uruchomił się, jednak jak można się tego spodziewać, po kliknięciu na przycisk "*Oblicz*" nic się nie dzieje.



Czas to zmienić..

Trochę kodu

Na początku wyłączmy nasz program zamykając go. By stworzyć funkcję wywołującą określone zdarzenia po naciśnięciu przycisk "*Oblicz*" klikamy go podwójnie LPM. Przechodzimy do pisania kodu. Ukaże nam się taki widok:

private void przycisk_Oblicz_Click(object sender, EventArgs e)

```
// Tutaj wstawiamy zdarzenia
```

to funkcja, która wywoła określone przez nas zdarzenia po naciśnięciu przycisku "Oblicz" (nazwanego przez nas przycisk_Oblicz). Zdarzenia do wykonania wpisujemy wewnątrz nawiasów klamrowych.

Na początek dodajmy zdarzenie, które wyświetli zdefiniowany przez nas tekst po naciśnięciu przycisku za pomocą wyskakującego okienka. Tak więc w nawiasach klamrowych wpisujemy:

MessageBox.Show("Tutaj zobaczysz swój wynik");

Całość powinna wyglądać tak:

```
namespace Program_do_obliczania_trojkata
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private void przycisk_Oblicz_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Tutaj zobaczysz swój wynik");
        }
    }
}
```

Ponownie uruchamiamy nasz program poprzez *F5* i naciskamy przycisk *Oblicz*. Efektem jest ukazanie się okienka z naszym tekstem.



Zmienne

Przypiszmy do długości boków litery *a,b,c* z czego c będzie przeciwprostokątną trójkąta. Chcemy także obliczyć obwód i pole tego trójkąta. Razem potrzebujemy pięciu zmiennych, które nam te wartości przechowają. Wprowadźmy to przed naszą informacją o wyniku.

Przykładowo wprowadzona przez użytkownika długość boku A odczytamy następująco.

decimal a = dl_boku_A.Value;

Tłumacząc: decimal – typ zmiennej

dl_boku_A – nazwa elementu jaką przypisaliśmy obiektowi, w którym użytkownik podaje nam długość tego boku (domyślnie był to obiekt numericUpDown1)

Value – wartość wskazana, wpisana przez użytkownika do elementu dl_boku_A

Podobnie postępujemy także ze zmienna obwód i pole. Są to po prostu wzory matematyczne

decimal obwod=(a+b+c); deciaml pole=((a*b)/2);



Ta wygląda nasz kawałek kodu po wpisaniu zmiennych. Zmieńmy teraz nasz tekst wyświetlany w wyskakującym okienku na wiadomość pokazująca nasz wynik:

MessageBox.Show("Obwód tego trójkąta wynosi: " + obwod + ", a tego pole jest równe: " + pole);



Po ponownym uruchomieniu programu, wprowadzeniu losowych danych i kliknięciu przycisku "Oblicz" otrzymujemy następującą informację:



Mała rzecz, a jak ważna

Jak wiadomo trójkąt prostokątny jest specyficznym rodzajem trójkąta. Więc co się stanie, gdy użytkownik poda nam długości, które nie tworzą tego trójkąta? Musimy się przed tym zabezpieczyć. Skorzystajmy więc z twierdzenia Pitagorasa , które mówi nam o tym, że trójkąt prostokątny otrzymamy wtedy, gdy suma kwadratów jego przyprostokątnych, będzie równa kwadratowi przeciwprostokątnej. Innymi słowy nasz warunek, jaki podane wartości muszą spełnić, by trójkąt został uznany za trójkąt prostokątny to:

a*a+b*b = c*c

Bok także nie może być równy 0, więc spełnione muszą być łącznie 4 założenia:

- a*a+b*b = c*c
- a>0
- b>0
- c>0

Skorzystajmy z instrukcji warunkowej if, by sprawdzić czy nasz warunek zaszedł

if (warunek) {

//zdarzenie, jeżeli warunek jest prawdziwy

}

else{

//zdarzenie w przeciwnym wypadku - warunek jest nieprawdziwy

};

W tym celu "przerabiamy" nasz kod wiadomości na instrukcję if.

```
if (a>0 && b>0 && c>0 && (a * a + b * b == c * c)) {
```

MessageBox.Show("Obwód tego trójkąta wynosi: " + obwod + ", a tego pole jest równe: " + pole);

}

else{

MessageBox.Show("Podane przez Ciebie długości nie tworzą trójkąta prostokątnego!");

};

Całość prezentuje się następująco:



Sprawdźmy teraz:



Jak widać program działa poprawnie.

Modyfikacje.

Zmodyfikujmy pola służące do wprowadzania długości boków. Zastąpmy pola *numericUpDown* polami *textBox*. Pole *textBox* umożliwia wpisanie ciągu znaków. Następnie ten ciąg należy przekonwertować na zmienną typu *double*, używając do tego odpowiedniego polecenia.

Kolejną modyfikacją jest dodanie przycisku wyboru *RadioButton*. Przy pomocy tego przycisku będziemy mogli wybierać czy ma być obliczane pole powierzchni czy obwód trójkąta.