

Technologie informacyjne

Programowanie w języku
Visual Basic .NET

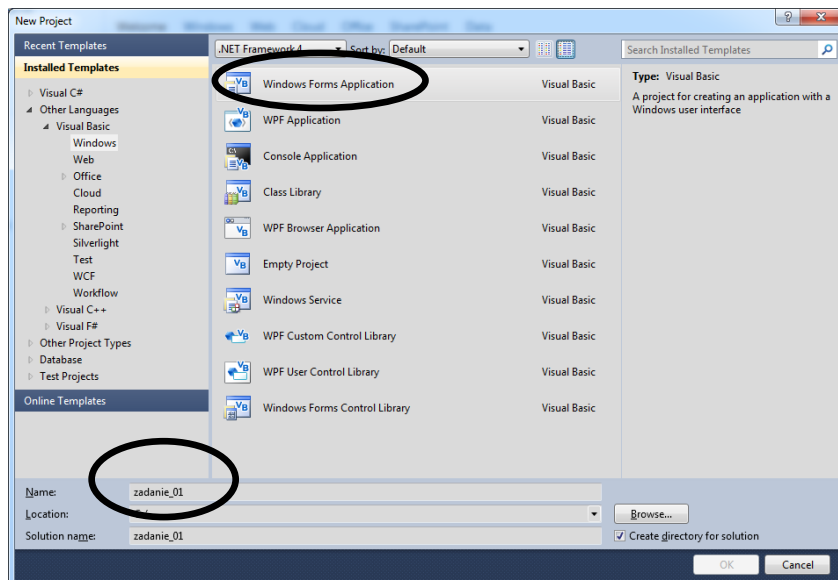
Instrukcja do zajęć laboratoryjnych

Opracował: Jacek DIAKUN

Ćwiczenie 1.

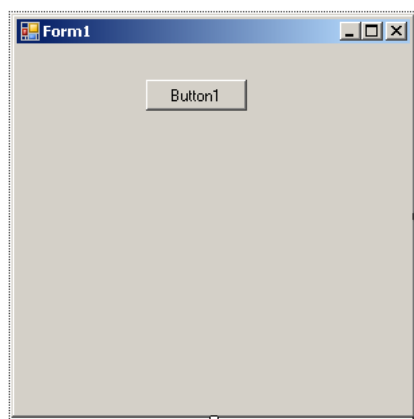
Utworzyć program zawierający jedno okno i przycisk. Po kliknięciu na przycisk program ma zakończyć swoje działanie.

1. Uruchom *Visual Studio* (*Start* → *Programy* → *Microsoft Visual Studio 2010* → *Microsoft Visual Studio 2010*).
2. Utwórz nowy projekt będący aplikacją systemu Windows stworzoną w języku Visual Basic .NET (*File* -> *New Project* -> *Visual Basic* -> *Windows* -> *Windows Application*), określając jego nazwę:

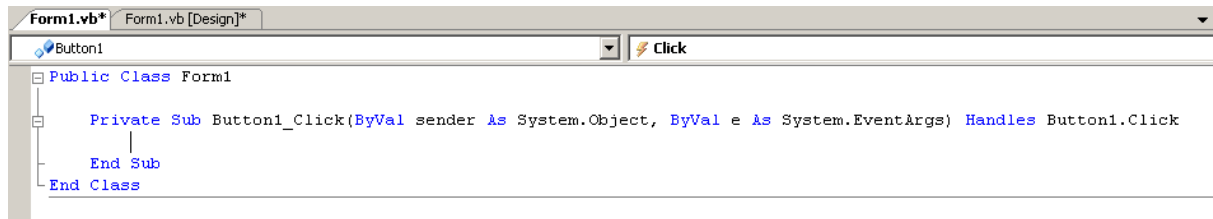


Po wybraniu powyższej opcji na ekranie pojawi się widok okna programu *Visual Studio* (patrz: *Załącznik 1*).

3. Na formularzu umieść komponent (przecignij myszą z Przybornika na projekt okna tworzonej aplikacji) przycisku (*Button*) zgodnie z poniższym rysunkiem:

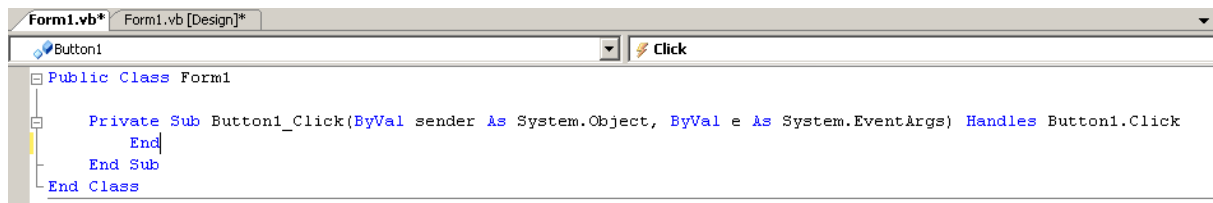


Kliknij dwukrotnie na przycisku z napisem *Button1* – pojawi się okno umożliwiające wprowadzenie kodu programu będącego reakcją na zajście zdarzenia pojedynczego kliknięcia lewym przyciskiem myszy na przycisk:



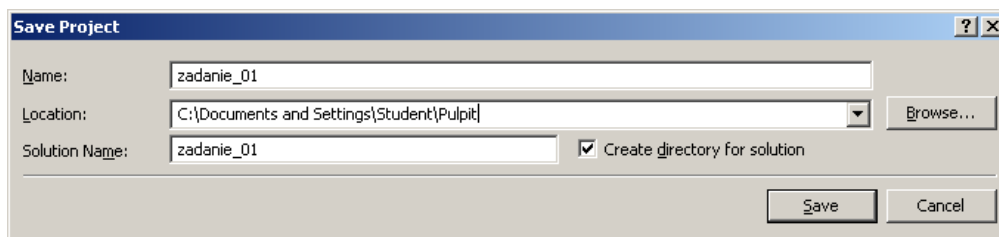
```
Public Class Form1
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    End Sub
End Class
```

4. Wprowadź następujący kod:



```
Public Class Form1
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    End Sub
End Class
```

5. Zapisz w wybranym katalogu dotychczas wykonaną pracę (*File* → *Save All*; zaakceptuj nazwy zaproponowane przez VB, zmieniając ewentualnie miejsce zapisu projektu na dysku):



6. Uruchom program pod kontrolą środowiska VS (Debug → Start Debugging lub klawisz *F5*) i sprawdź jego działanie.
7. Na podstawie wykonanego projektu utwórz samodzielną aplikację systemu *Windows* (inaczej: skompiluj program do postaci wykonywalnej) (*Build* → *Build zadanie_01*). Postać wykonywalna programu zostanie zapisana w miejscu zapisu projektu w katalogu *Bin/Debug*

Uwagi:

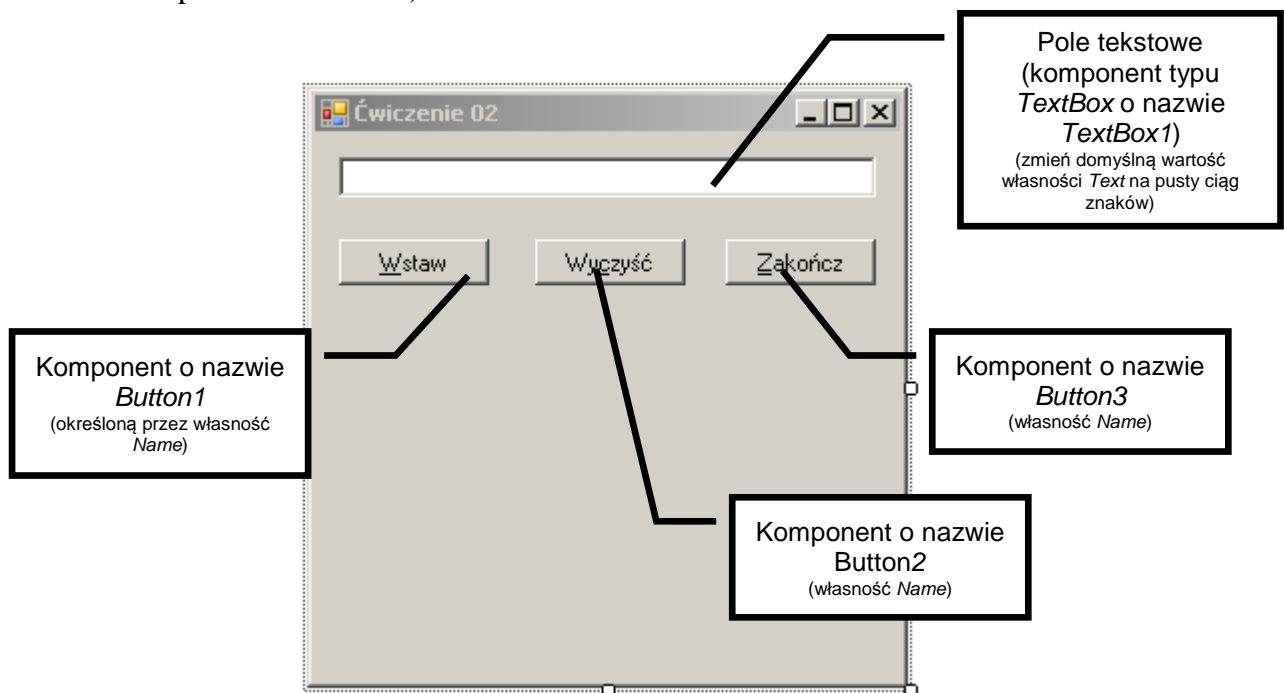
1. Otwarcie nowego projektu (rozpoczęcie tworzenia nowego programu działającego w środowisku *Windows*).
2. Tworzony program będzie zawierał jedno okno (które na etapie projektowania nazywane jest *formularzem*) o nazwie *Form1* (własność *Name* w oknie *Properties*). Na środku okna umieszczono jeden komponent – przycisk o nazwie *Button1*.
3. Po kliknięciu na przycisk ma zajść określona reakcja (zakończenie działania programu). Należy więc utworzyć fragment kodu programu (wstawić do stworzonego szkieletu procedury) będący reakcją na zdarzenie kliknięcia na przycisk.
4. Procedura obsługi zdarzenia zawiera jedną instrukcję języka – procedurę *End*, kończącą działanie programu.

Do samodzielnego wykonania:

1. Nadaj własnościom *Text* komponentu *Button1* oraz formularza *Form1* wartości odpowiednio: *Koniec* i *Ćwiczenie nr 1*. Co określa własność *Name* dla tych komponentów?
2. Zmień wartość *Koniec* na *Konie&c*. Jakie działanie powoduje umieszczenie znaku „&” we własności *Text* przycisku?
3. Co powoduje zmiana wartości następujących własności formularza:
 - a. *FormBorderStyle* (spróbuj zmieniać wielkość okna uruchomionego programu dla kolejnych wartości przypisywanych tej własności),
 - b. *MaximizeBox*, *MinimizeBox* i *ControlBox* (wypróbuj dla *FormBorderStyle = Sizeable*)
4. Co określają następujące własności:
 - a. *WindowState*,
 - b. *StartPosition*?

Ćwiczenie 2.

1. Otwórz nowy projekt.
2. Zaprojektuj formularz wg poniższego rysunku (zmień odpowiednio wartość własności *Text* komponentu *TextBox1*):



3. Uzupełnij kod obsługi zdarzenia kliknięcia na komponencie (przycisku) *Button1*:

```
TextBox1.Text = "Visual Basic"
```

1

4. Uzupełnij kod obsługi zdarzenia kliknięcia na komponencie (przycisku) *Button2*:

```
TextBox1.Text = ""
```

2

5. Uzupełnij kod do programu w taki sposób, aby po kliknięciu na komponencie (przycisku) *Button3* program kończył swoje działanie.

Uwagi:

- 1 nadanie określonej własności komponentu odbywa się teraz z poziomu kodu programu – tym samym zmiana dokonuje się w czasie działania programu, a nie, jak poprzednio, jest określana „ręcznie” przed jego uruchomieniem. Zmiana wartości określonej własności komponentu odbywa się w kodzie programu wg schematu:

```
nazwaKomponentu.własność = wartość
```

- 2 wstawienie „pustego” ciągu znaków usuwa wpisany tekst.

Do samodzielnego wykonania:

1. Dodaj nową linię (przed instrukcjami *End Sub*) do procedury obsługi zdarzenia kliknięcia na komponencie (przycisku) *Button2* i dopisz następujący kod:

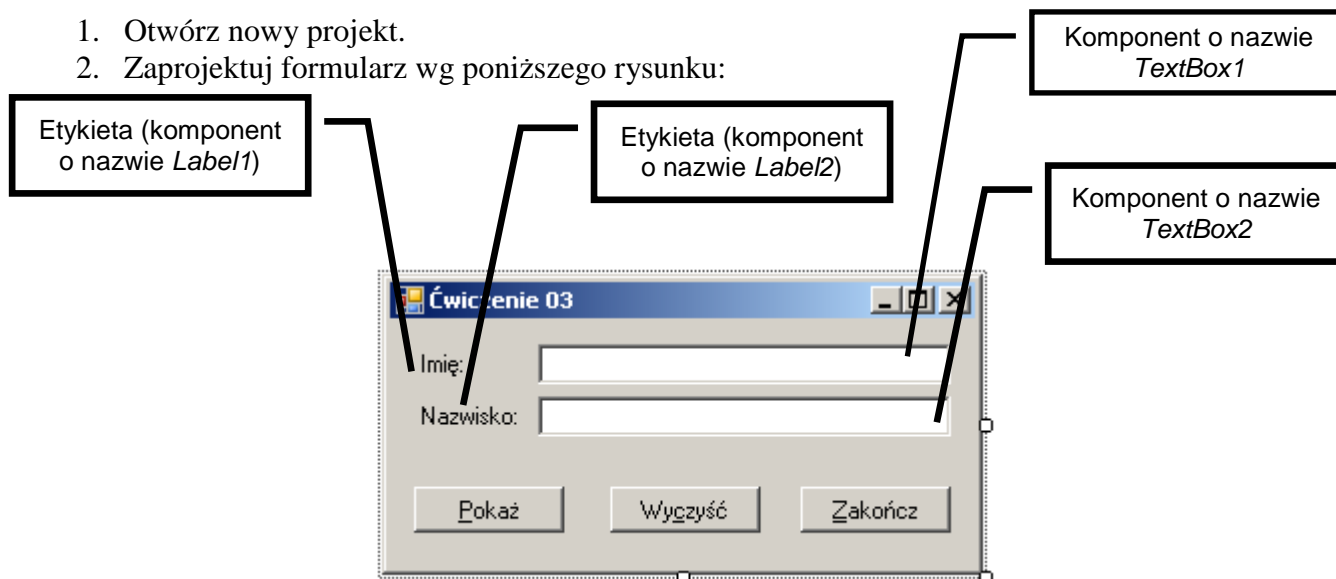
```
TextBox1.Focus
```

Co spowodowała powyższa zmiana?

2. Jaka jest różnica pomiędzy własnościami: *Name* oraz *Text*?

Ćwiczenie 3.

1. Otwórz nowy projekt.
2. Zaprojektuj formularz wg poniższego rysunku:



3. Uzupełnij kod do programu w taki sposób, aby po kliknięciu na komponencie (przycisku) *Wyczyść* znaki wprowadzone w polach tekstowych były usuwane.
4. Uzupełnij kod do programu w taki sposób, aby po kliknięciu na komponencie (przycisku) *Zakończ* program kończył swoje działanie.
5. Uzupełnij kod obsługi zdarzenia kliknięcia na komponencie (przycisku) *Pokaż* w następujący sposób:

```
MsgBox(TextBox1.Text & TextBox2.Text)
```

6. Zapisz zmiany i przetestuj działanie programu.

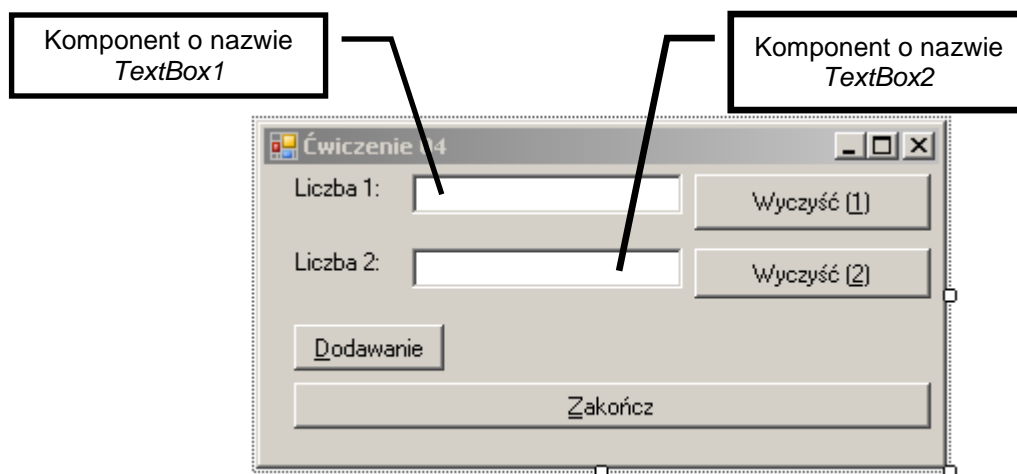
Do samodzielnego wykonania:

1. Zmodyfikuj kod z punktu 5. w taki sposób, aby znaki w wyświetlanym oknie były od siebie oddzielone (zmień argument instrukcji *MsgBox*).

Ćwiczenie 4.

Napisać program dodający dwie wprowadzone przez użytkownika liczby.

1. Otwórz nowy projekt.
2. Zaprojektuj formularz wg poniższego rysunku (nie przedstawiono nazw wszystkich komponentów):



3. Uzupełnij kod do programu w taki sposób, aby po kliknięciu na przyciskach z napisami *Wyczyść (1)* i *Wyczyść (2)* znaki wprowadzone w odpowiednich polach tekstowych były usuwane.
4. Uzupełnij kod do programu w taki sposób, aby po kliknięciu na przycisku z napisem *Zakończ* program kończył działanie.
5. Uzupełnij kod obsługi zdarzenia kliknięcia na przycisku z napisem *Dodawanie* w następujący sposób:

```

Dim liczba1 As Single
Dim liczba2 As Single
Dim wynik As Single

liczba1 = CSng(TextBox1.Text)
liczba2 = CSng(TextBox2.Text)

wynik = liczba1 + liczba2

MsgBox(wynik, , "Wynik dodawania:")

```

6. Dodaj na początku kodu programu (1. linia) następujące instrukcje:

```
Option Explicit
```

7. Zapisz zmiany i przetestuj działanie programu.

Do samodzielnego wykonania:

1. Uzupełnij formularz o dodatkowe komponenty (przyciski: Odejmowanie, Mnożenie, Dzielenie).
2. Dodaj kod programu (obsługa kliknięcia na przyciskach z napisami Odejmowanie, Mnożenie i Dzielenie).

Ćwiczenie 5.

Napisz program obliczający miejsca zerowe równania:

$$ax^2 + bx + c = 0$$

Formularz zaprojektuj zgodnie z poniższym rysunkiem:

Ćwiczenie 6.

Napisz program wykonujący 4 działania na liczbach zespolonych. Formularz zaprojektuj zgodnie z poniższym rysunkiem:

The image shows a screenshot of a Visual Basic form titled "Liczby zespolone". The form is divided into two main sections, each enclosed in a separate frame (GroupBox). The left frame, labeled "Liczba 1.", contains two text boxes: "Re:" and "Im:". The right frame, labeled "Liczba 2.", also contains two text boxes: "Re:" and "Im:". Below these input sections are four buttons representing arithmetic operations: "+", "-", "*", and "/". At the bottom of the form is a large button labeled "Zakończ". Two callout boxes with arrows point to the frames: "Ramka (komponent o nazwie GroupBox1)" points to the left frame, and "Ramka (komponent o nazwie GroupBox2)" points to the right frame.

Reguły działań na liczbach zespolonych ($Z_1 = \text{Re}_1 + j \text{Im}_1$, $Z_2 = \text{Re}_2 + j \text{Im}_2$):

$$Z_1 + Z_2 = (\text{Re}_1 + \text{Re}_2) + j(\text{Im}_1 + \text{Im}_2)$$

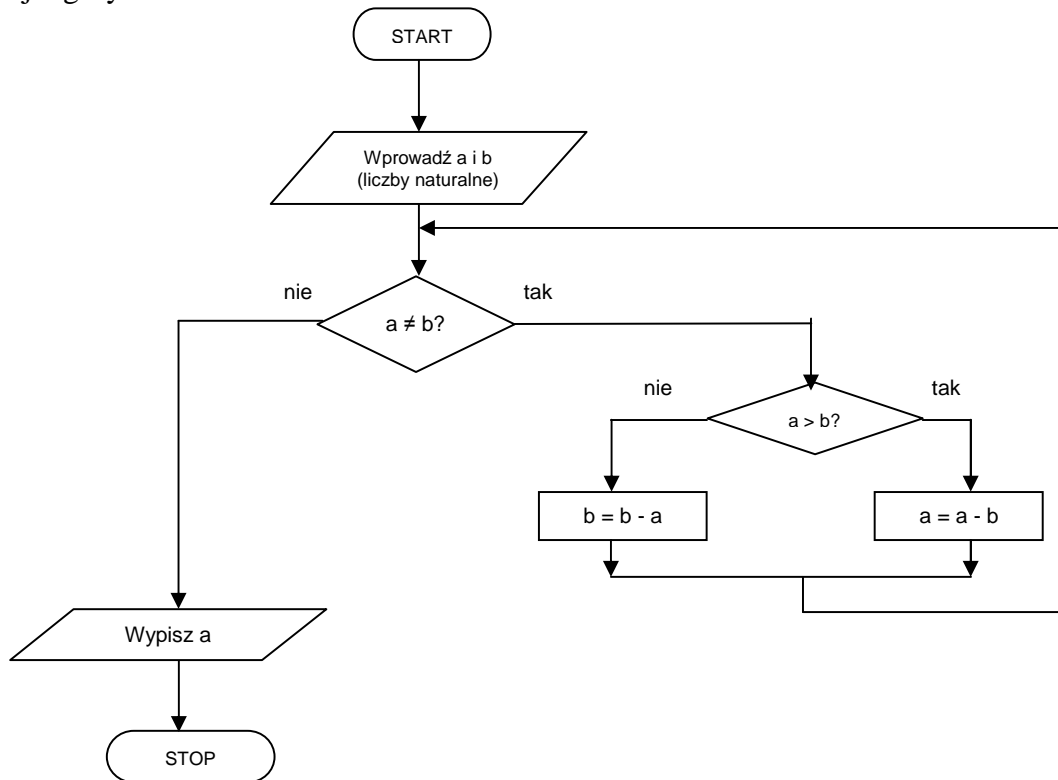
$$Z_1 - Z_2 = (\text{Re}_1 - \text{Re}_2) + j(\text{Im}_1 - \text{Im}_2)$$

$$Z_1 Z_2 = (\text{Re}_1 \text{Re}_2 - \text{Im}_1 \text{Im}_2) + j(\text{Re}_1 \text{Im}_2 + \text{Re}_2 \text{Im}_1)$$

$$\frac{Z_1}{Z_2} = \frac{\text{Re}_1 \text{Re}_2 + \text{Im}_1 \text{Im}_2}{\text{Re}_2^2 + \text{Im}_2^2} + j \frac{\text{Re}_2 \text{Im}_1 - \text{Re}_1 \text{Im}_2}{\text{Re}_2^2 + \text{Im}_2^2}; \text{Re}_2, \text{Im}_2 \neq 0$$

Ćwiczenie 7.

Napisz program obliczający największy wspólny dzielnik dwóch liczb zgodnie z podanym poniżej algorytmem:



Formularz zaprojektuj samodzielnie. Wynik wyprowadź za pomocą instrukcji *MsgBox*.

Ćwiczenie 8.

Napisz program przedstawiający postać dwójkową wprowadzonej przez użytkownika liczby naturalnej (w systemie dziesiętnym). Formularz zaprojektuj samodzielnie. Zabezpiecz program przed wprowadzeniem błędnych danych. Wynik wyprowadź za pomocą instrukcji *MsgBox*.

Ćwiczenie 9.

Dana jest pewna kwota pieniędzy (np. 1236 PLN). Załóżmy, że dysponujemy banknotami o nominałach: 200, 100, 50, 20 i 10 PLN oraz monetami o nominałach 5, 2 i 1 PLN. Napisz program, umożliwiający wypłacenie podanej przez użytkownika kwoty (tylko złote, bez groszy) przy użyciu jak najmniejszej liczby banknotów i monet. Formularz zaprojektuj samodzielnie. Zabezpiecz program przed wprowadzeniem błędnych danych. Wynik wyprowadź za pomocą instrukcji *MsgBox*.

Ćwiczenie 10.

Napisz program wypisujący w kolejności rosnącej wszystkie dzielniki podanej liczby naturalnej (większej od 0). Formularz zaprojektuj samodzielnie. Zabezpiecz program przed wprowadzeniem błędnych danych. Wynik wyprowadź za pomocą instrukcji *MsgBox*.

Ćwiczenie 11.

Liczba doskonała jest to liczba naturalna, której suma dzielników (oprócz siebie samej) jest równa tej liczbie (np. 6 jest liczbą doskonałą, ponieważ $6 = 1 + 2 + 3$). Napisz program sprawdzający, czy podana przez użytkownika liczba naturalna (większa od 0) jest doskonała. Formularz zaprojektuj samodzielnie. Zabezpiecz program przed wprowadzeniem błędnych danych. Wynik wyprowadź za pomocą instrukcji *MsgBox*.

Ćwiczenie 12.

Napisz program sprawdzający, czy podana przez użytkownika liczba naturalna (większa od 0) jest pierwsza. Formularz zaprojektuj samodzielnie. Zabezpiecz program przed wprowadzeniem błędnych danych. Wynik wyprowadź za pomocą instrukcji *MsgBox*.

Ćwiczenie 13.

Dany jest ciąg liczb: 1, $-1/2$, $1/3$, $-1/4$, ..., $1/9999$, $-1/10000$. Napisz program wyliczający sumę wyrazów tego ciągu tak, aby:

- sumowane były wszystkie wyrazy od lewej do prawej,
- sumowane były wszystkie wyrazy od prawej do lewej,
- sumowane były osobno dodatnie i osobno ujemne wyrazy od lewej do prawej,
- sumowane były osobno dodatnie i osobno ujemne wyrazy od prawej do lewej.

Wyniki wyprowadź w taki sposób, aby możliwe było ich porównanie. Czy uzyskane wyniki są identyczne?

Ćwiczenie 14.

Dane są dwa wektory:

$$\begin{aligned}x &= (10^{20}, 1223, 10^{18}, 10^{15}, 3, -10^{12}) \\y &= (10^{20}, 2, -10^{22}, 10^{13}, 2111, 10^{16})\end{aligned}$$

Obliczyć wartość iloczynu skalarnego tych wektorów.

Ćwiczenie 15.

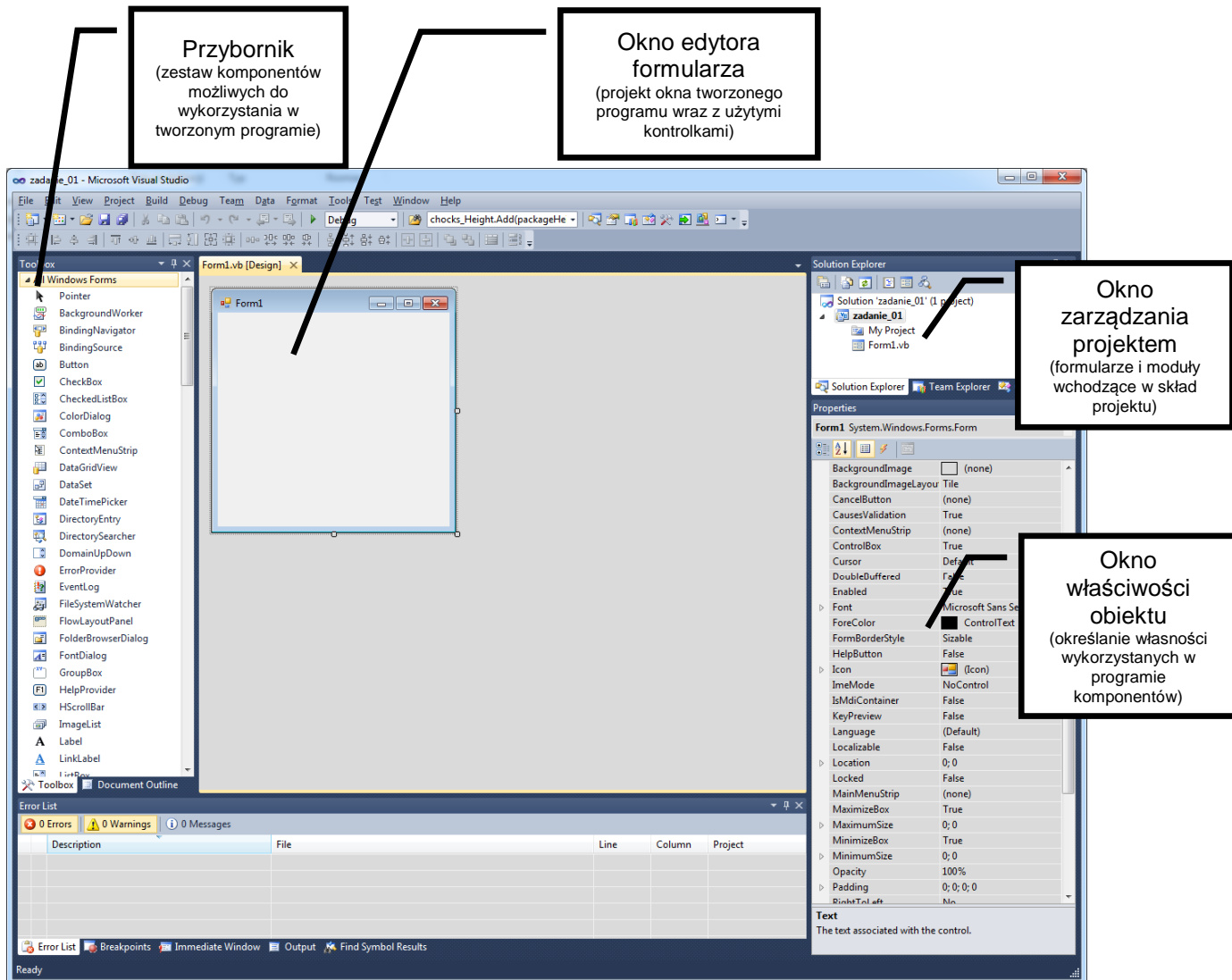
Napisz program znajdujący wszystkie liczby pierwsze mniejsze od $N = 10000$. Zadanie zrealizuj przy pomocy algorytmu *sita Eratostenesa*.

Ćwiczenie 16.

Zmodyfikuj program z ćwiczenia 15. w taki sposób, aby znajdowane były wszystkie liczby pierwsze mniejsze lub równe od N (wartość N wprowadzana przez użytkownika).

Załącznik 1.

Widok okna programu *MS Visual Studio 2010* (opcja tworzenia programów w języku *Visual Basic*).



Załącznik 2.

Zmienne w Visual Basic

1. Ogólny schemat deklaracji zmiennych:

```
Dim nazwa_zmiennej As typ_zmiennej
```

Przykłady:

```
Dim liczba_calkowita As Integer  
Dim liczbaRzeczywista As Single  
Dim wartosc_logiczna As Boolean  
Dim ciagZnakow As String
```

2. Typy danych

Typ danych	Wartość	Zakres	Wielkość pamięci
Byte	całkowita	od 0 do 255	1 B
Integer	całkowita	od -32768 do 32767	2 B
Long	całkowita	od -2 247 483 648 do 2 147 483 647	4 B
Single	zmiennoprzecinkowa pojedynczej precyzji	od -3.402823e38 do 3.402823e38	4 B
Double	zmiennoprzecinkowa podwójnej precyzji	od -0.79769313486232e308 do 1.79769313486232e308	8 B
String	ciąg znaków	od 0 do $2 \cdot 10^9$ znaków (Win 9x/2x)	10 B + długość ciągu
Boolean	logiczna	True lub False	1 B
Currency	walutowa (z ustaloną kropką dziesiętną)	od -922 337 203 685 477.5808 do 922 337 203 685 477.5807	8 B
Date	data	od 01.01.0100 do 31.01.9999	8 B
Object	adres obiektu	referencja do obiektu	4 B
Variant	każdy z powyższych typów	zgodnie z powyższymi typami	16 B (dane liczbowe) 22 B + długość ciągu (ciągi znaków)

Uwagi:

1. Nazwa zmiennej musi zaczynać się od litery; poniższa deklaracja jest niepoprawna:

```
Dim 5a As Integer
```

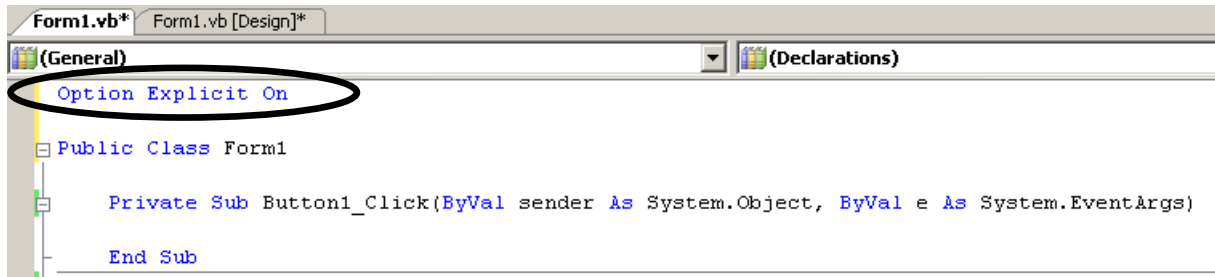
2. Istnieje możliwość deklaracji kilku zmiennych w jednej linii.

Przykład:

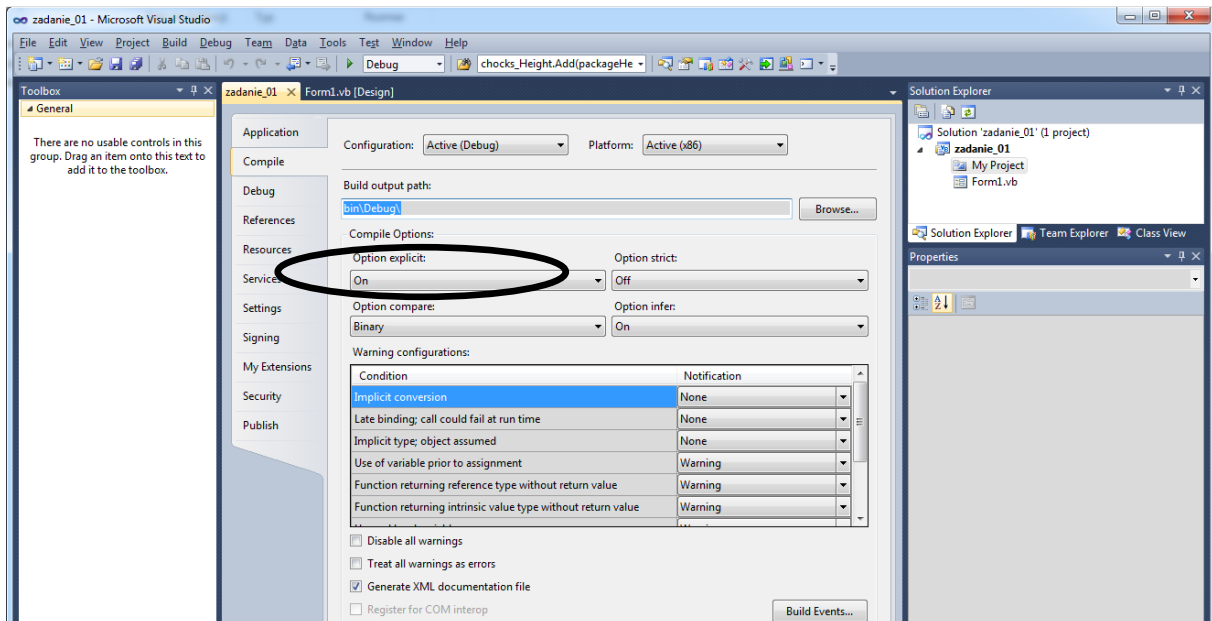
```
Dim i, j, k As Integer
```

Powyższa deklaracja jest deklaracją trzech zmiennych: *i, j* oraz *k* typu *Integer*.

3. Standardowo (domyślnie) deklaracja zmiennych w VB .NET jest wymagana (w przeciwieństwie do poprzednich wersji języka, tzn. do Visual Basic 6). Dodatkowo wymuszenie deklaracji zmiennych można osiągnąć przez:
 - a. klauzulę *Option Explicit On* na początku kodu programu (pierwsza linia):



- b. zaznaczenie opcji *On* (*My Project* → *Compile* → *Compile Options* → *Option Explicit*):



Załącznik 3.

Instrukcja warunkowa

Instrukcja warunkowa określa wykonanie pewnej grupy (jednej, kilku, kilkunastu...) instrukcji języka w zależności od spełnienia określonego warunku.

Postaci instrukcji warunkowej:

```
a)
If warunek Then
  instrukcja_1
  instrukcja_2 } grupa instrukcji wykonywanych w przypadku spełnienia warunku
  ...
  instrukcja_n }
End If
```

Przykład:

```
If dzielnik <> 0 Then
  wynikDzielenia = dzielna / dzielnik
End If
```

b)

```
If warunek Then
  instrukcja_1
  instrukcja_2 } grupa instrukcji wykonywanych w przypadku spełnienia warunku
  ...
  instrukcja_n }
Else
  instrukcja_1
  instrukcja_2 } grupa instrukcji wykonywanych w przypadku niespełnienia warunku
  ...
  instrukcja_n }
End If
```

Przykład:

```
If dzielnik <> 0 Then
  wynikDzielenia = dzielna / dzielnik
Else
  MsgBox "Dzielenie przez 0!"
End If
```

Załącznik 4.

Tablice w Visual Basic

1) Ogólny schemat deklaracji tablic:

```
Dim nazwa_tablicy(zakres) As typ_zmiennej
```

Deklaracja tablicy o liczbie elementów równej $zakres+1$ (indeksy mają wartość od 0 do $zakres$)

Przykłady:

```
Dim liczba(10) As Integer
```

Deklaracja 11-elementowej tablicy *liczba* zawierające liczby typu *Integer* – kolejne elementy tablicy: *liczba(0)*, *liczba(1)*, *liczba(2)*, ..., *liczba(9)*, *liczba(10)*.

```
Dim macierz(0 To 100) As Single
```

Deklaracja 101-elementowej tablicy *macierz* zawierające liczby typu *Single* – kolejne elementy tablicy: *macierz(0)*, *macierz(1)*, ..., *macierz(99)*, *macierz(100)*.

2) Tablice deklarowane dynamicznie:

a) deklaracja:

```
Dim nazwa_tablicy() As typ_zmiennej
```

b) dynamiczna zmiana wielkości tablicy (określenie zakresu w kodzie programu):

```
ReDim nazwa_tablicy(zakres)
```

Przykład:

```
Dim tablica() As Double
...
ReDim tablica(50)
...
ReDim tablica(200)
```