Wykład 3
Typy danych w C++/CLI, typy fundamentalne, operacje wejścia/wyjścia, właściwości klasy (property)

Zofia Kruczkiewicz

Zagadnienia

1. Elementarne typy danych C++/CLI
2. Operacje wejścia/wyjścia w programach konsolowych CLR
3. Definicja i znaczenie property (właściwości)
4. Zastosowanie w postaci biblioteki typu dll klasy TKalkulator_2 opartej na properties w projekcie typu CLR Console Application
5. Zastosowanie klasy TKalkulator_2 opartej na properties za pomocą pliku nagłówkowego w projekcie typu C++/CLI Windows Forms

1. Elementarne typy danych (1)

Visual C++ type		.NET Framework type
bool	(1 bajt)	**System.Boolean**
signed char	(1 bajt)	**System.SByte**
unsigned char	(1 bajt)	**System.Byte**
wchar_t	(2 bajty)	**System.Char**
double, long double	(8 bajtów)	**System.Double**
float	(4 bajty)	**System.Single**
int, **signed int**, **long**, **signed long**	(4 bajty)	**System.Int32**
unsigned int, **unsigned long**	(4 bajty)	**System.UInt32**
__int64, **signed __int64**	(8 bajtów)	**System.Int64**
unsigned __int64	(8 bajtów)	**System.UInt64**
short, **signed short**	(2 bajty)	**System.Int16**
unsigned short	(2 bajty)	**System.UInt16**
void		**System.Void**

Elementarne typy danych (2)

Równoważne definicje	
Definicje za pomocą typów fundamentalnych (są wtedy mapowane do typów klas wartości w CLR). Preferowane w programach C++/CLI – uniezależniają od konkretnej implementacji CLI	**Definicje za pomocą typów C++/CLR (klasy wartości)**
int ile = 0; **double** arg1 = 0;	**System::Int32** ile = 0; lub **System::Int64** ile = 0; **System::Double** arg1 = 0;
void oblicz (**int** a) { }	**System::Void** oblicz (**System::Int32** a) { } lub **System::Void** oblicz (**System::Int64** a) { }

2. Operacje wejścia/wyjścia w programach konsolowych CLR (1)

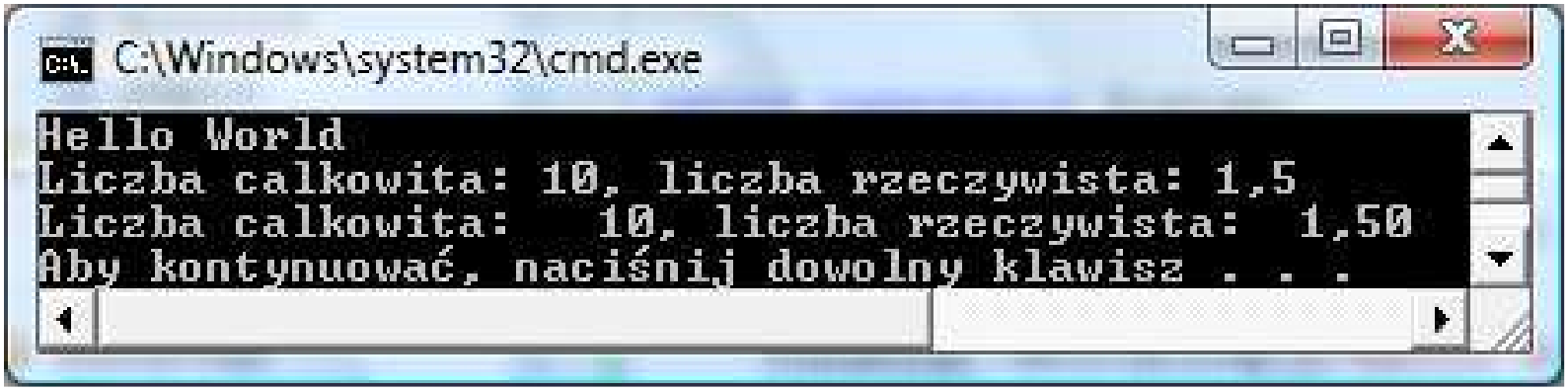
- **Operacje wyjścia**

```
#include "stdafx.h"
using namespace System;

int main(array<System::String ^> ^args)
{
 Console::WriteLine(L"Hello World");
 int a =10;
 double b = 1.5;
 Console::WriteLine(L"Liczba calkowita: {0}, liczba rzeczywista: {1}", a, b);
 Console::WriteLine(L"Liczba calkowita: {0,4},liczba rzeczywista: {1,5:F2}",a,b);
 return 0;
}
```

Numery argumentów znajdujących się za łańcuchem formatującym

Formatowanie postacie znakowej argumentów:

1) **Wyprowadzenie wartości 10 na 4-znakowym polu**
2) **Wyprowadzenie wartości 1.5 na 5-znakowym polu z 2-ma pozycjami dziesiętnymi**



Operacje wejścia/wyjścia w programach konsolowych CLR (2)

- **Operacje wejścia**

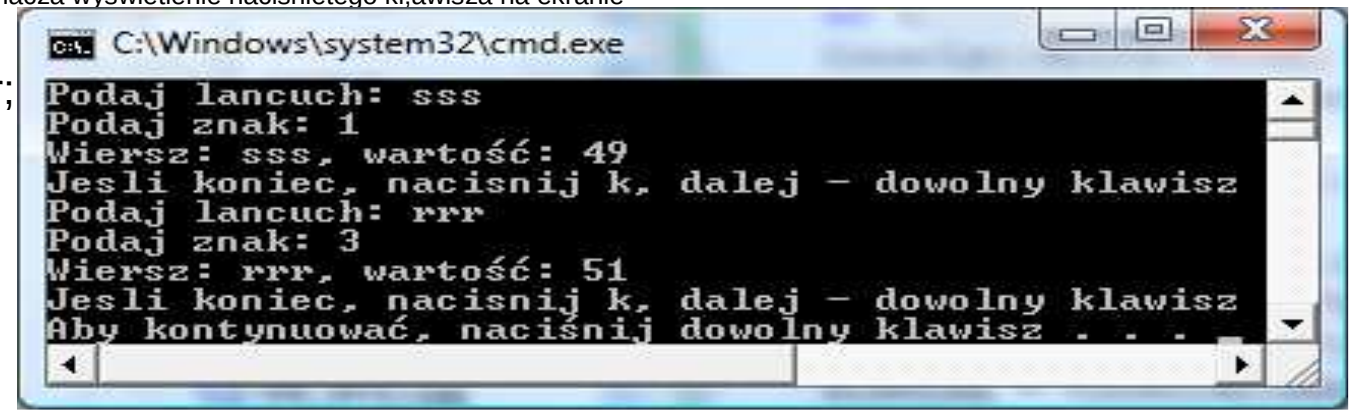
```
 // WE_WY1.cpp : main project file.
#include "stdafx.h"
using namespace System;
int main(array<System::String ^> ^args)
{    ConsoleKeyInfo klawisz;
     char pom;
     do {
     Console::Write("Podaj lancuch: ");
     String^ wiersz = Console::ReadLine();
     Console::Write("Podaj znak: ");
     char znak1 = Console::Read();
     Console::Read();
     int liczba = System::Int32::Parse(""+znak1);
     Console::WriteLine(L"Wiersz: {0}, wartość: {1}", wiersz, liczba);
     Console::WriteLine(L"Jesli koniec, nacisnij k, dalej - dowolny klawisz");
     klawisz = Console::ReadKey(true);
     // false lub brak parametru metody oznacza wyswietlenie nacisnietego kl;awisza na ekranie
     Console::Read();
     pom = klawisz.KeyChar;
     } while (pom!='k');
   return 0;
}
```

Pobranie ciągu znaków razem ze znakami klawisza Enter i utworzenie referencyjnego (class ref) obiektu typu String zawierającego wprowadzony łańcuch

Wprowadzanie pojedynczych znaków bez znaków klawisza Enter

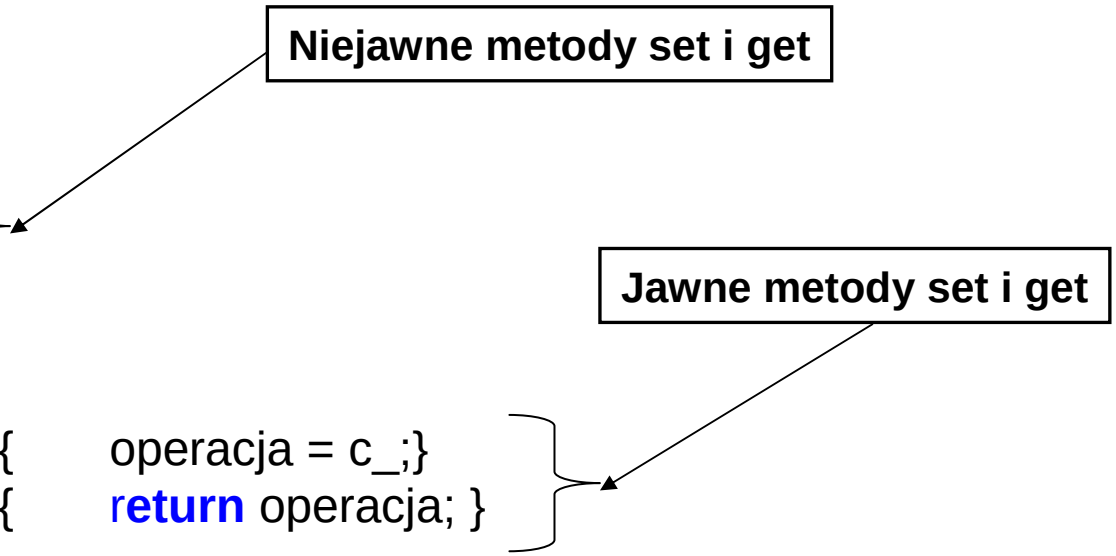
Czyszczenie bufora klawiatury przez odczytanie pozostawionych znaków – w tym wypadku po klawiszu Enter

Zwracanie naciśniętego klawisza jako obiektu klasy wartości (class value)



3. Definicja i znaczenie property (właściwości) – składowa klasy wartości lub klasy referencyjnej, która wywołuje metody, do których odwołuje się jak do pól, lecz nie są to lokalizacje pól przechowujących dane

```
// TKalkulator2.h
#pragma once
using namespace System;
namespace TKalkulator2 {
public ref class TKalkulator_2
{
    public:
    property double arg1;
    property double arg2;
    property double wynik;
    property wchar_t oper
    {
      void set (wchar_t c_)      {     operacja = c_;}
      wchar_t get()              {     return operacja; }
    }
    private: int poprawna;
             wchar_t operacja;
             literal char plus = '+';
             literal char minus = '-';
             literal char mnoz = '*';
             literal char dziel = '/';
```



Jawna metoda *get* i niejawna metoda *set* dla property *wyniki*

```
public:
   TKalkulator_2 (double a,double b, wchar_t c)
     { arg1=a;     arg2=b;      oper=c;}
   TKalkulator_2 ()
     { arg1=1;     arg2=1;      oper='+';}
property int wyniki
  {
   int get ()
   {  poprawna = 1;
      switch (oper)
     {  case plus    : wynik = arg1 + arg2;
        case minus : wynik = arg1 - arg2;
        case mnoz  : wynik = arg1 * arg2;
        case dziel    : arg2!=0 ? wynik=arg1/arg2 : poprawna=3; break;
        default: poprawna = 2;
      }
     return poprawna;  }
  }
```

Wywołanie niejawne metod set dla properties: *oper, arg1, arg2*

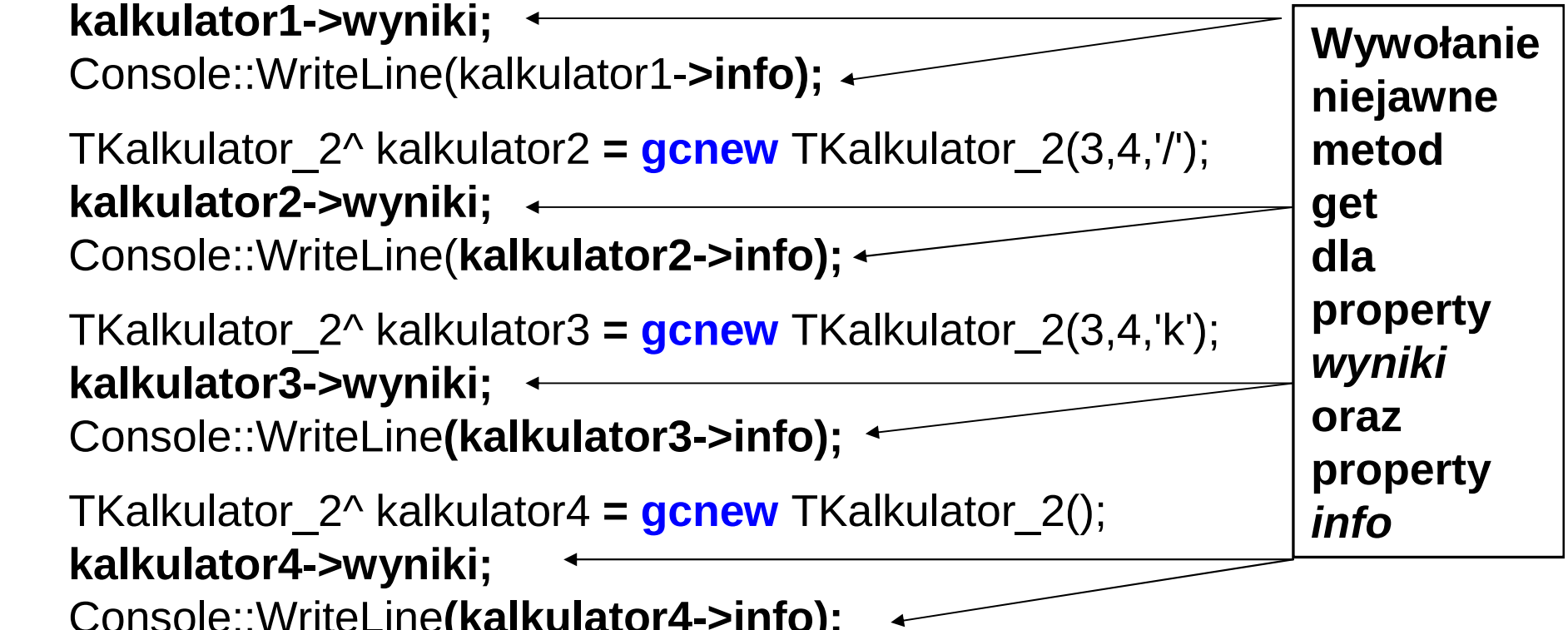
Wywołanie niejawne metod get dla properties: *oper, arg1, arg2 oraz set* dla property *wynik*

Jawna metoda *get* i niejawna metoda *set* dla property *info*

```
property String^ info
    {
     String^ get()
     {
       if (poprawna==1)
         return arg1+L" "+oper+L" "+arg2+L" = "+wynik;
       else if (poprawna==2)
         return L"Operacja "+oper+L" jest niepoprawna.";
       else
         return L"Argument 2 równy "+arg2+L" dzielenie przez zero.";
     }
  }
};
}
```

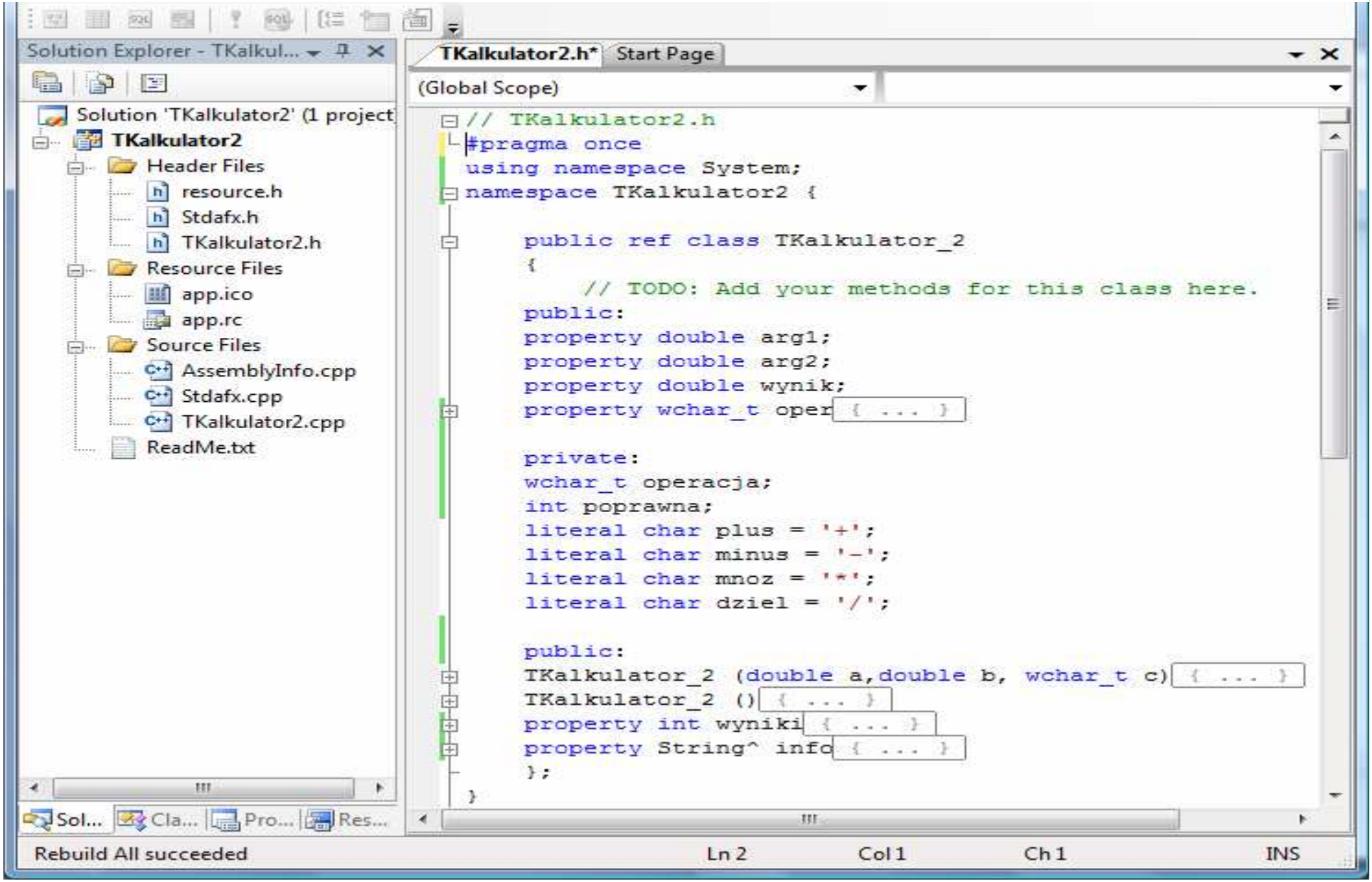
Wywołanie niejawne metod get dla properties: *oper, arg1, arg2, wyniki*

```
// Kalkulator_5.cpp : main project file.
#include "stdafx.h"
#using <TKalkulator2.dll>
using namespace System;
using namespace TKalkulator2;

int main(array<System::String ^> ^args)
{  TKalkulator_2^ kalkulator1 = gcnew TKalkulator_2(3,0,'/');
   kalkulator1->wyniki;
   Console::WriteLine(kalkulator1->info);

   TKalkulator_2^ kalkulator2 = gcnew TKalkulator_2(3,4,'/');
   kalkulator2->wyniki;
   Console::WriteLine(kalkulator2->info);

   TKalkulator_2^ kalkulator3 = gcnew TKalkulator_2(3,4,'k');
   kalkulator3->wyniki;
   Console::WriteLine(kalkulator3->info);

   TKalkulator_2^ kalkulator4 = gcnew TKalkulator_2();
   kalkulator4->wyniki;
   Console::WriteLine(kalkulator4->info);
   return 0;
}
```



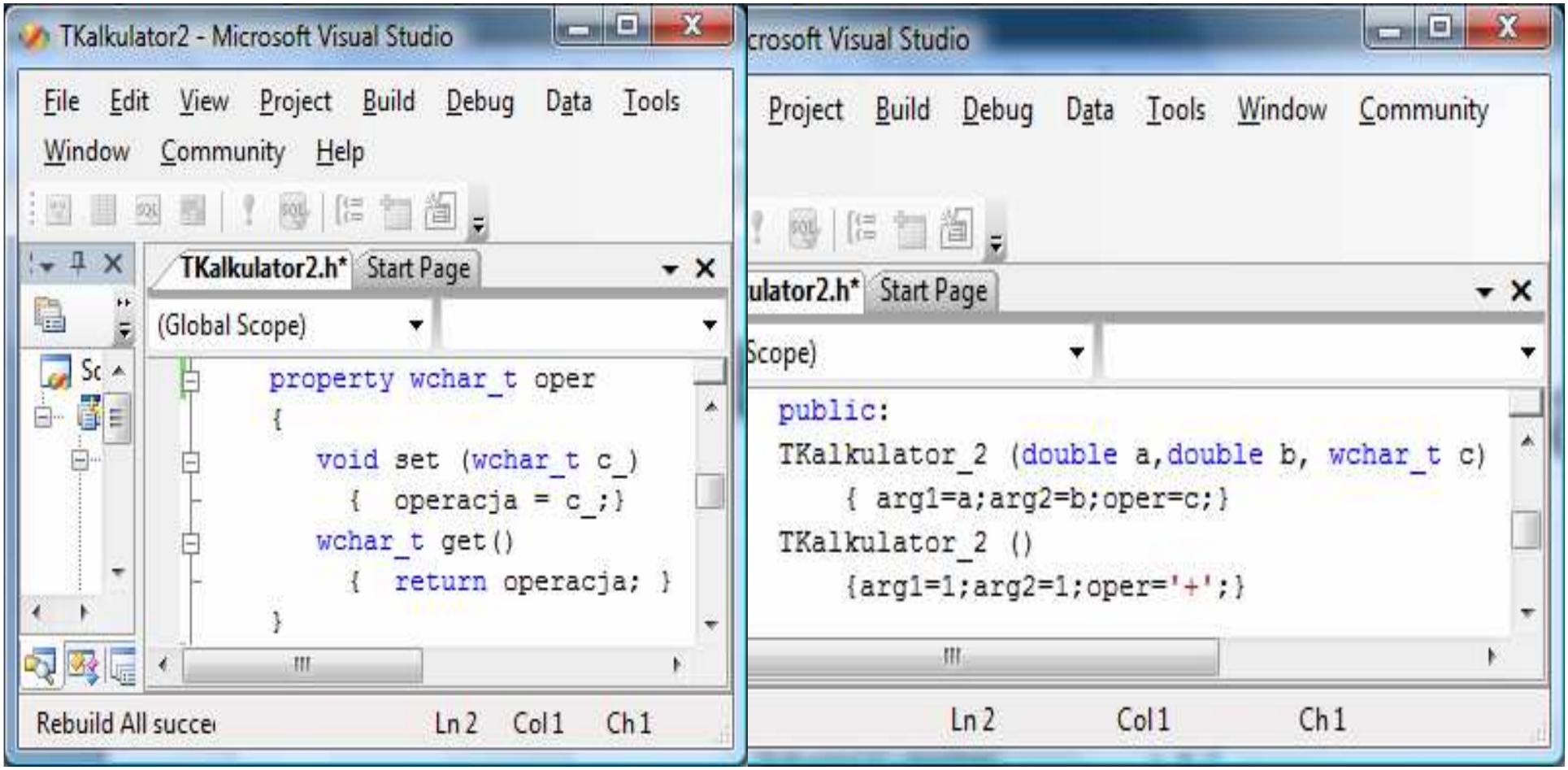
4. Zastosowanie w postaci biblioteki typu dll klasy TKalkulator_2 opartej na properties, w projekcie typu CLR Console Application

(4.1. wykonanie projektu typu Class Library)

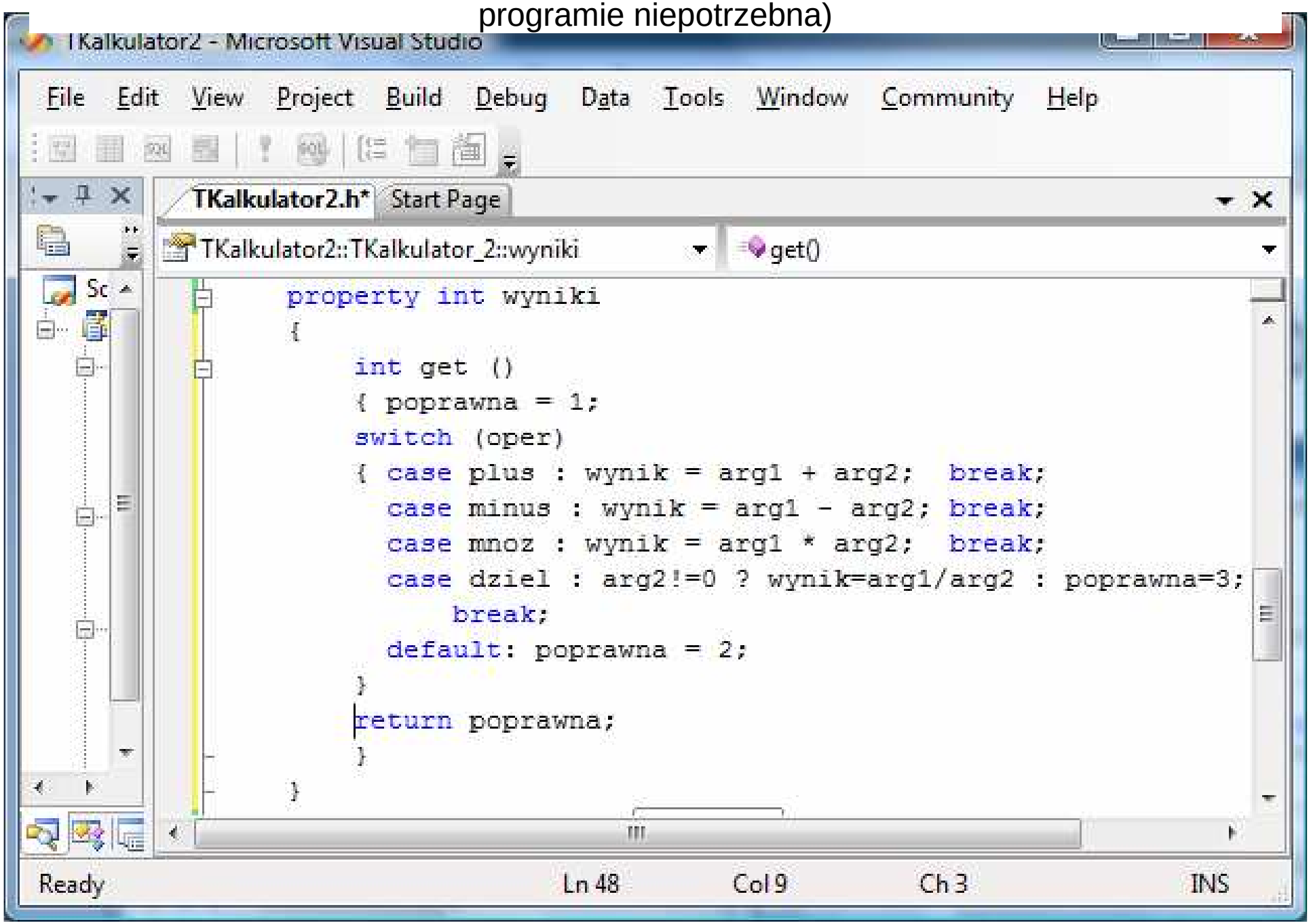


Właściwości do obsługi pola operacja

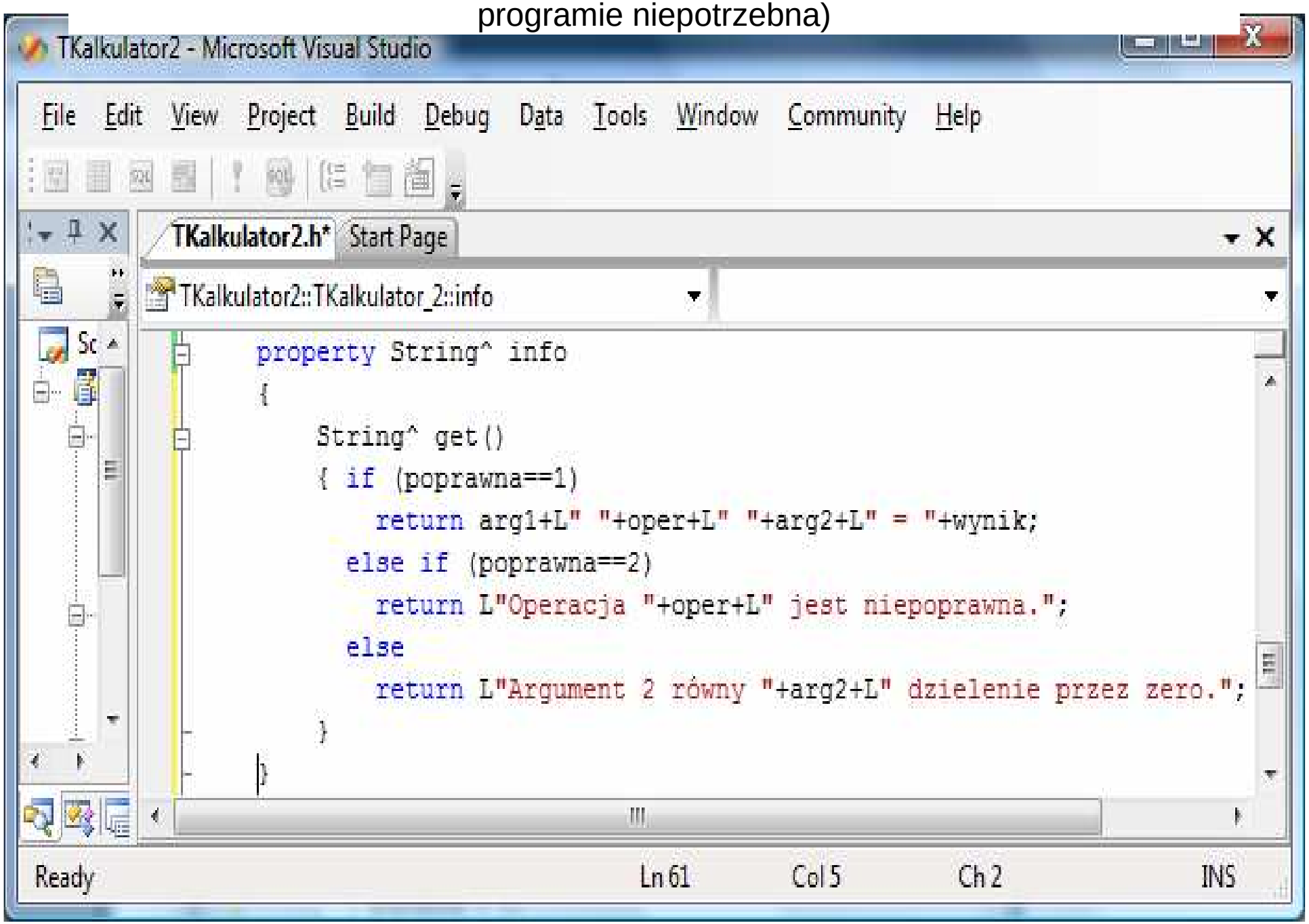
Przeciążone konstruktory



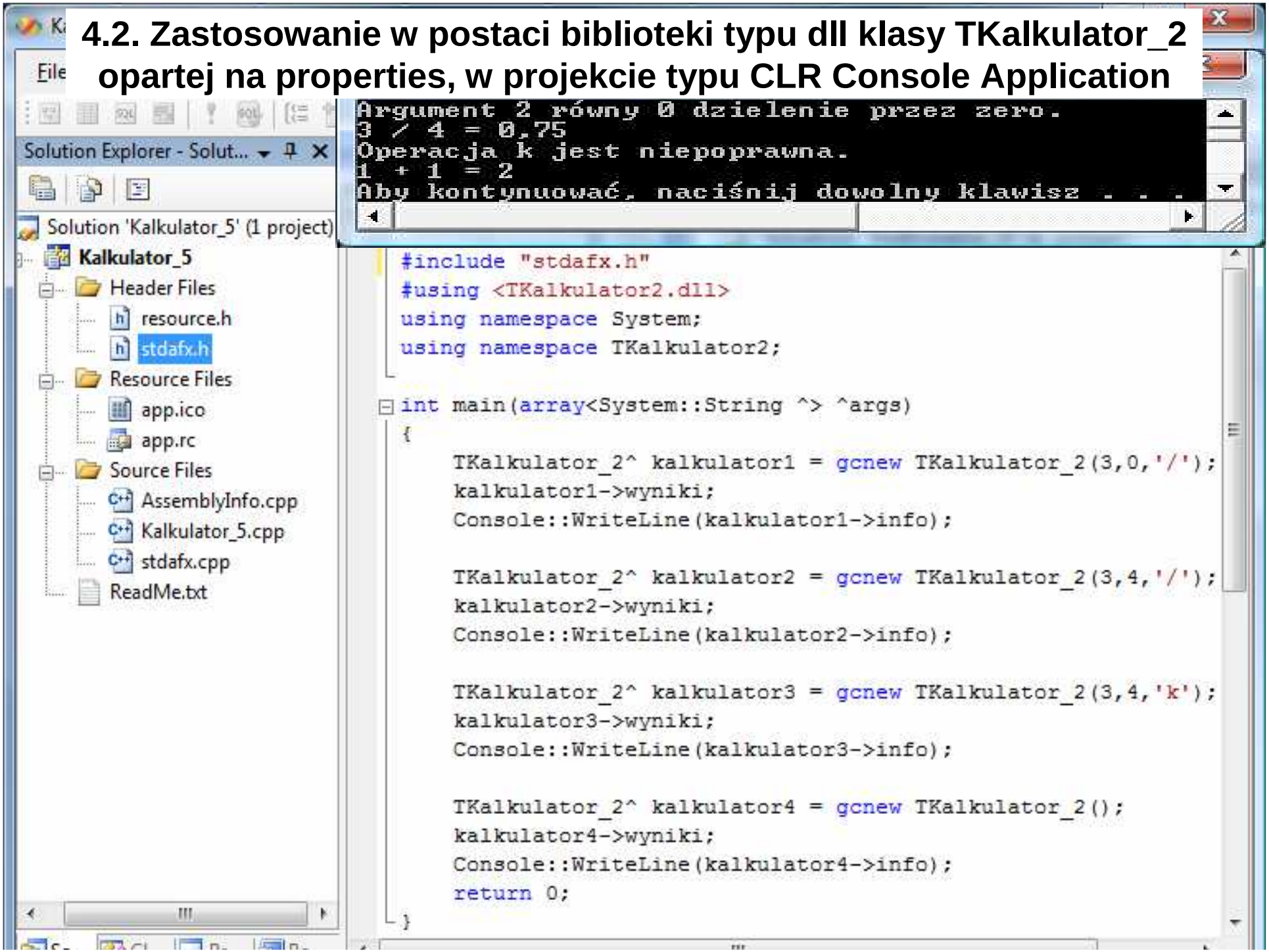
Właściwość ***wyniki***: zdefiniowano metodę ***get***, metoda ***set*** jest domyślna (w programie niepotrzebna)



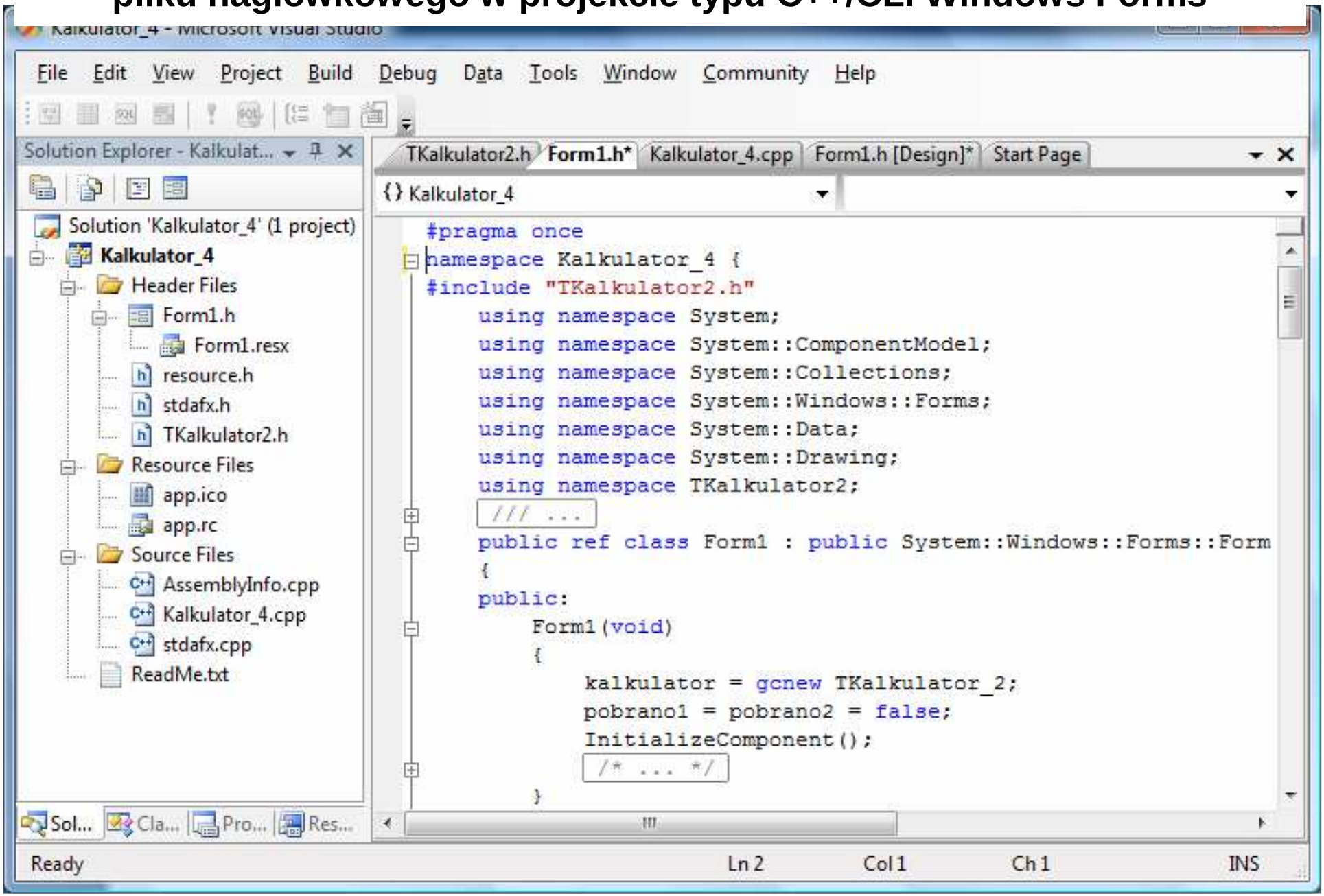
Właściwość ***info:*** zdefiniowano metodę ***get***, metoda ***set*** jest domyślna (w programie niepotrzebna)

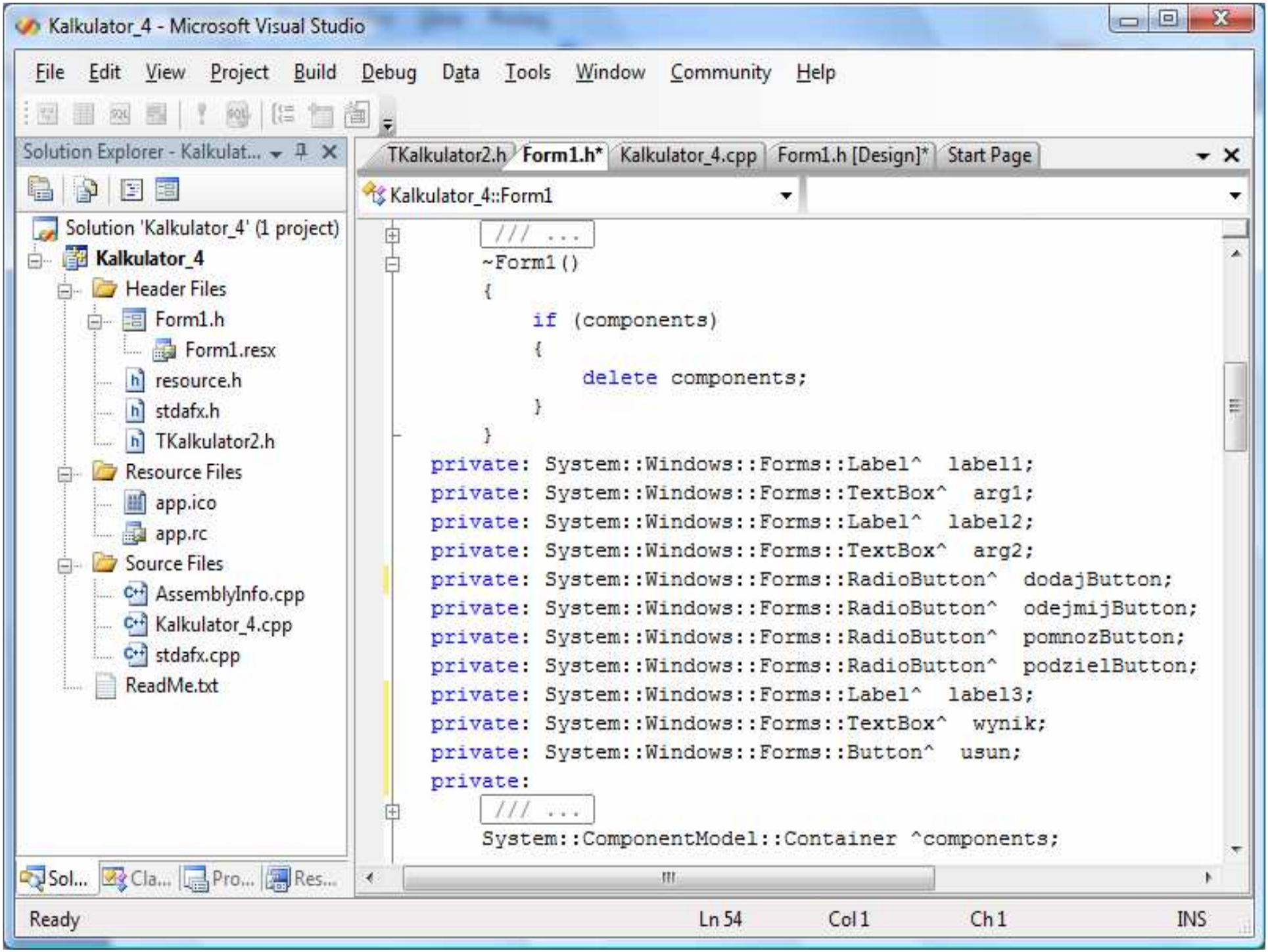


4.2. Zastosowanie w postaci biblioteki typu dll klasy TKalkulator_2 opartej na properties, w projekcie typu CLR Console Application

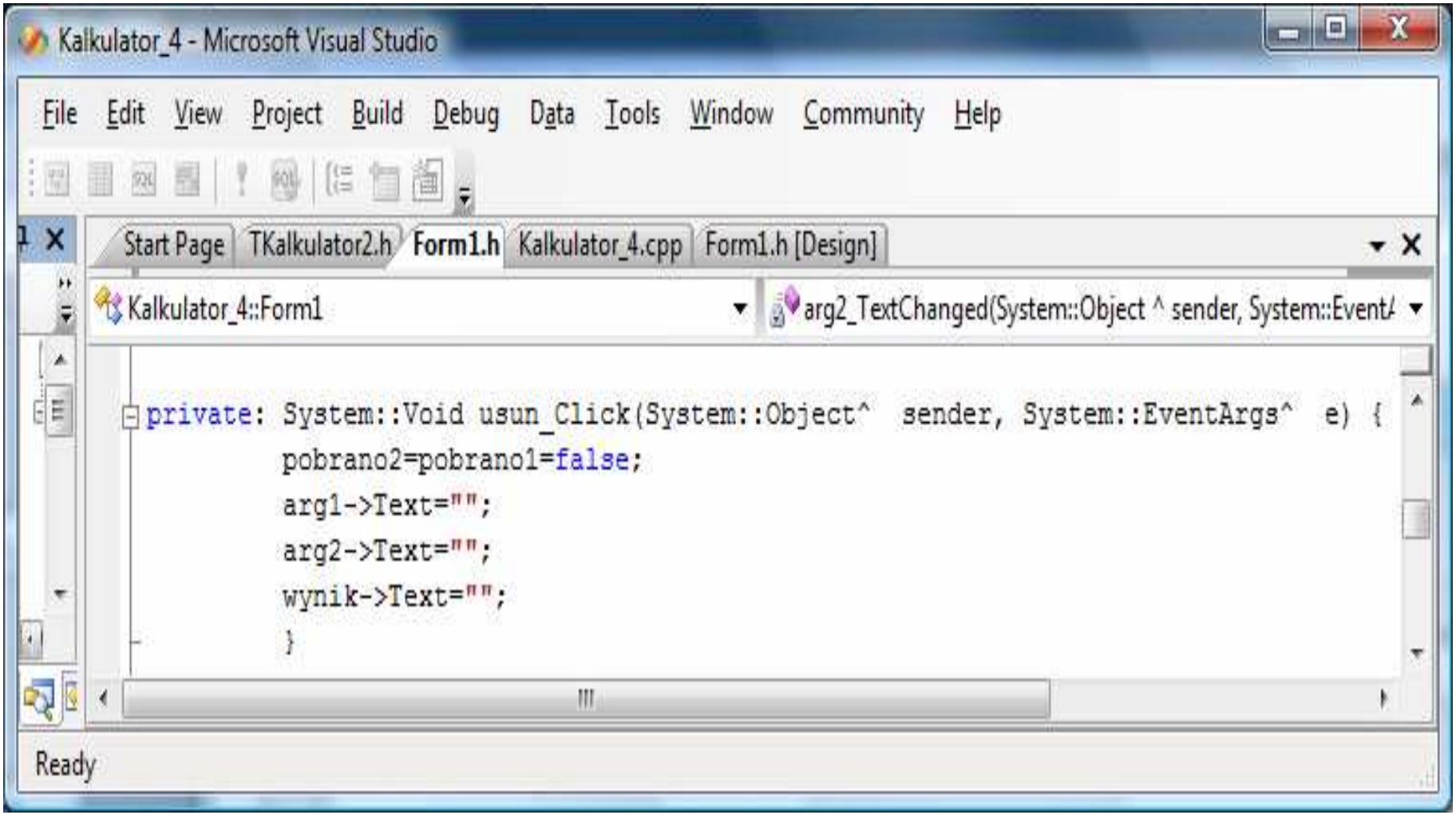


5. Zastosowanie klasy TKalkulator_2 opartej na properties za pomocą pliku nagłówkowego w projekcie typu C++/CLI Windows Forms

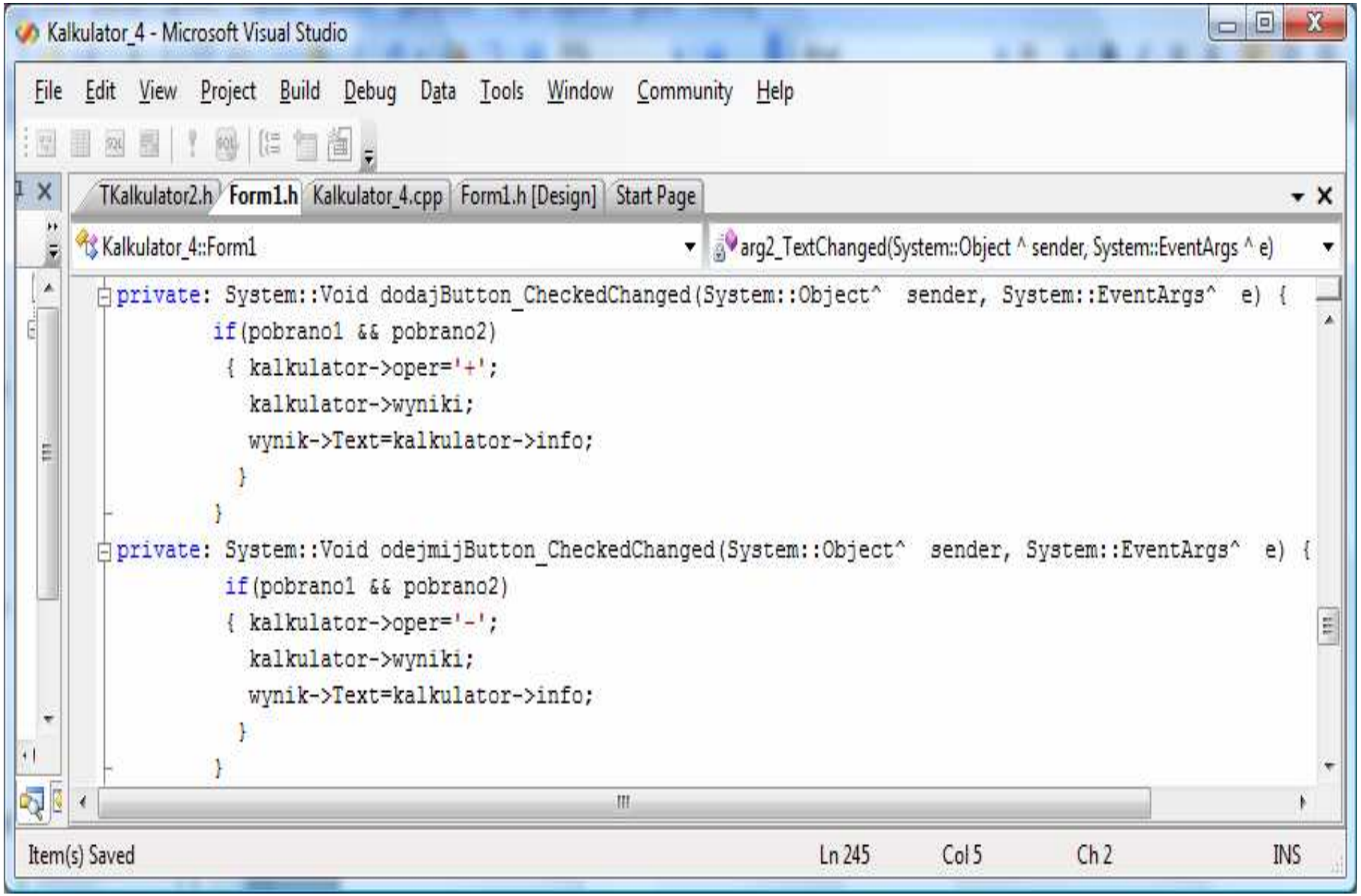



Kalkulator_4 - Microsoft Visual Studio
File Edit View Project Build Debug Data Tools Window Community Help
Solution Explorer - Kalkulat...
Solution 'Kalkulator_4' (1 project)
Kalkulator_4
Header Files
Form1.h
Form1.resx
resource.h
stdafx.h
TKalkulator2.h
Resource Files
app.ico
app.rc
Source Files
AssemblyInfo.cpp
Kalkulator_4.cpp
stdafx.cpp
ReadMe.txt
Sol... Cla... Pro... Res...
TKalkulator2.h Form1.h* Kalkulator_4.cpp Form1.h [Design]* Start Page
Kalkulator_4::Form1
/// ...
~Form1()
{
if (components)
{
delete components;
}
}
private: System::Windows::Forms::Label^ label1;
private: System::Windows::Forms::TextBox^ arg1;
private: System::Windows::Forms::Label^ label2;
private: System::Windows::Forms::TextBox^ arg2;
private: System::Windows::Forms::RadioButton^ dodajButton;
private: System::Windows::Forms::RadioButton^ odejmijButton;
private: System::Windows::Forms::RadioButton^ pomnozButton;
private: System::Windows::Forms::RadioButton^ podzielButton;
private: System::Windows::Forms::Label^ label3;
private: System::Windows::Forms::TextBox^ wynik;
private: System::Windows::Forms::Button^ usun;
private:
/// ...
System::ComponentModel::Container ^components;
Ready
Ln 54
Col 1
Ch 1
INS

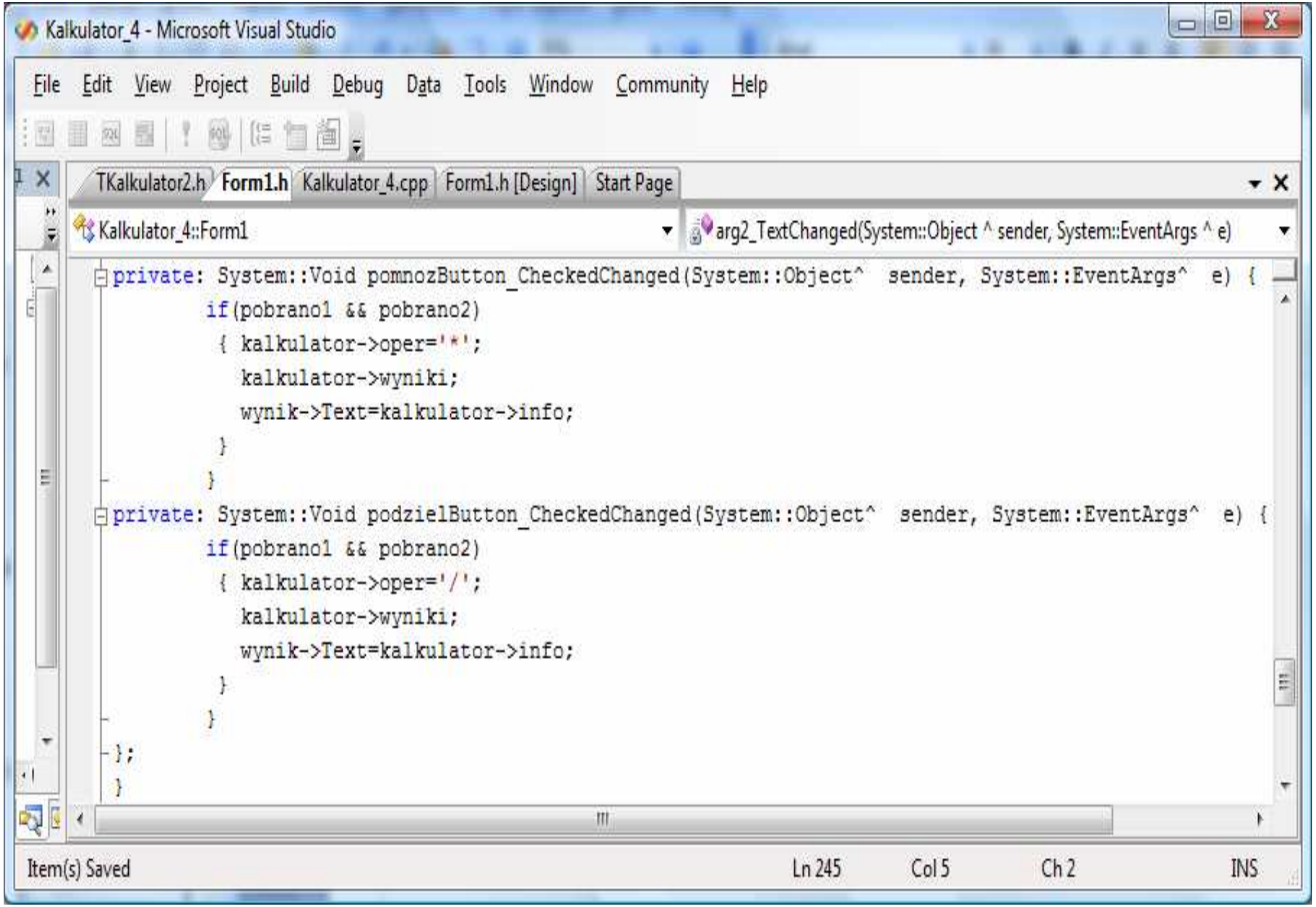
Czyszczenie pól tekstowych w oknie kalkulatora



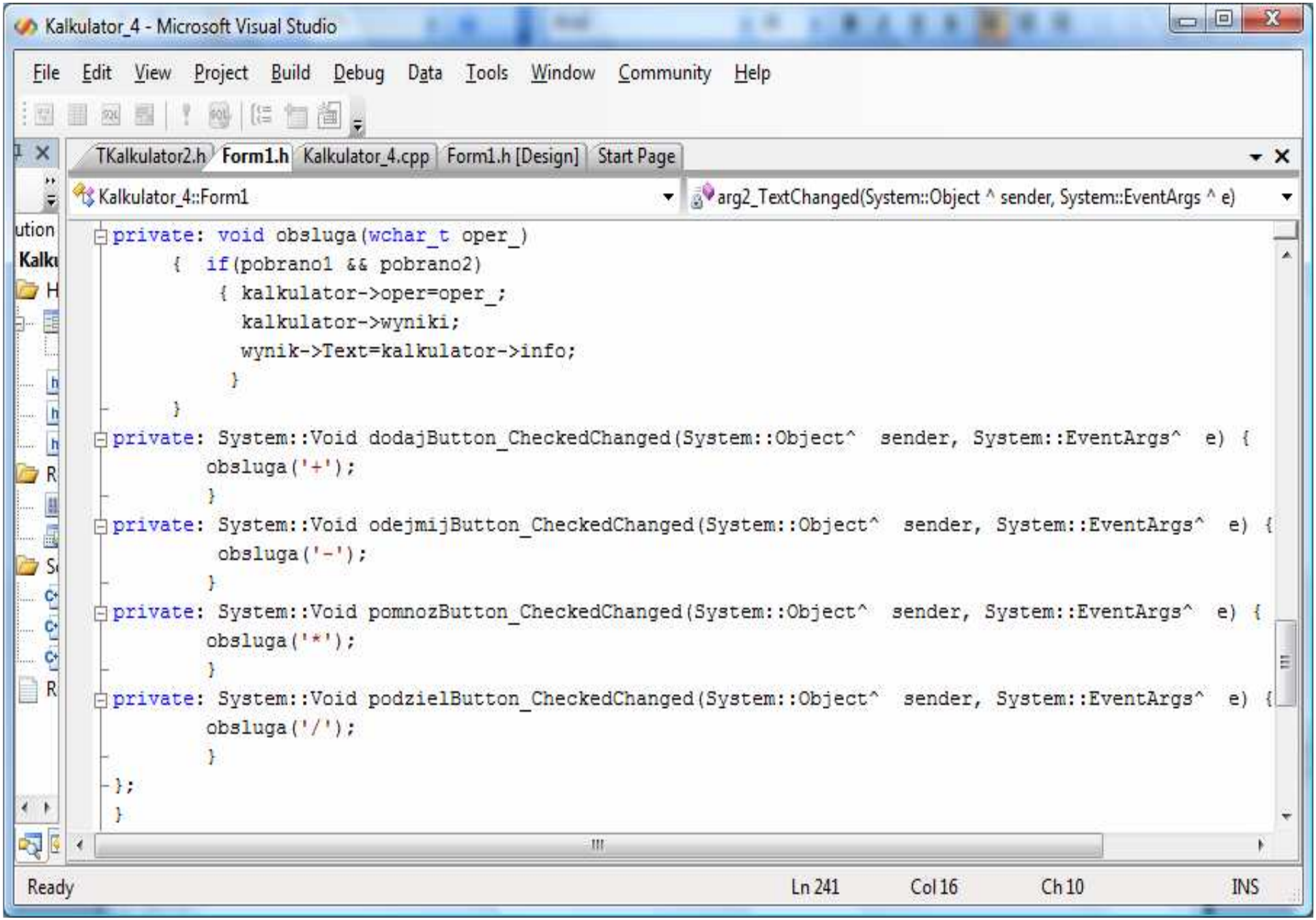
Obsługa zdarzeń wyboru operacji: ”+”, ”-”



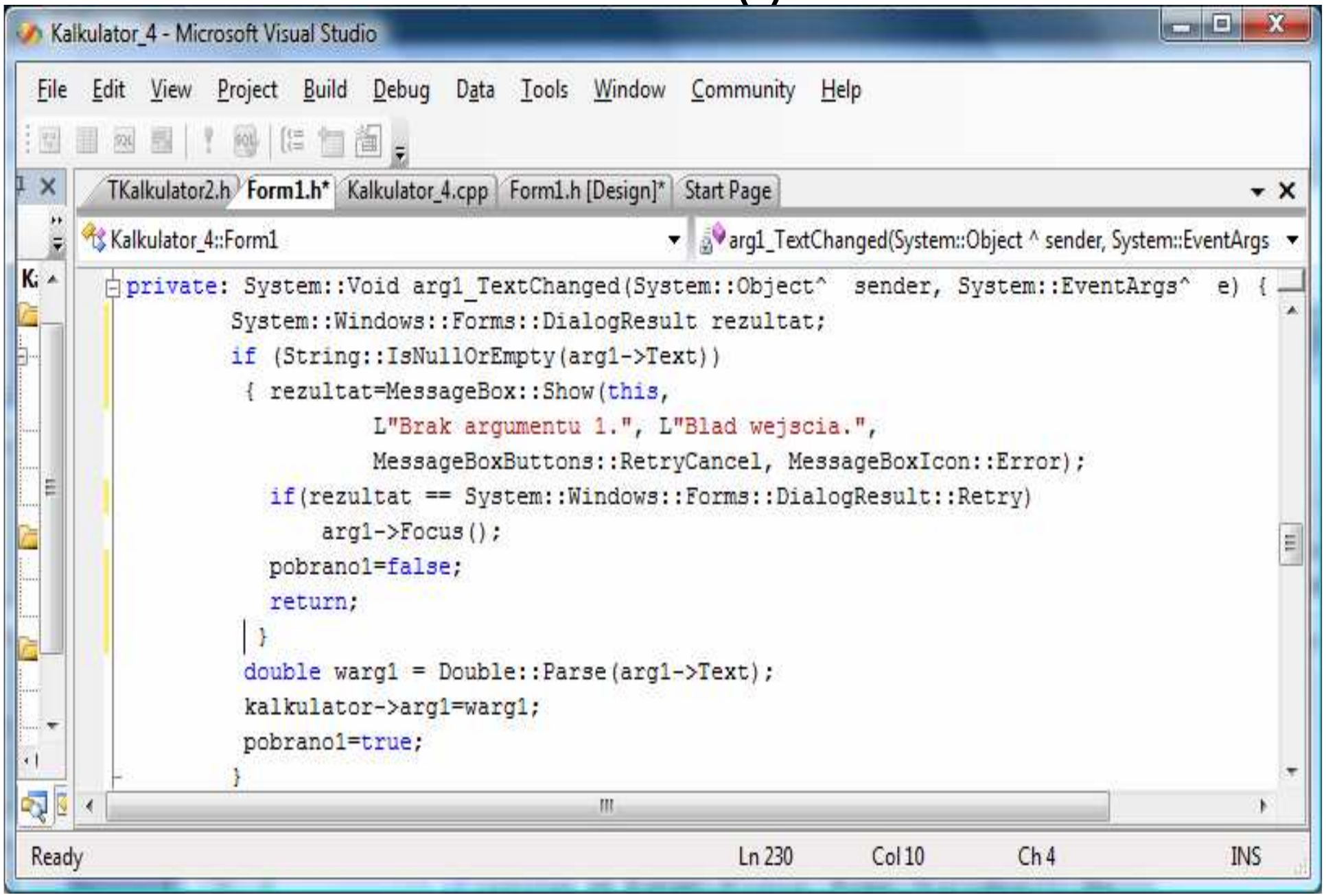
Obsługa zdarzeń wyboru operacji: ”*”, ”/”



Poprawa kodu obsługi wyboru typu operacji



5.3.Obsługa błędów wprowadzonych danych typu double – instrukcja warunkowa (1)

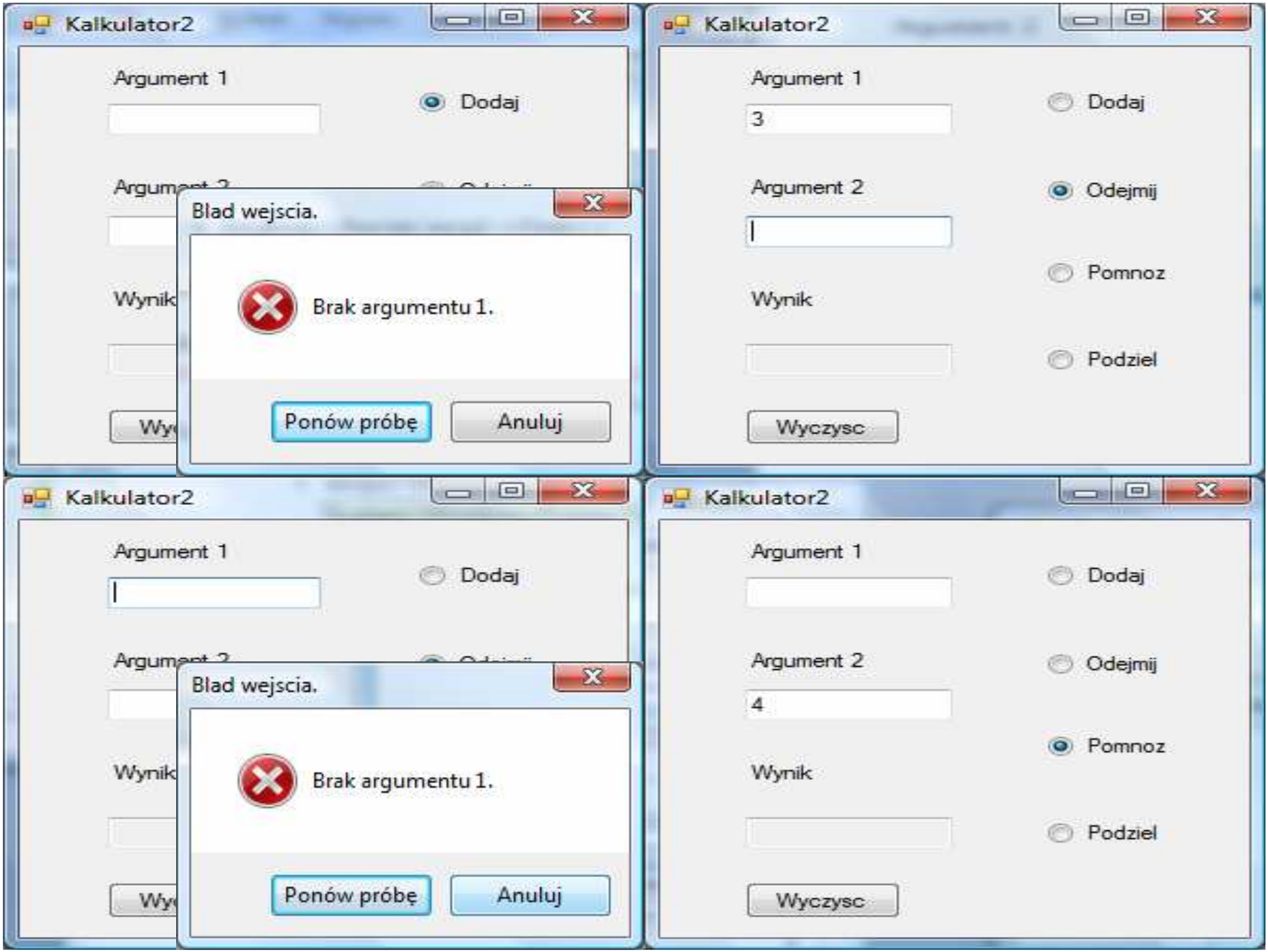


Obsługa niektórych błędów wprowadzonych danych typu double – instrukcja warunkowa (2)

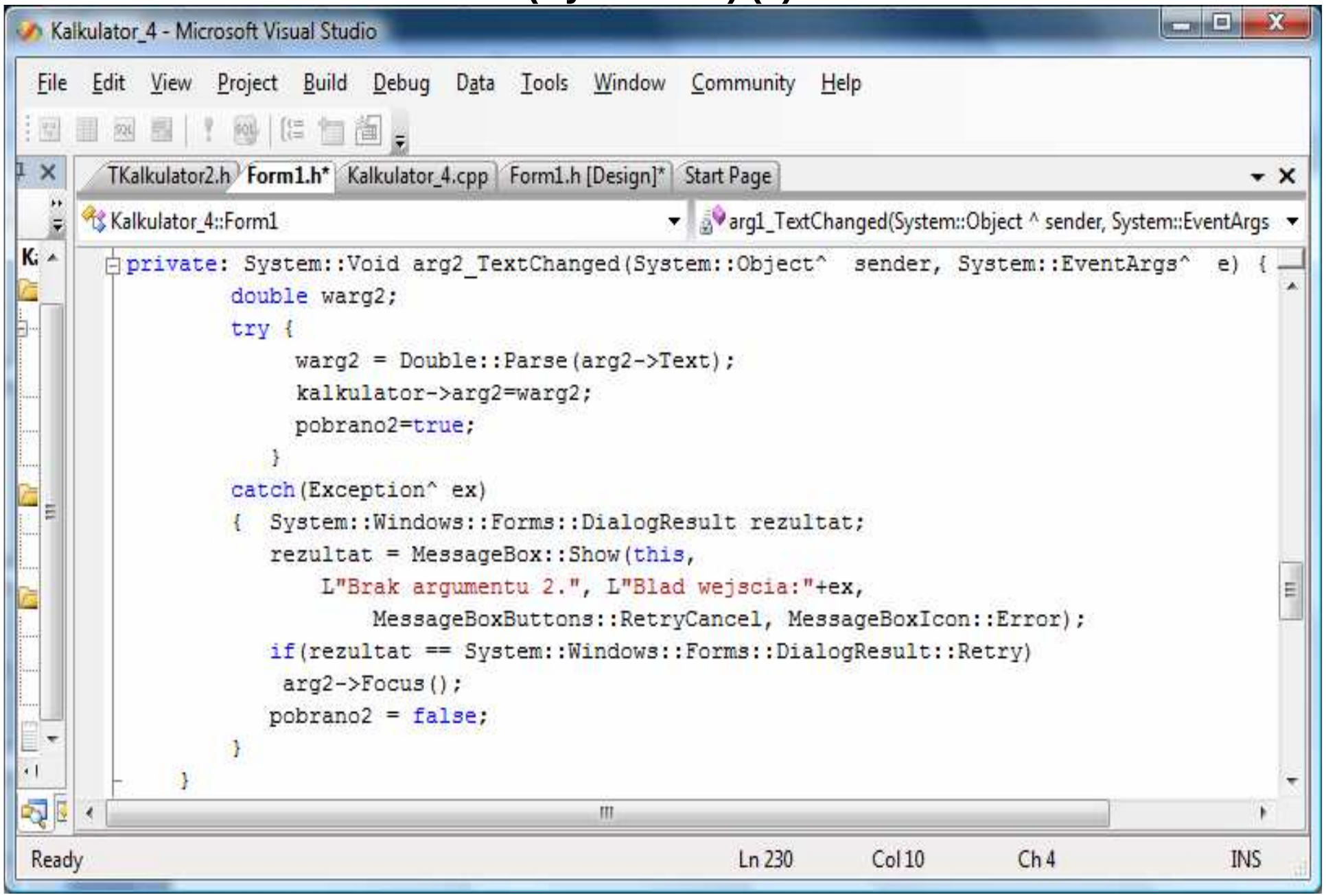
```
private: System::Void arg1_TextChanged(
    System::Object^  sender,
    System::EventArgs^  e)
{ System::Windows::Forms::DialogResult rezultat;
  if (String::IsNullOrEmpty(arg1->Text))
   { rezultat = MessageBox::Show(
       this,
       L"Brak argumentu 1.",
       L"Blad wejscia.",
       MessageBoxButtons::RetryCancel,
       MessageBoxIcon::Error);
     if(rezultat == System::Windows::Forms::DialogResult::Retry)
       arg1->Focus();
     pobrano1 = false;
     return;
    }
  double warg1 = Double::Parse(arg1->Text);
  kalkulator->arg1 = warg1;
  pobrano1 = true; }
```

W warunku instrukcji if w przypadku pustego ciągu znaków do odwzorowania na wartość double lub brak łańcucha wystąpi błąd (wynik metody String::IsNullOrEmpty **jest równy true). We wskazanym miejscu nastąpi wtedy wywołanie standardowego okienka dialogowego z klawiszami Retry i Cancel. Po naciśnięciu klawisz Retry kursor wraca do pola tekstowego, w którym wprowadzono błędne dane i nastąpi przerwanie metody.**

W przeciwnym wypadku (również w przypadku niewłaściwego ciągu znaków) nastąpi odwzorowanie wprowadzonego ciągu znaków na wartość double, które odbędzie się bez błędu tylko w wypadku poprawnych wartości wprowadzonego ciągu znaków


Kalkulator2
Argument 1
Dodaj
Argument 2
Odejmij
Pomnoz
Wynik
Podziel
Wyczysc
Blad wejscia.
Brak argumentu 1.
Ponów próbę
Anuluj
Kalkulator2
Argument 1
3
Dodaj
Argument 2
Odejmij
Pomnoz
Wynik
Podziel
Wyczysc
Kalkulator2
Argument 1
Dodaj
Blad wejscia.
Brak argumentu 1.
Ponów próbę
Anuluj
Kalkulator2
Argument 1
Dodaj
Argument 2
4
Odejmij
Pomnoz
Wynik
Podziel
Wyczysc

5.4. Obsługa błędów wprowadzonych danych typu double – wyjątki (try …catch) (1)

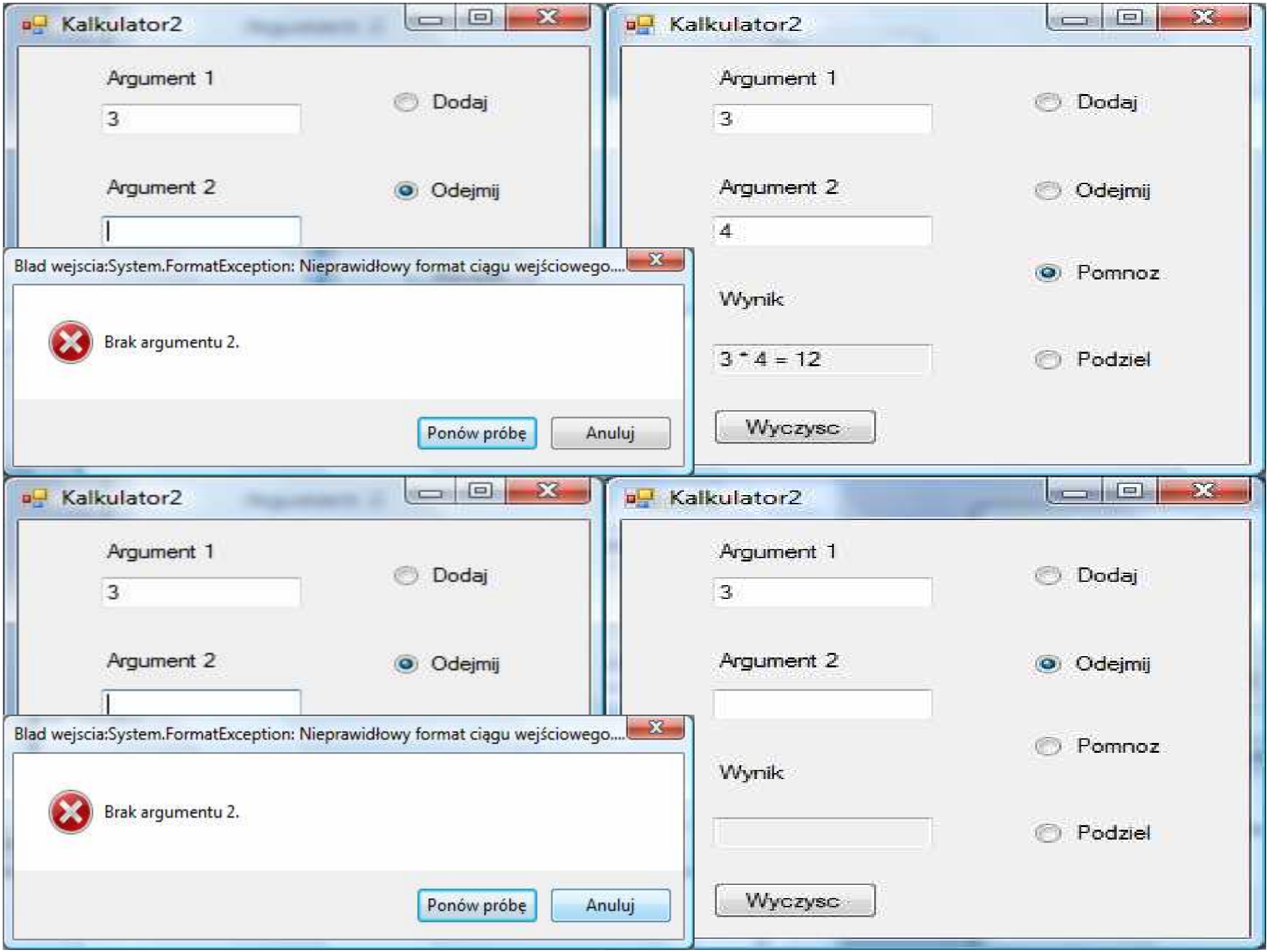


Obsługa błędów wprowadzonych danych typu double Wyjątki – bloki try i catch (2)

```
private: System::Void arg2_TextChanged(
   System::Object^  sender, System::EventArgs^  e)
{ double warg2;
  try {
       warg2 = Double::Parse(arg2->Text);
       kalkulator->arg2=warg2;
       pobrano2=true;  }
   catch(Exception^ ex)
     {  System::Windows::Forms::DialogResult rezultat;
        rezultat = MessageBox::Show(
             this, L"Brak argumentu 2.",
             L"Blad wejscia:"+ex,
             MessageBoxButtons::RetryCancel,
             MessageBoxIcon::Error);
             if(rezultat == System::Windows::Forms::DialogResult::Retry)
                arg2->Focus();
        pobrano2 = false;
     }  }
```

W bloku try w przypadku niewłaściwego ciągu znaków do odwzorowania na wartość double lub brak łańcucha wystąpi błąd w postaci obiektu referencyjnego typu pochodnego Exception. We wskazanym miejscu nastąpi wtedy przerwanie wykonania programu i przejście do najbliższego bloku catch wg dopasowania wygenerowanego wyjątku do typu wyjątku w bloku catch. Wyjątek Exception w przykładzie pasuje do każdego typu wyjątku.

Do obsługi wyjątku zastosowano standardowe okienko dialogowe z klawiszami Retry i Cancel. Po naciśnięciu klawisza Retry kursor wraca do pola tekstowego, w którym wprowadzono błędne dane.


Kalkulator2
Argument 1
3
Argument 2
Dodaj
Odejmij
Blad wejscia:System.FormatException: Nieprawidłowy format ciągu wejściowego....
Brak argumentu 2.
Ponów próbę
Anuluj
Kalkulator2
Argument 1
3
Argument 2
4
Dodaj
Odejmij
Pomnoz
Wynik
3 * 4 = 12
Podziel
Wyczysc
Kalkulator2
Argument 1
3
Argument 2
Dodaj
Odejmij
Blad wejscia:System.FormatException: Nieprawidłowy format ciągu wejściowego....
Brak argumentu 2.
Ponów próbę
Anuluj
Kalkulator2
Argument 1
3
Argument 2
Dodaj
Odejmij
Pomnoz
Wynik
Podziel
Wyczysc