

Wykład 6

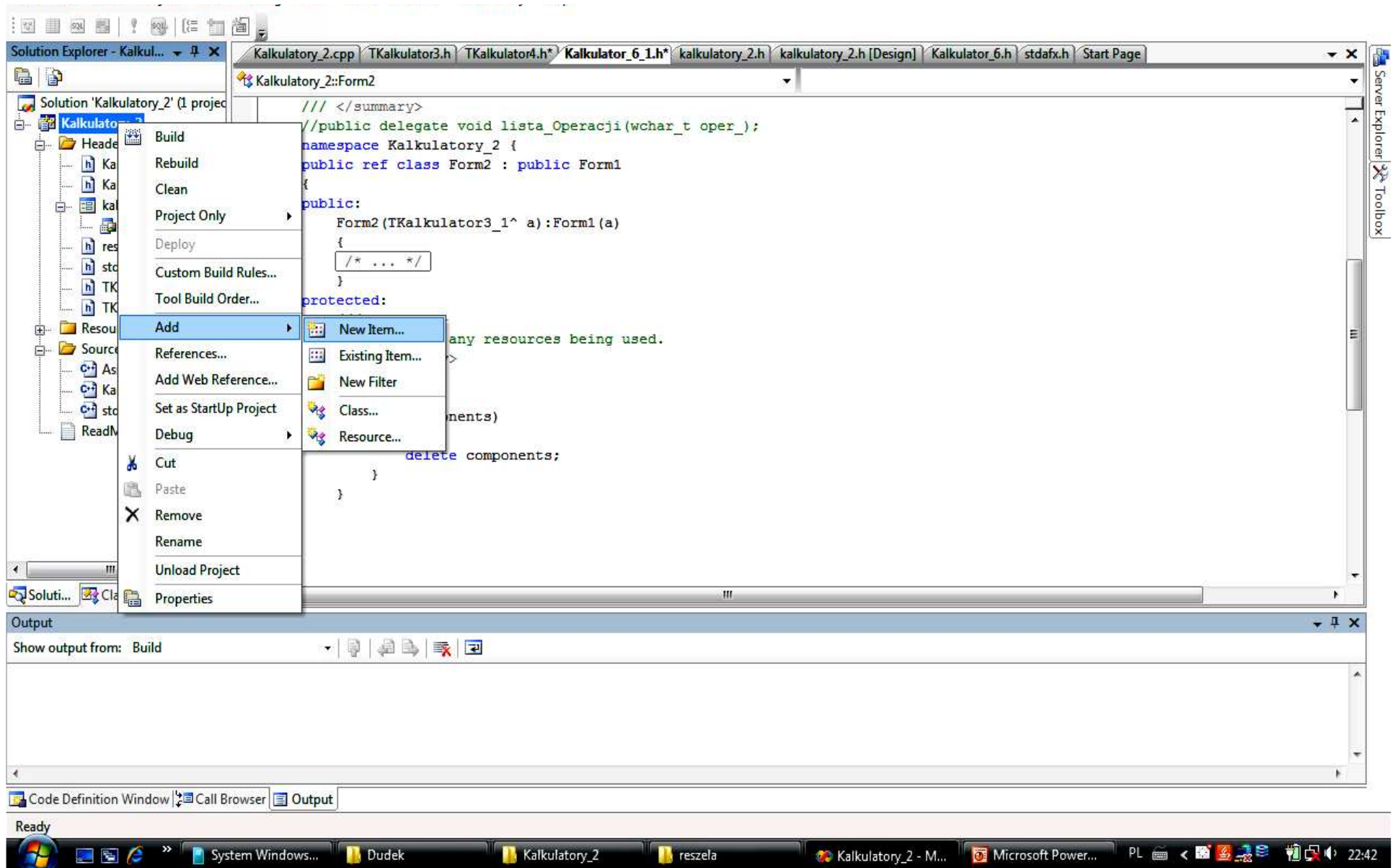
Dziedziczenie – cd., pliki

Autor: Zofia Kruczkiewicz

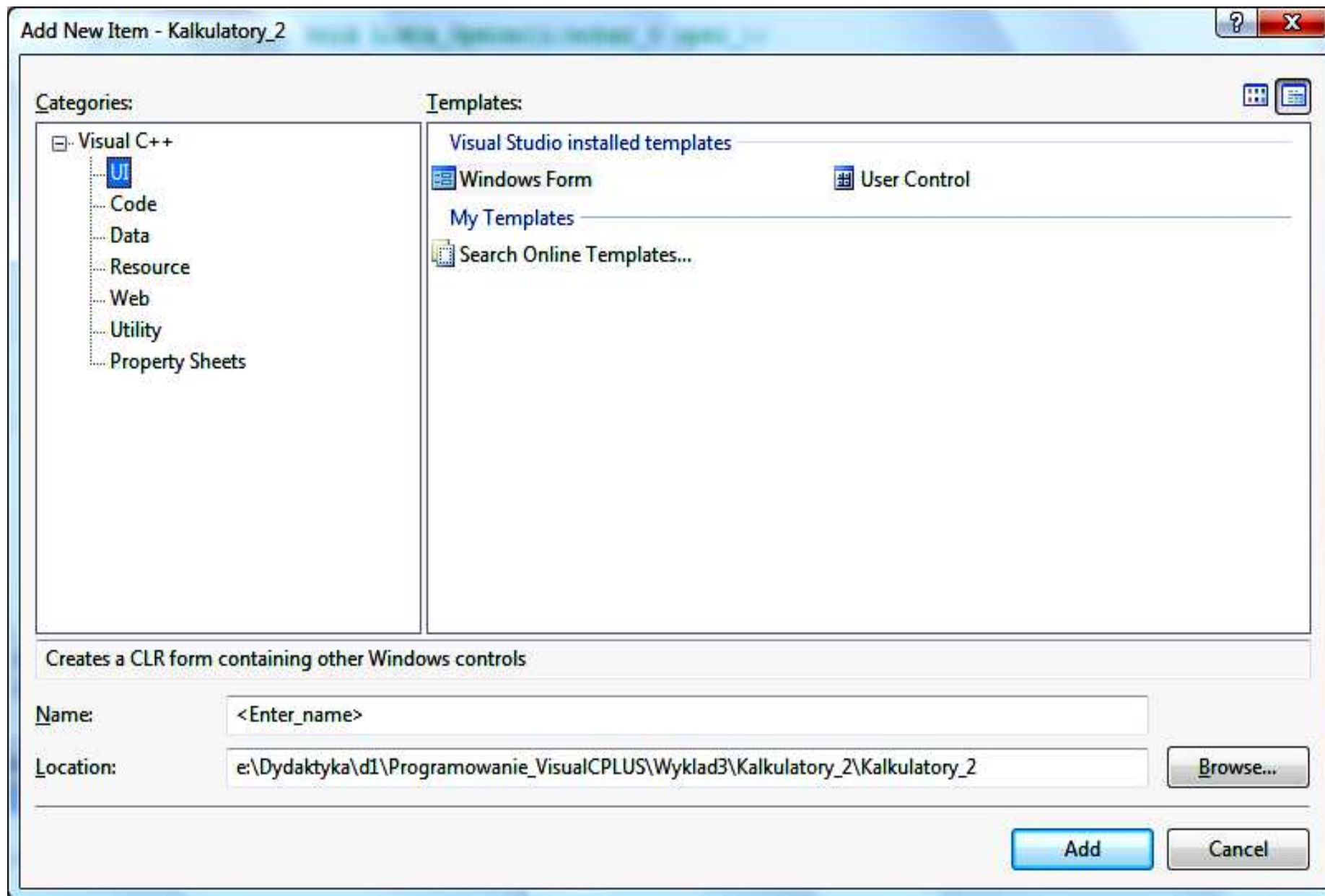
Zagadnienia

- 1. Dziedziczenie cd.**
- 2. Pliki - serializacja**

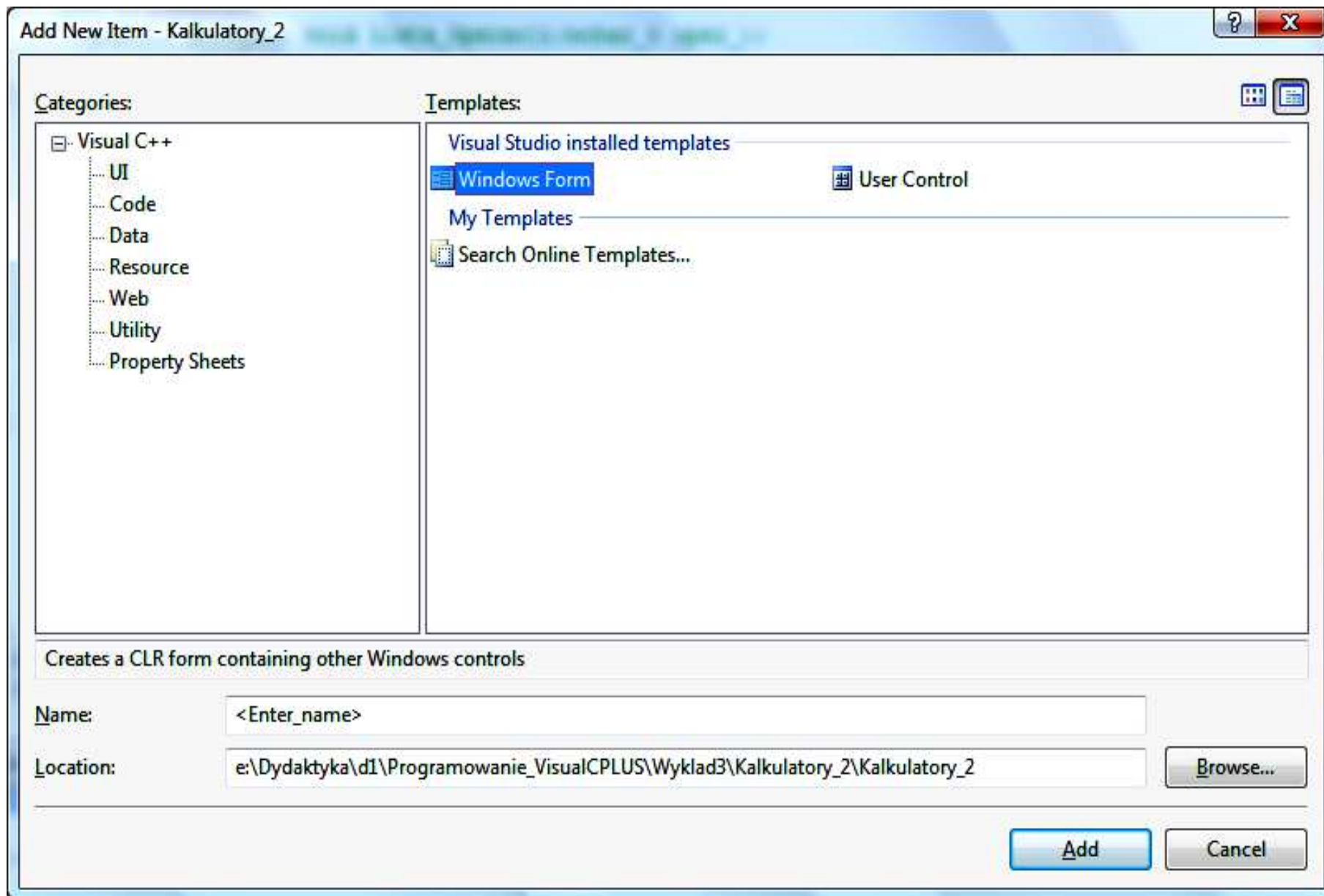
1. Dziedziczenie – aplikacja Kalkulatory_2 typu Windows Forms prezentująca dziedziczenie formularzy obsługujących rodzinę kalkulatorów TKalkulator_3 oraz TKalkulator3_1



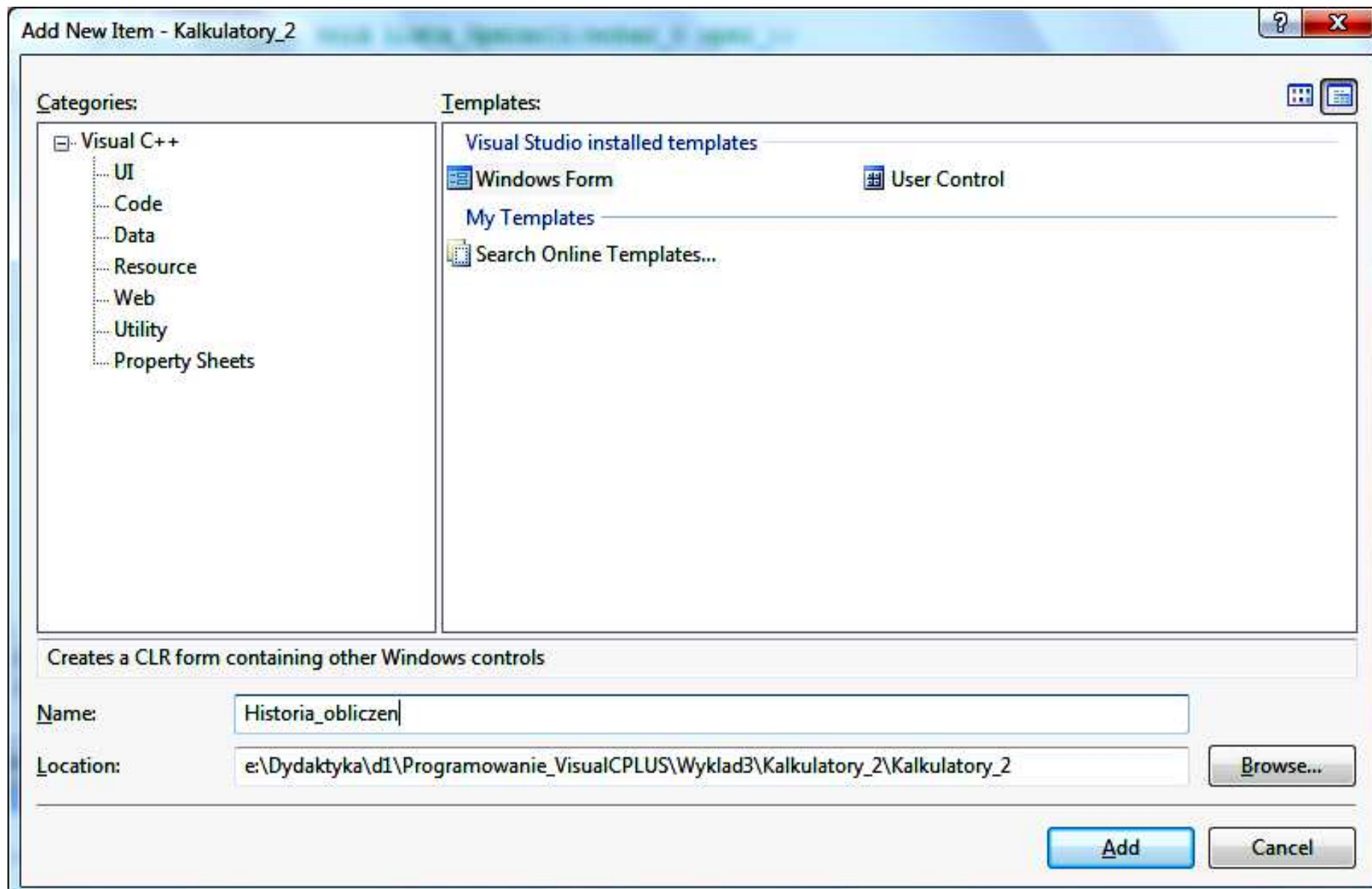
Wstawianie nowego formularza typu Windows Forms (1)



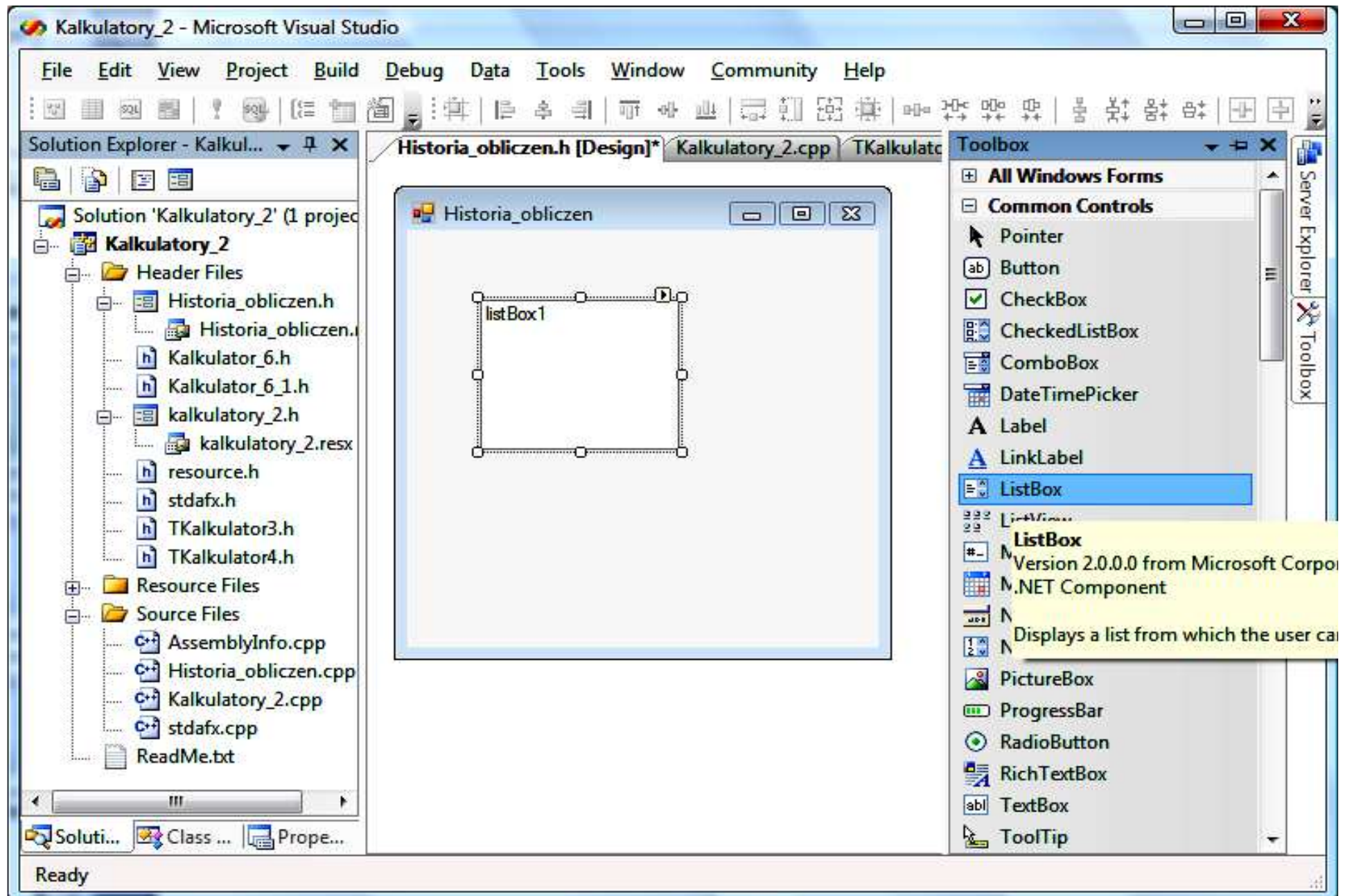
Wstawianie nowego formularza typu Windows Forms (2)



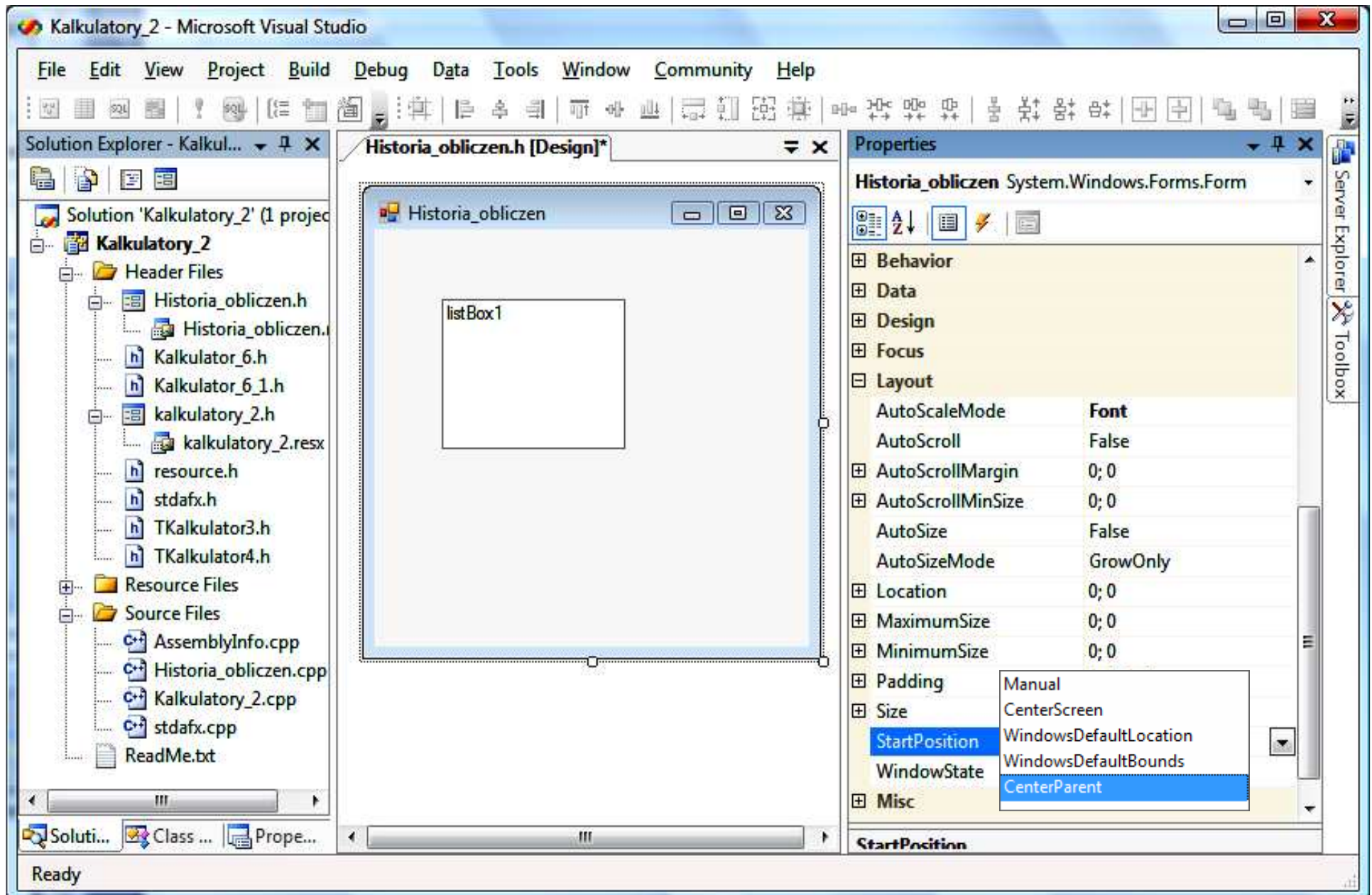
Wstawianie nowego formularza typu Windows Forms (3) w celu przekształcenia go w okno dialogowe *Historia_obliczen* wywoływane z formularza kalkulatora pochodnego typu TKalkulator3_1



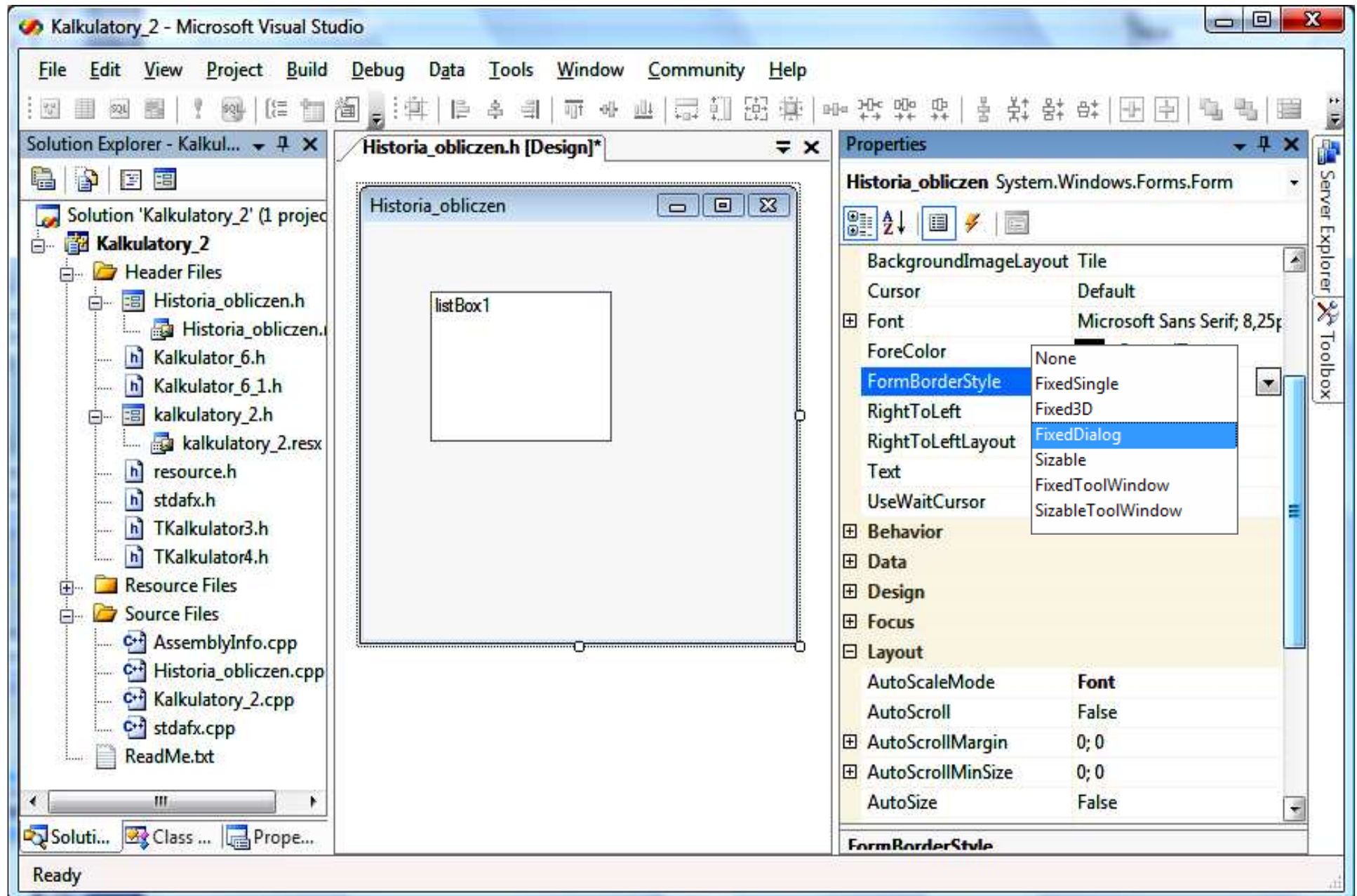
Wstawianie do formularza komponentu typu **ListBox**



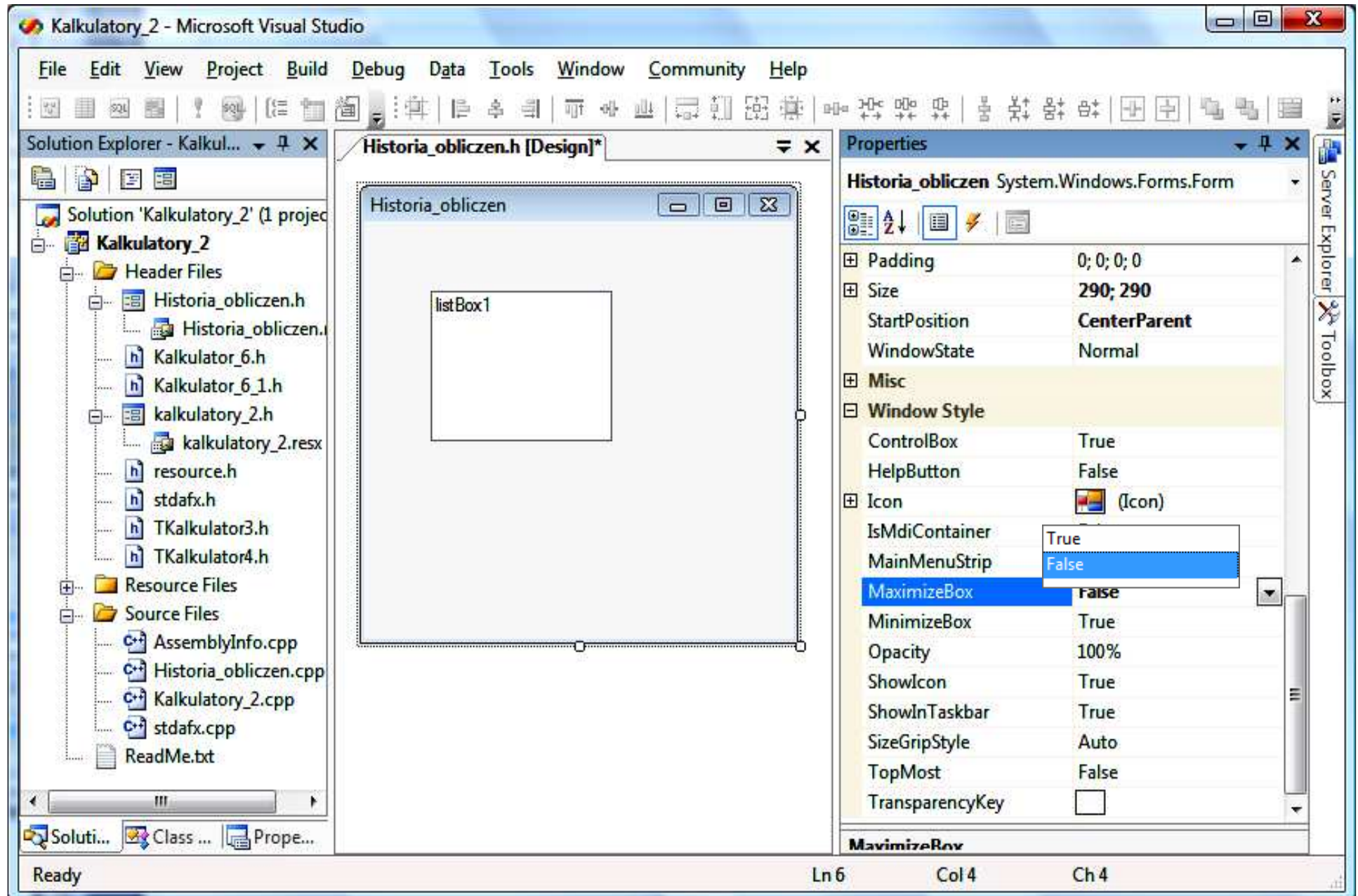
Tworzenie okna dialogowego typu *Historia_obliczen* wywoływanego z formularza kalkulatora pochodnego TKalkulator3_1 (1)



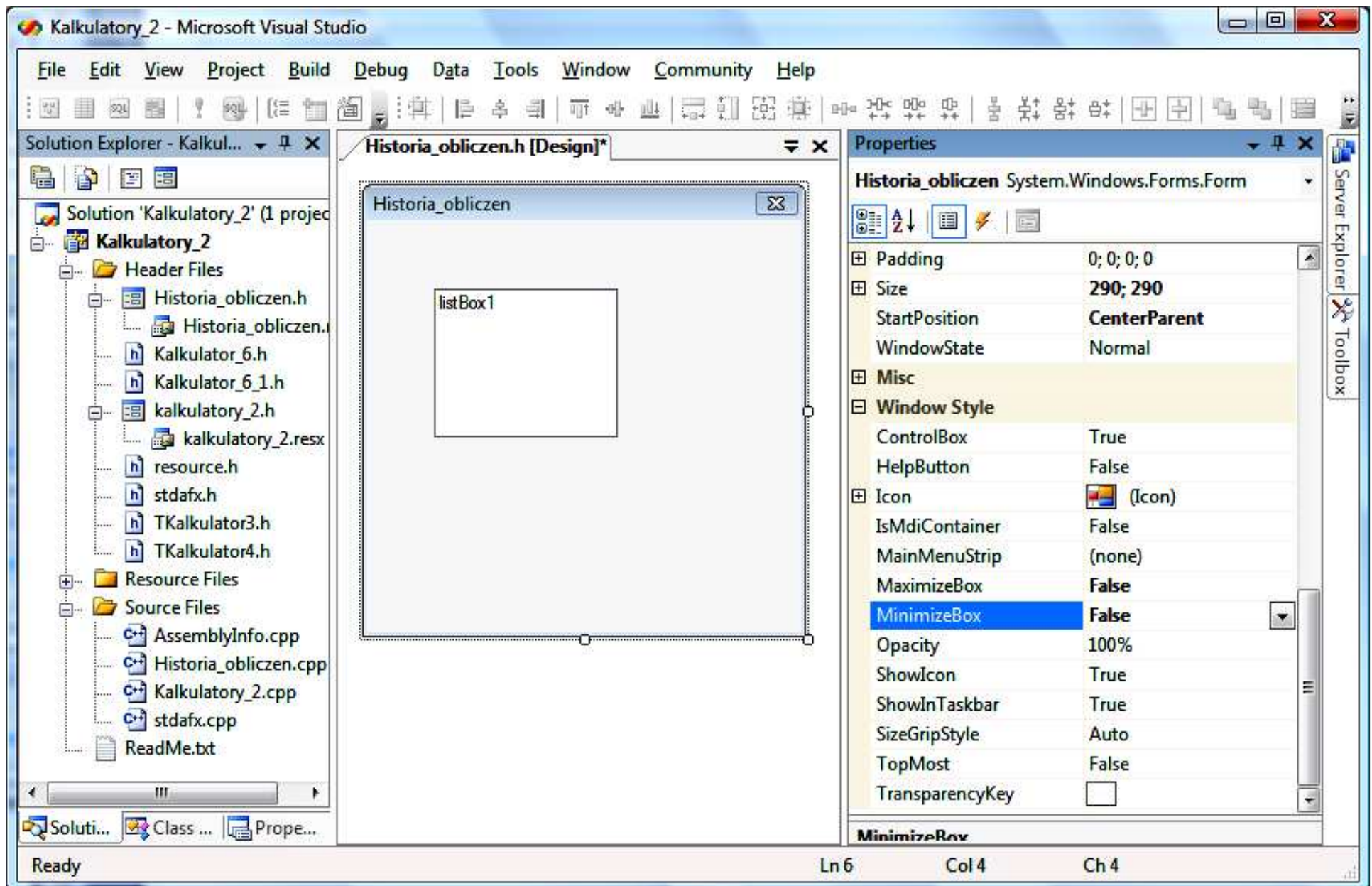
Tworzenie okna dialogowego typu *Historia_obliczen* wywoływanego z formularza kalkulatora pochodnego TKalkulator3_1 (2)



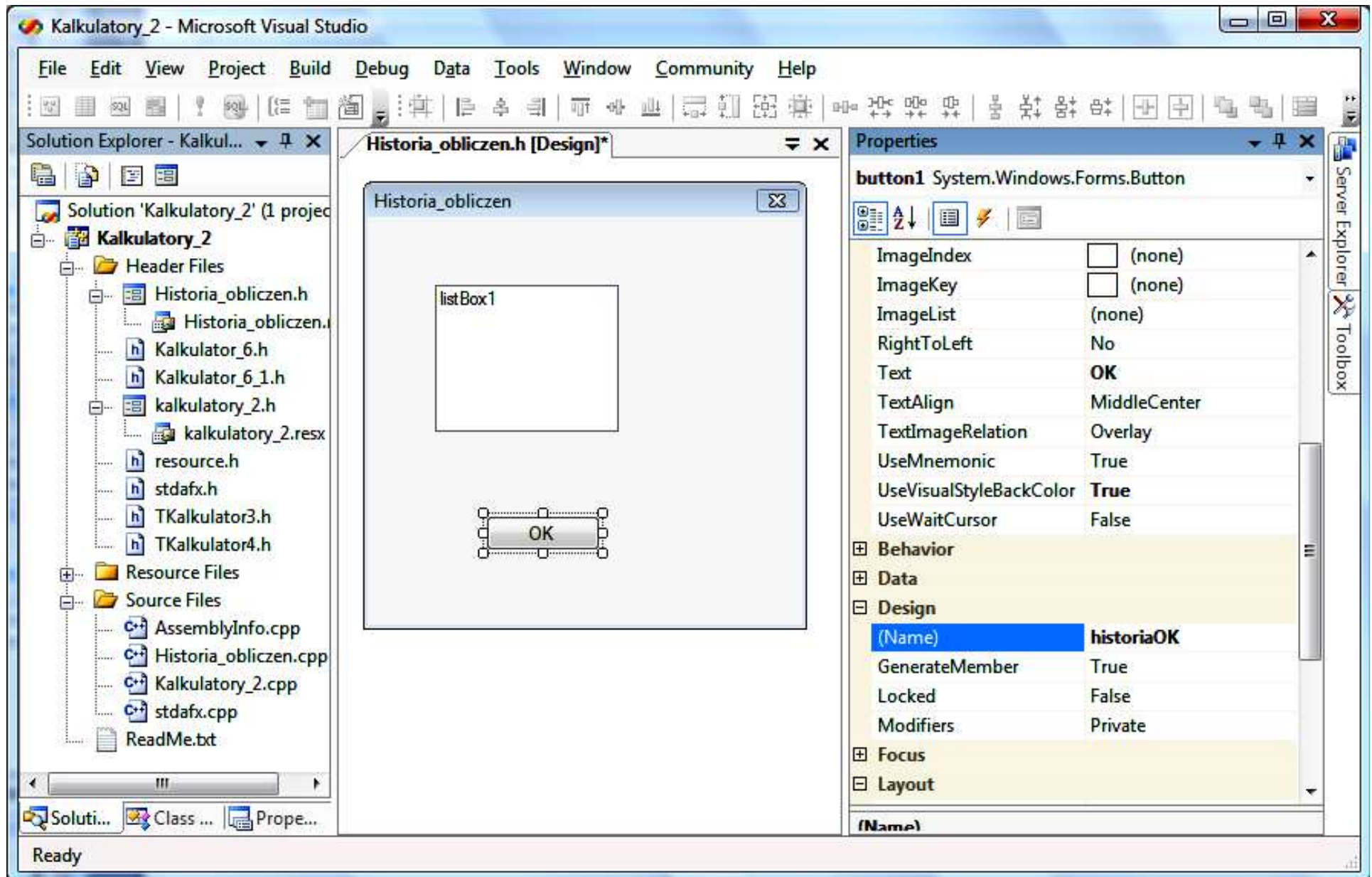
Tworzenie okna dialogowego typu *Historia_obliczen* wywoływanego z formularza kalkulatora pochodnego TKalkulator3_1 (3)



Tworzenie okna dialogowego typu *Historia_obliczen* wywoływanego z formularza kalkulatora pochodnego TKalkulator3_1 (4)



Tworzenie okna dialogowego typu *Historia_obliczen* wywoływanego z formularza kalkulatora pochodnego TKalkulator3_1 (5)



Tworzenie okna dialogowego typu *Historia_obliczen* wywoływanego z formularza kalkulatora pochodnego TKalkulator3_1 (6)

The image shows a screenshot of Microsoft Visual Studio with the following components:

- Solution Explorer:** Shows the project structure for 'Kalkulatory_2', including Header Files (Historia_obliczen.h, Kalkulator_6.h, etc.), Resource Files, and Source Files (AssemblyInfo.cpp, Historia_obliczen.cpp, etc.).
- Design View:** Displays the 'Historia_obliczen' dialog box. It contains a list box labeled 'lista_obliczen' and an 'OK' button.
- Properties Window:** Shows the properties for the selected 'lista_obliczen' control (System.Windows.Forms.ListBox).

Property	Value
AllowDrop	False
ColumnWidth	0
ContextMenuStrip	(none)
DrawMode	Normal
Enabled	True
HorizontalExtent	0
HorizontalScrollbar	False
ImeMode	NoControl
IntegralHeight	True
ItemHeight	13
MultiColumn	False
ScrollAlwaysVisible	False
SelectionMode	One
Sorted	False
TabIndex	0
TabStop	True
UseTabStops	True
Visible	True
Data	
Design	
(Name)	lista_obliczen
GenerateMember	True
Locked	False
Modifiers	Private

Tworzenie okna dialogowego typu *Historia_obliczen* wywoływanego z formularza kalkulatora pochodnego TKalkulator3_1 (7)

The screenshot displays the Microsoft Visual Studio IDE with the following components:

- Solution Explorer:** Shows the project structure for 'Kalkulatory_2'. The 'Source Files' folder contains 'Historia_obliczen.cpp', 'TKalkulator3.h', and 'TKalkulator4.h'.
- Code Editor:** Displays the 'Historia_obliczen.h' file with the following code:

```
lista_obliczen
```
- Form Designer:** Shows a dialog box window titled 'Historia_obliczen' with a text box containing 'lista_obliczen' and an 'OK' button.
- Properties Window:** Shows the properties for the 'Historia_obliczen' form. The 'Misc' section is expanded, and the 'AcceptButton' property is set to 'historiaOK'.

Property	Value
AutoSize	False
AutoSizeMode	GrowOnly
Location	0; 0
MaximumSize	0; 0
MinimumSize	0; 0
Padding	0; 0; 0; 0
Size	290; 290
StartPosition	CenterParent
WindowState	Normal
Misc	(none)
AcceptButton	historiaOK
CancelButton	(none)
KeyPreview	False
Window Style	
ControlBox	True
HelpButton	False
Icon	(Icon)
IsMdiContainer	False
MainMenuStrip	(none)
MaximizeBox	False
MinimizeBox	False
Opacity	100%
ShowIcon	True
ShowInTaskbar	True
SizeGripStyle	Auto

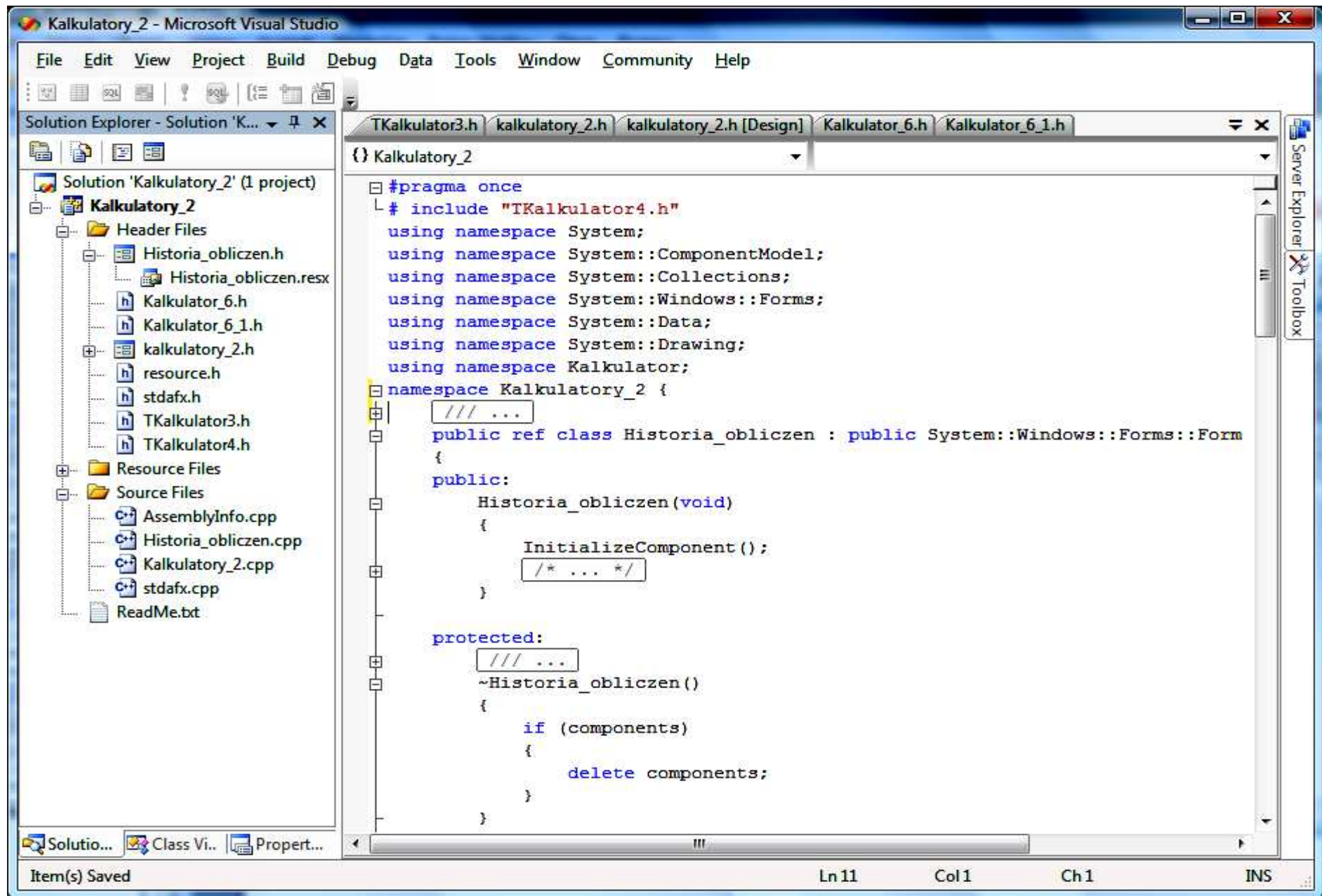
Tworzenie okna dialogowego typu *Historia_obliczen* wywoływanego z formularza kalkulatora pochodnego TKalkulator3_1 (8)

The image shows a screenshot of Microsoft Visual Studio with the following components:

- Solution Explorer:** Shows a project named 'Kalkulatory_2' with a tree structure including Header Files, Resource Files, and Source Files. The file 'Historia_obliczen.h' is selected.
- Design View:** Displays a dialog box titled 'Historia_obliczen'. It contains a list box labeled 'lista_obliczen' and an 'OK' button.
- Properties Window:** Shows the properties for the selected 'historiaOK System.Windows.Forms.Button'. The 'DialogResult' property is expanded, showing options: None, OK, Cancel, Abort, Retry, Ignore, Yes, and No. 'OK' is currently selected.

Property	Value
FlatAppearance	
FlatStyle	Standard
Font	Microsoft Sans Serif; 8,25
ForeColor	ControlText
Image	(none)
ImageAlign	MiddleCenter
ImageIndex	(none)
ImageKey	(none)
ImageList	(none)
RightToLeft	No
Text	OK
TextAlign	MiddleCenter
TextImageRelation	Overlay
UseMnemonic	True
UseVisualStyleBackColor	True
UseWaitCursor	False
Behavior	
AllowDrop	False
AutoEllipsis	False
ContextMenuStrip	(none)
DialogResult	OK
Enabled	
TabIndex	
TabStop	
UseCompatibleTextR	

Programowanie okna dialogowego typu **Historia_obliczen**



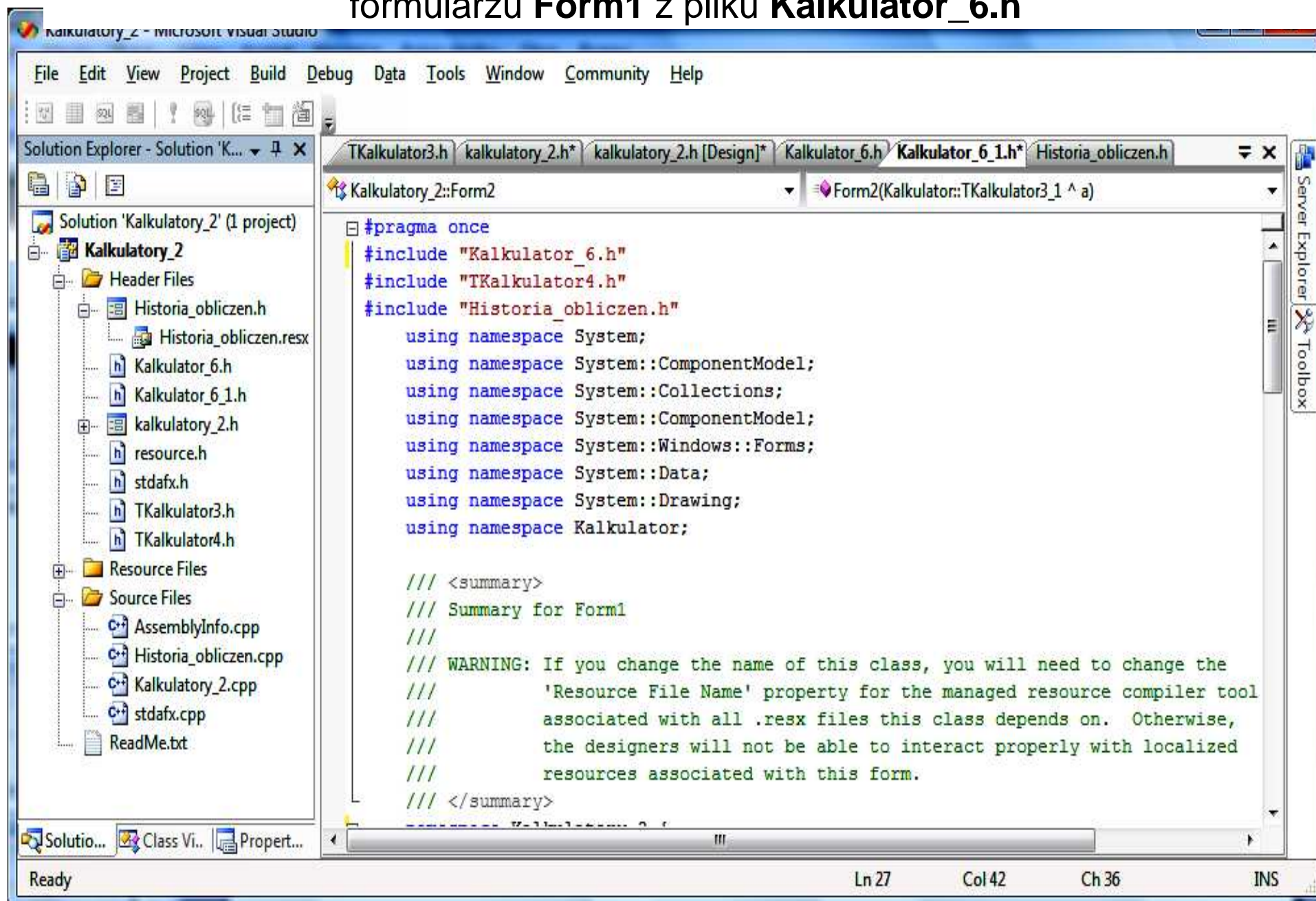
Dodawanie listy do **ListBox** za pomocą **metody archiwum** z klasy **TKalkulator3_1**, która dostarcza historię wykonywanych obliczeń

The screenshot displays the Visual Studio IDE with the following components:

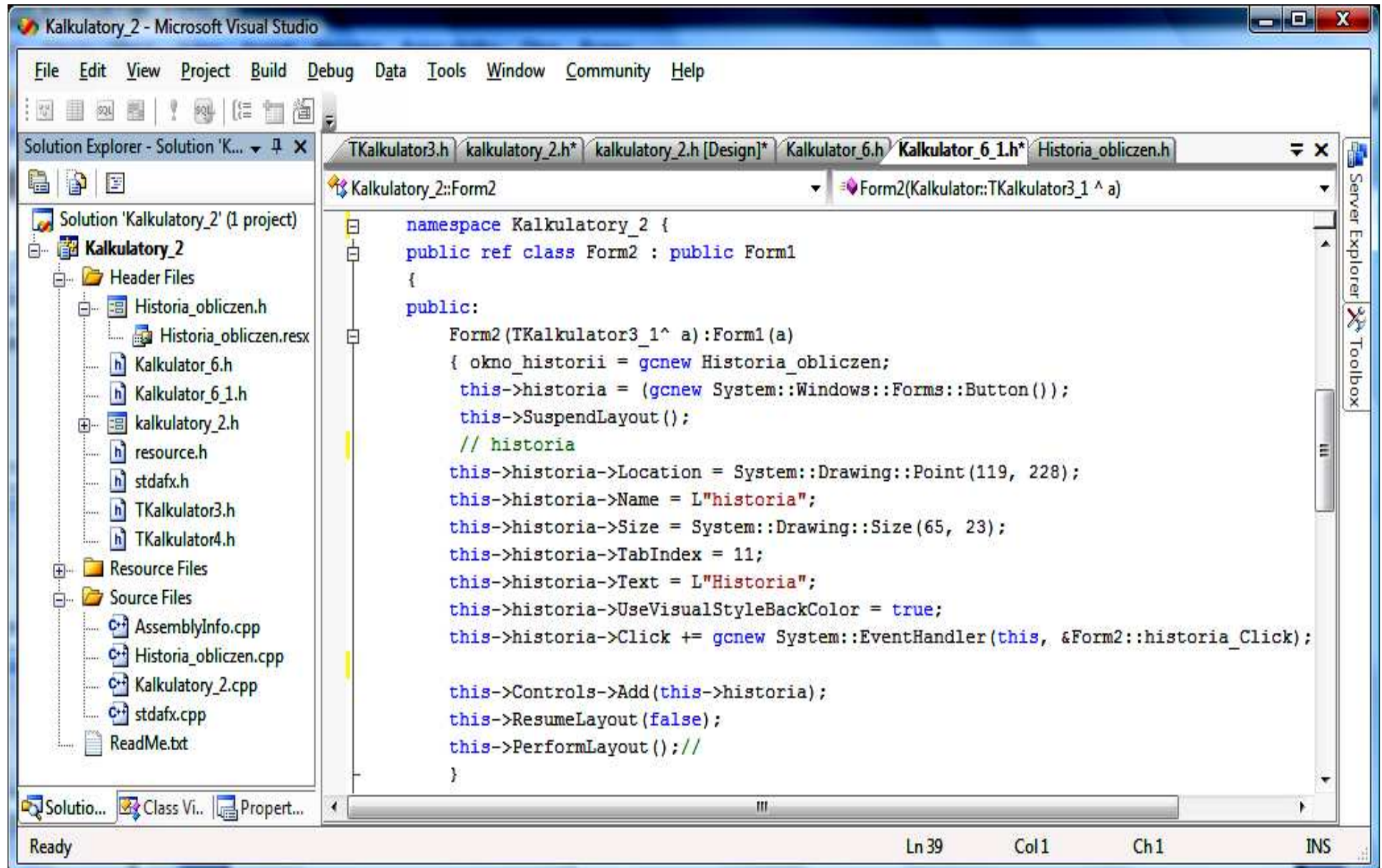
- Solution Explorer:** Shows the project 'Kalkulatory_2' with folders for Header Files, Resource Files, and Source Files. The 'Historia_obliczen.h' file is selected in the Header Files folder.
- Code Editor:** Displays the implementation of the 'Historia_obliczen' class in 'Historia_obliczen.cpp'. The code includes:
 - Private members: `System::Windows::Forms::ListBox^ lista_obliczen;` and `System::Windows::Forms::Button^ historiaOK;`
 - A comment: `/// ...`
 - A private member: `System::ComponentModel::Container ^components;`
 - A public method: `void Lista_wynikow(TKalkulator3_1^ kalkulator)` which:
 - Calls `lista_obliczen->BeginInitUpdate();`
 - Clears the list: `lista_obliczen->Items->Clear();`
 - Iterates over the calculator's history (indices 0 to 4):
 - Constructs a string `String^ pom` by concatenating `kalkulator->archiwum(i)` with `safe_cast<wchar_t>(kalkulator->archiwum(i+1))`, `kalkulator->archiwum(i+2)`, and `kalkulator->archiwum(i+3)`.
 - Adds the string to the list: `lista_obliczen->Items->Add(pom);`
 - Catches exceptions: `catch(Exception^ ex)`
 - Ends the update: `lista_obliczen->EndInitUpdate();`

The status bar at the bottom indicates 'Item(s) Saved', 'Ln 96', 'Col 37', 'Ch 18', and 'INS'.

Formularz **Form2** w pliku **Kalkulator_6_1.h** dziedziczący składowe po formularzu **Form1** z pliku **Kalkulator_6.h**



Uzupełnienie inicjalizacji nowego formularza w konstruktorze (ręczne dodanie w kodzie przycisku **historia** typu **Button** wywołującego okno dialogowe **okno_historii** z historią obliczeń dla drugiego kalkulatora)



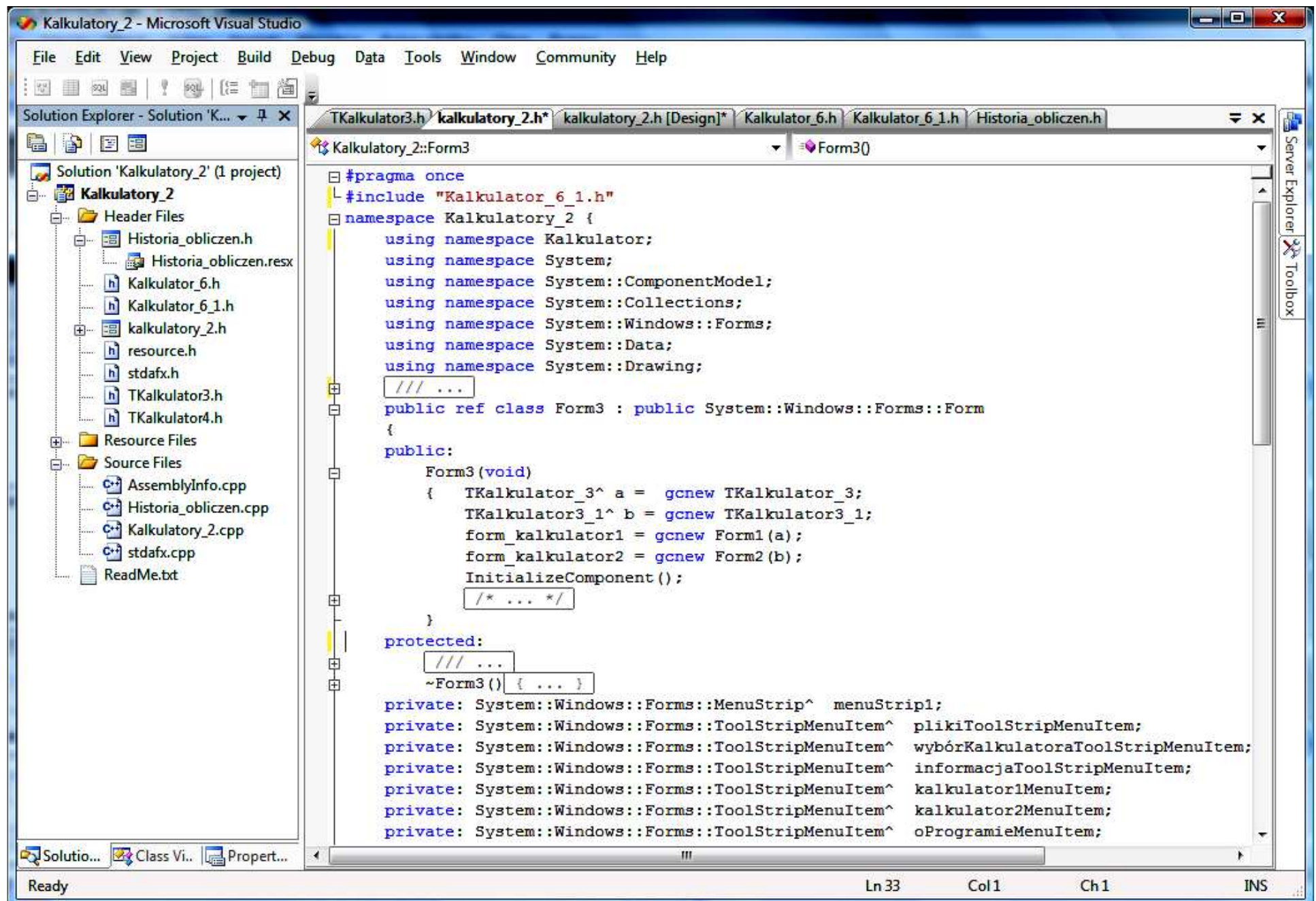
Obsługa zdarzenia klikania na przycisk **historia** w formularzu **Form2**

The screenshot displays the Microsoft Visual Studio IDE for a project named 'Kalkulatory_2'. The Solution Explorer on the left shows the project structure, including header files like 'Historia_obliczen.h' and 'Kalkulator_6.h', resource files, and source files like 'Kalkulatory_2.cpp'. The main editor window shows the implementation of the 'Form2' class in 'Kalkulatory_2.cpp'. The code includes a destructor, a constructor, and a click event handler for the 'historia' button. The event handler calls 'safe_cast' to get the calculator object, updates the results list, and shows a dialog box.

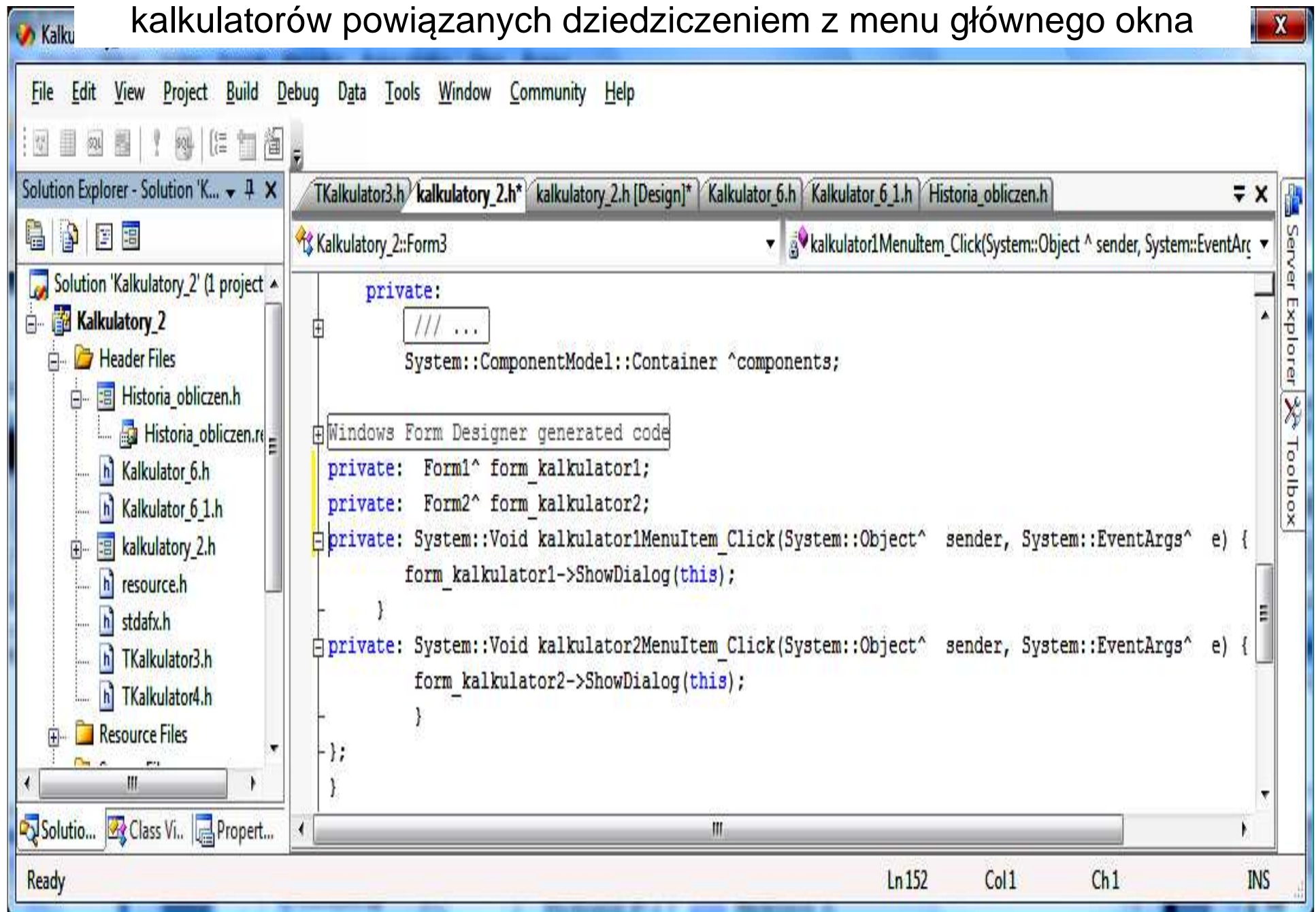
```
protected:
    /// <summary>
    /// Clean up any resources being used.
    /// </summary>
    ~Form2 ()
    {
        if (components)
        {
            delete components;
        }
    }
private: System::Windows::Forms::Button^ historia;
private: Historia_obliczen^ okno_historii;
private: System::Void historia_Click(System::Object^ sender, System::EventArgs^ e) {
    TKalkulator3_1^ kalkulator_ = safe_cast <TKalkulator3_1^>(kalkulator->kalkulator);
    okno_historii->Lista_wynikow(kalkulator_);
    ::DialogResult rezultat = okno_historii->ShowDialog(this);
}
};
}
```

Ready Ln 39 Col 1 Ch 1 INS

Aplikacja Kalkulatory_2 z formularzem typu Form3 z dwoma kalkulatorami



Obsługa zdarzeń listy podmenu – wywołanie formularzy dwóch kalkulatorów powiązanych dziedziczeniem z menu głównego okna



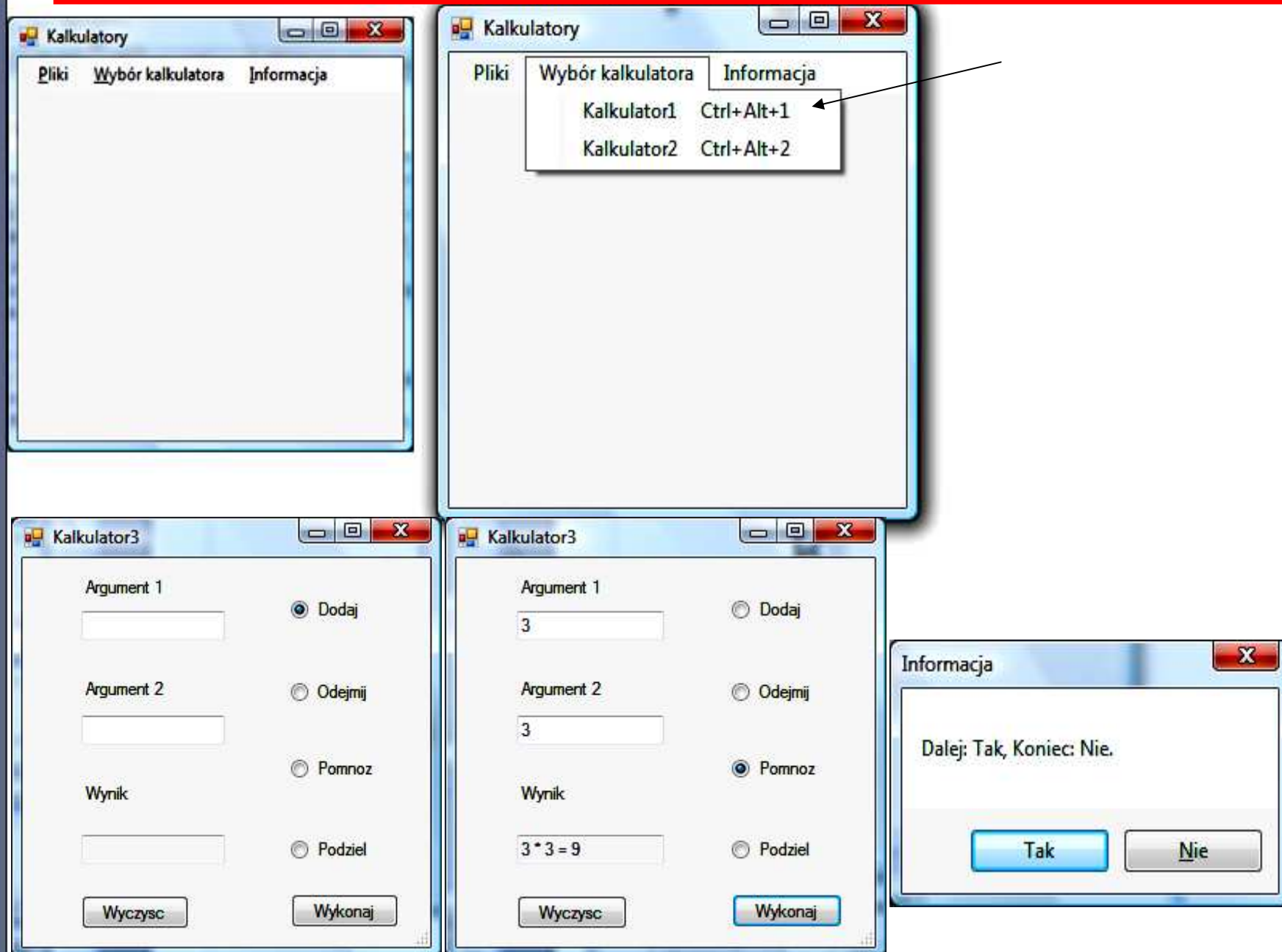
The screenshot displays the Visual Studio IDE with the following components:

- Menu:** File, Edit, View, Project, Build, Debug, Data, Tools, Window, Community, Help.
- Solution Explorer:** Shows a solution named 'Kalkulatory_2' containing a project 'Kalkulatory_2'. The project structure includes:
 - Header Files: Historia_obliczen.h, Historia_obliczen.rc, Kalkulator_6.h, Kalkulator_6_1.h, kalkulatory_2.h, resource.h, stdafx.h, TKalkulator3.h, TKalkulator4.h.
 - Resource Files: (empty folder).
- Code Editor:** Displays the implementation of the `kalkulator1MenuItem_Click` event handler in `Kalkulatory_2::Form3`. The code is as follows:

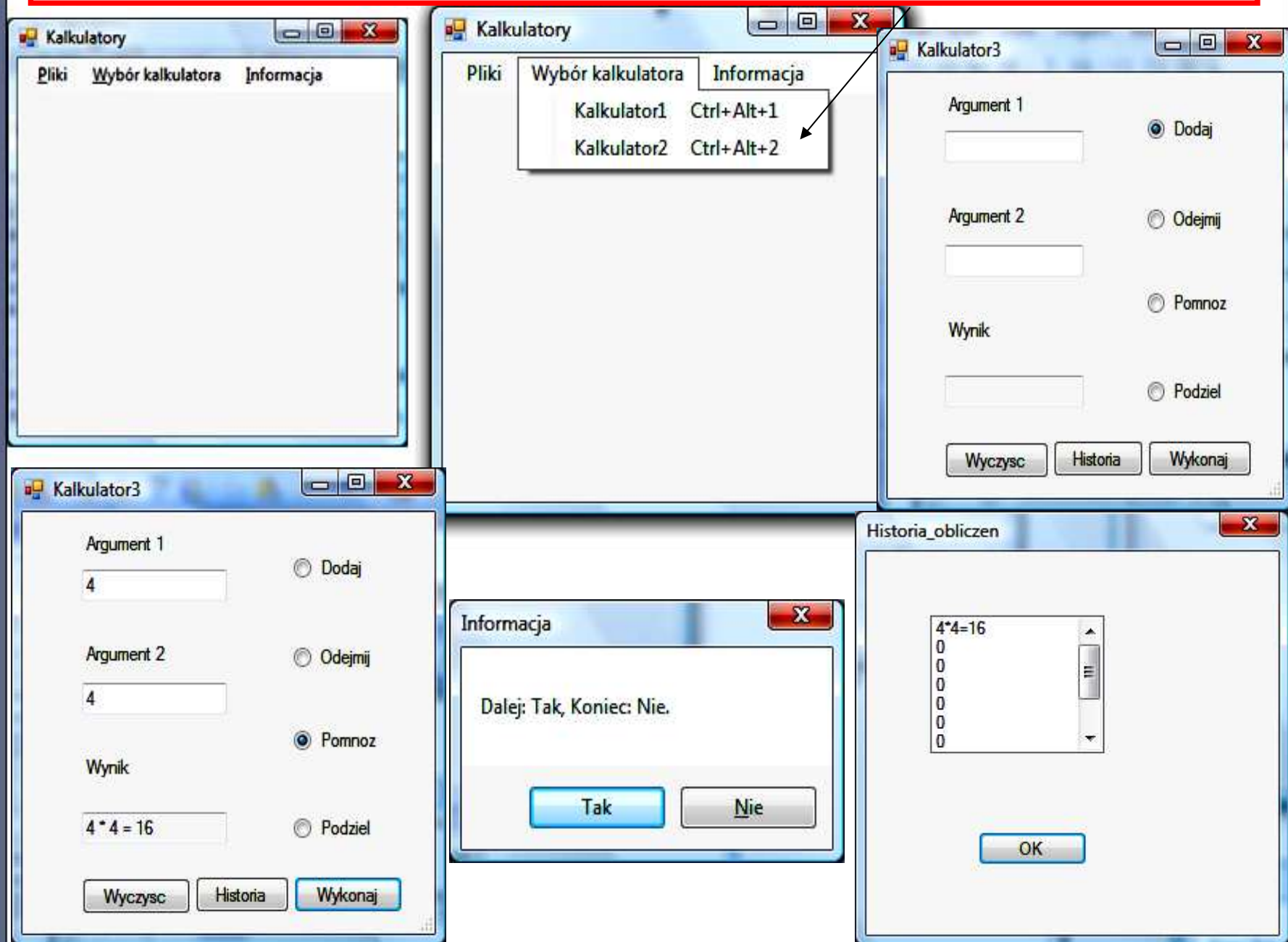
```
private:
    /// ...
    System::ComponentModel::Container ^components;

Windows Form Designer generated code
private: Form1^ form_kalkulator1;
private: Form2^ form_kalkulator2;
private: System::Void kalkulator1MenuItem_Click(System::Object^ sender, System::EventArgs^ e) {
    form_kalkulator1->ShowDialog(this);
}
private: System::Void kalkulator2MenuItem_Click(System::Object^ sender, System::EventArgs^ e) {
    form_kalkulator2->ShowDialog(this);
};
};
```
- Status Bar:** Shows 'Ready', 'Ln152', 'Col1', 'Ch1', and 'INS'.

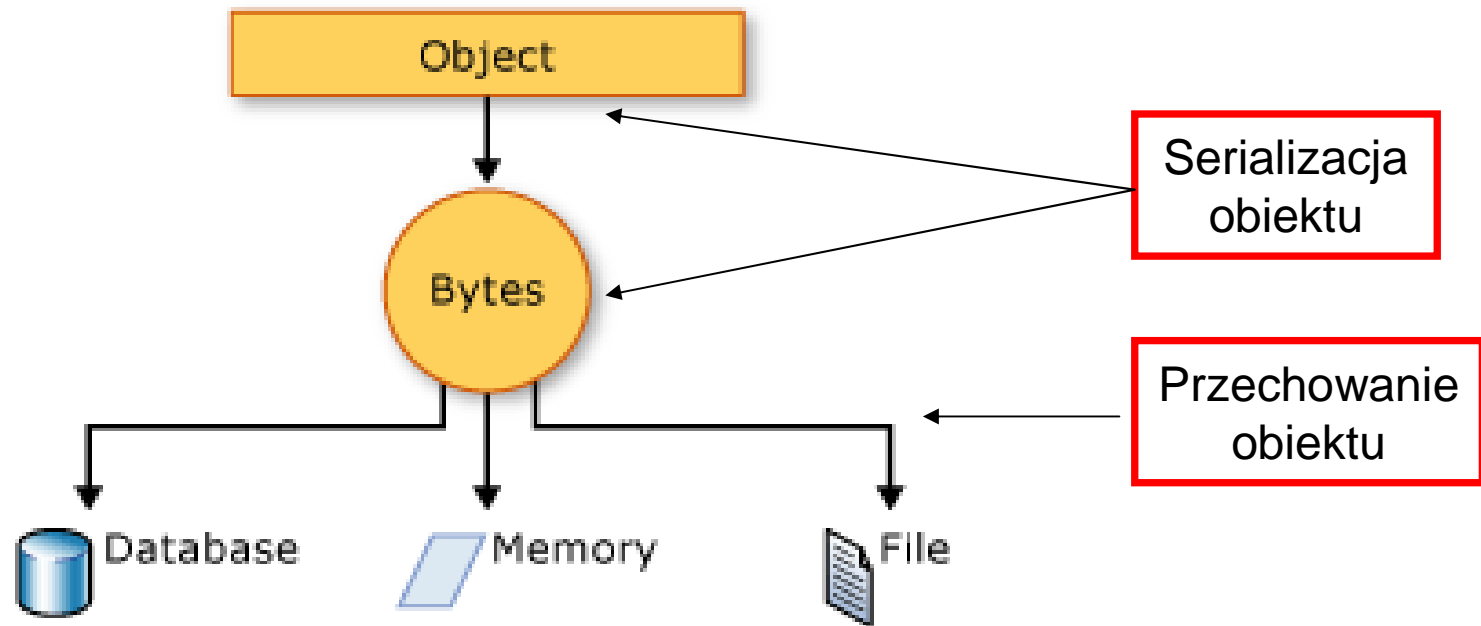
Działanie programu - Wywołanie kalkukatora bez pamięci (Kalkulator1)



Działanie programu - Wywołanie kalkulatora z pamięcią (Kalkulator2)



2. Pliki – serializacja (1)



- **Serializacja** jest procesem zamiany obiektu na strumień bajtów w celu utrwalenia w pamięci, bazie danych lub pliku. Podstawowym zadaniem jest zachowanie stanu obiektów w celu ich odtworzenia.
- **Serializowane są:** nazwa klasy, asemblacje, dane składowe obiektu
- Proces odwrotny nazywa się **deserializacją**.

Serializacja (2)

Zastosowanie serializacji

- Przechowywanie obiektów wraz ze stanem (programów)
- Przesyłanie obiektów w postaci strumieni bajtów przez sieć między aplikacjami typu Web Service lub przesyłanie obiektów jako łańcuchów XML przez firewall, obsługa bezpieczeństwa danych lub specyficznych danych użytkownika.

Serializacja obiektów

- Składowe: obiekty przeznaczone do serializacji i deserializacji, strumień zawierający strumień bajtów oraz format and a Formatter. Zapewniają to klasy pakietu **System.Runtime.Serialization**
- Zastosowanie atrybutów typu **SerializableAttribute** w celu identyfikacji serializowanych obiektów. Wyjątek typu **SerializationException** jest generowany w momencie braku tych atrybutów.
- Można wykluczyć wybrane atrybuty obiektów z serializacji za pomocą atrybutów typu **NonSerializedAttribute**. Jeśli pole serializowane zawiera wskaźnik, uchwyt lub inne specyficzne dane trudne do odtworzenia w innym środowisku, dlatego powinny być wyłączone z serializacji przez programistę.
- Jeśli serializowana klasa zawiera referencje do obiektów innych klas, wtedy w celu serializowania tych obiektów składowych należy klasy tych obiektów oznaczyć za pomocą atrybutów typu **SerializableAttribute**.
- Jeśli referencje do tego samego obiektu znajdują się w innych obiektach, podczas serializacji tych obiektów tylko raz jest ten „wspólny” obiekt zamieniany na strumień bajtów i zapisywany w strumieniu.

Serializacja (3)

Serializacja binarna

- Binarna serializacja używa binarnego formatu kodowania strumienia bajtów w pamięci lub sieciowych strumieniach opartych na gniazdach. Jest to wydajny sposób serializacji, jednak niezalecany do przesyłania przez firewall.

Serializacja XML

- **Serializacja XML** serializuje publiczne pola i tzw „properties” obiektu lub parametry i wartości zwracane przez metody i przekształca je w strumień łańcuchów XML wg wyznaczonego schematu XML jako dokument XML (*XML Schema definition language (XSD) document*). Pakiet **System.Xml.Serialization** zawiera klasy potrzebne do serializacji i deserializacji XML.
- Należy posługiwać się atrybutami serializacji w celu zarządzania serializacją obiektów za pomocą **XmlSerializer**.

Serializacja SOAP

- Serializacja XML może być zastosowana do serializacji wg specyfikacji protokołu SOAP. SOAP jest protokołem opartym na XML, zastosowanym do transportu wywołań procedur, przydatnych do zarządzania wiadomościami generowanymi przez **XML Web Service**.

Przykład 1 – na podstawie MSDN (strumienie typu SOAP)

```
#using <system.dll>
#using <system.messaging.dll>
#using <System.Runtime.Serialization.Formatters.Soap.dll>
using namespace System;
using namespace System::IO;
using namespace System::Runtime::Serialization::Formatters::Soap;
    [Serializable] // A test object that needs to be serialized.
ref class TestSimpleObject
{
    private: int member1;
        String^ member2;
        String^ member3;
        double member4;

public:
    [NonSerialized] // A field that is not serialized.
    String^ member5;
    TestSimpleObject()
    {
        member1 = 11;
        member2 = "hello";
        member3 = "hello";
        member4 = 3.14159265;
        member5 = "hello world!"; }

    void Print()
    {
        Console::WriteLine( "member1 = ' {0}' ", member1 );
        Console::WriteLine( "member2 = ' {0}' ", member2 );
        Console::WriteLine( "member3 = ' {0}' ", member3 );
        Console::WriteLine( "member4 = ' {0}' ", member4 );
        Console::WriteLine( "member5 = ' {0}' ", member5 ); } };
```

```
int main()
{
    //Creates a new TestSimpleObject object.
    TestSimpleObject^ obj = gcnew TestSimpleObject;
    Console::WriteLine( "Before serialization the Object* contains: " );
    obj->Print();
    //Opens a file and serializes the object into it in binary format.
    Stream^ stream = File::Open( "data.xml", FileMode::Create );
    SoapFormatter^ formatter = gcnew SoapFormatter;
    formatter->Serialize( stream, obj );
    stream->Close();
    //Empties obj.
    obj = nullptr;
    //Opens file S"data.xml" and deserializes the object from it.
    stream = File::Open( "data.xml", FileMode::Open );
    formatter = gcnew SoapFormatter;
    obj = dynamic_cast<TestSimpleObject^>(formatter->Deserialize( stream ));
    stream->Close();
    Console::WriteLine( "" );
    Console::WriteLine( "After deserialization the object contains: " );
    obj->Print();
}
```

data - WordPad

Plik Edycja Widok Wstaw Format Pomoc

Ⓜ Ⓜ Ⓜ Ⓜ Ⓜ Ⓜ Ⓜ Ⓜ Ⓜ Ⓜ Ⓜ Ⓜ

```
<SOAP-ENV:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:clr="http://schemas.microsoft.com/soap/encoding clr/1.0" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body>
<a1:TestSimpleObject id="ref-1" xmlns:a1
="http://schemas.microsoft.com/clr/assem/SerialSOAP%2C%20Version%
3D1.0.3169.41348%2C%20Culture%3Dneutral%2C%20PublicKeyToken%3Dnull">
<member1>11</member1>
<member2 id="ref-3">hello</member2>
<member3 href="#ref-3"/>
<member4>3.14159265</member4>
</a1:TestSimpleObject>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Aby uzyskać Pomoc, naciśnij klawisz F1.

C:\Windows\system32\cmd.exe

```
Before serialization the Object* contains:
member1 = ' 11'
member2 = ' hello'
member3 = ' hello'
member4 = ' 3,14159265'
member5 = ' hello world!'

After deserialization the object contains:
member1 = ' 11'
member2 = ' hello'
member3 = ' hello'
member4 = ' 3,14159265'
member5 = ' '

Aby kontynuować, naciśnij dowolny klawisz . . .
```

Przykład 2 – na podstawie MSDN (strumienie binarne)

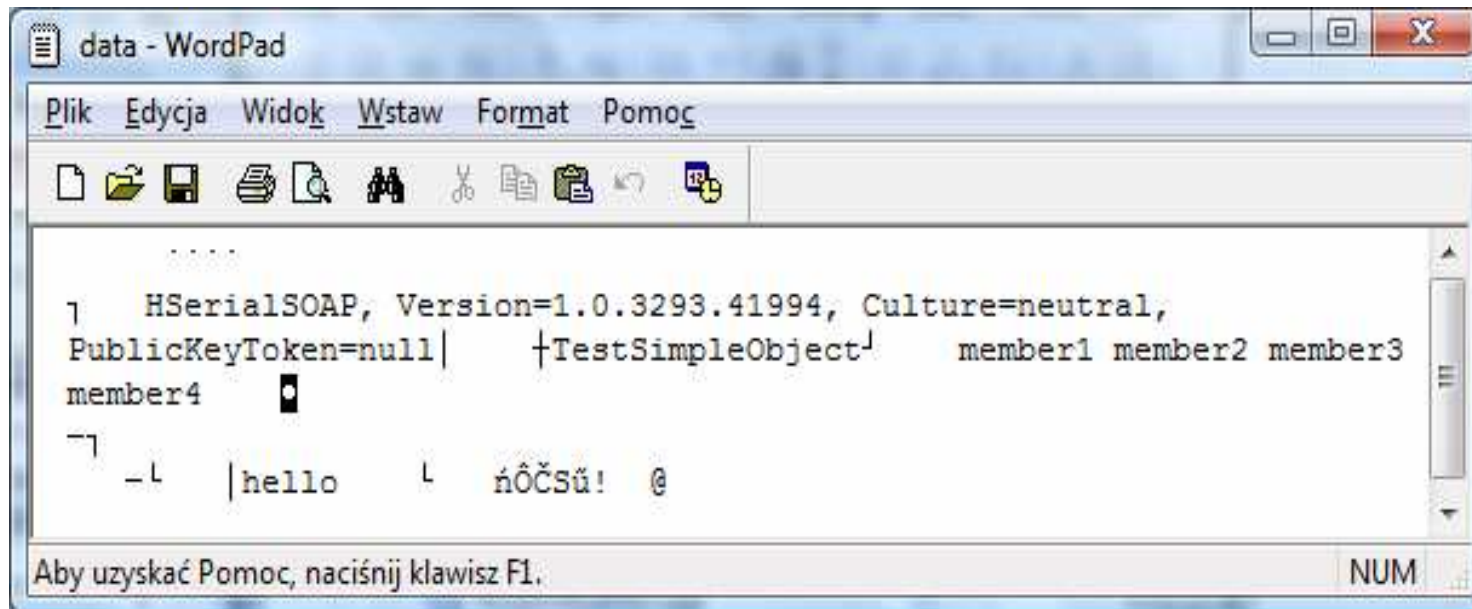
```
#using <system.dll>
#using <system.messaging.dll>
using namespace System;
using namespace System::IO;
using namespace System::Runtime::Serialization::Formatters::Binary;
    [Serializable] // A test object that needs to be serialized.
ref class TestSimpleObject
{
    private: int member1;
        String^ member2;
        String^ member3;
        double member4;

public:
    [NonSerialized] // A field that is not serialized.
    String^ member5;
    TestSimpleObject()
    {
        member1 = 11;
        member2 = "hello";
        member3 = "hello";
        member4 = 3.14159265;
        member5 = "hello world!"; }

    void Print()
    {
        Console::WriteLine( "member1 = ' {0}' ", member1 );
        Console::WriteLine( "member2 = ' {0}' ", member2 );
        Console::WriteLine( "member3 = ' {0}' ", member3 );
        Console::WriteLine( "member4 = ' {0}' ", member4 );
        Console::WriteLine( "member5 = ' {0}' ", member5 ); } };
```

```
int main()
{
    //Creates a new TestSimpleObject object.
    TestSimpleObject^ obj = gcnew TestSimpleObject;
    Console::WriteLine( "Before serialization the Object* contains: " );
    obj->Print();
    //Opens a file and serializes the object into it in binary format.
    Stream^ stream = File::Open( "data.xml", FileMode::Create );
    BinaryFormatter^ formatter = gcnew BinaryFormatter();
    formatter->Serialize( stream, obj );
    stream->Close();
    //Empties obj.
    obj = nullptr;
    //Opens file S"data.xml" and deserializes the object from it.
    stream = File::Open( "data.xml", FileMode::Open );
    formatter = gcnew BinaryFormatter();
    obj = dynamic_cast<TestSimpleObject^>(formatter->Deserialize( stream ));
    stream->Close();
    Console::WriteLine( "" );
    Console::WriteLine( "After deserialization the object contains: " );
    obj->Print();
}
```

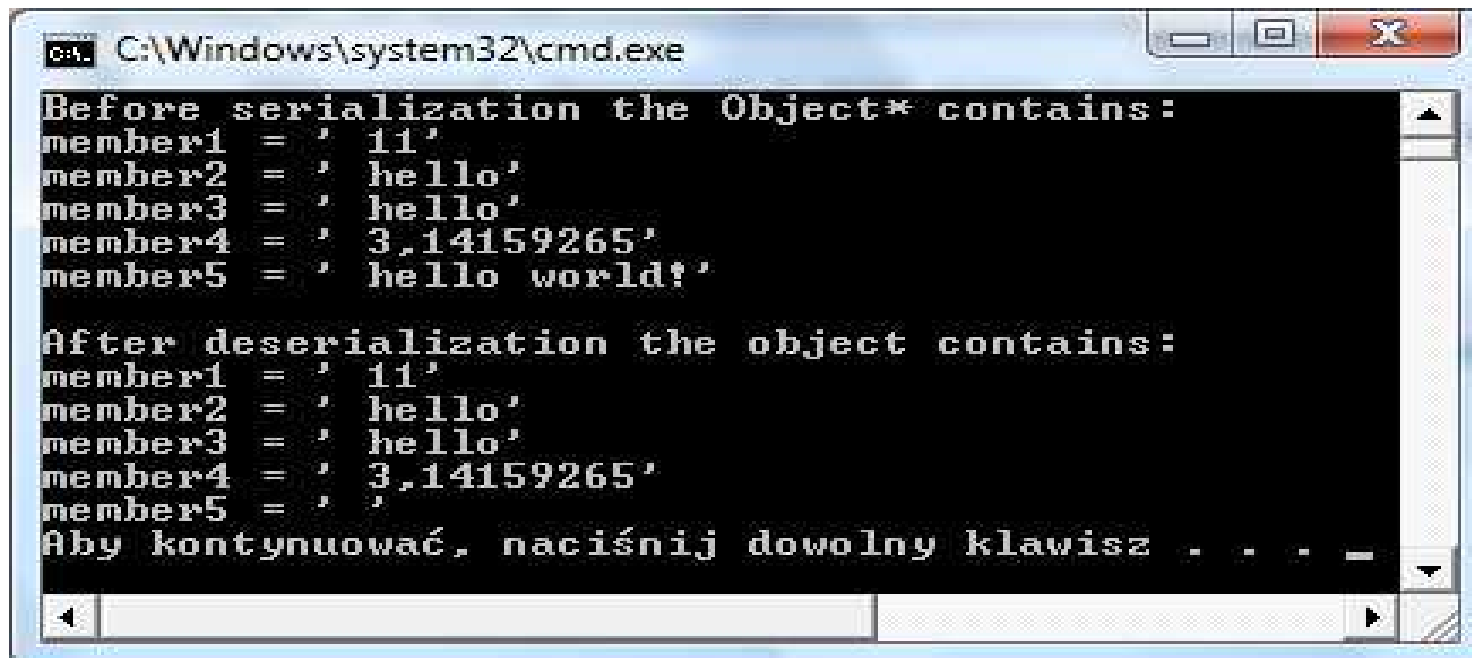

Zastosowanie serializacji binarnej



A screenshot of a WordPad window titled "data - WordPad". The window contains a binary dump of a serialized object. The text is as follows:

```
.....  
[ HSerialSOAP, Version=1.0.3293.41994, Culture=neutral,  
PublicKeyToken=null| +TestSimpleObject| member1 member2 member3  
member4  
-L |hello L nÔČSŭ! @
```

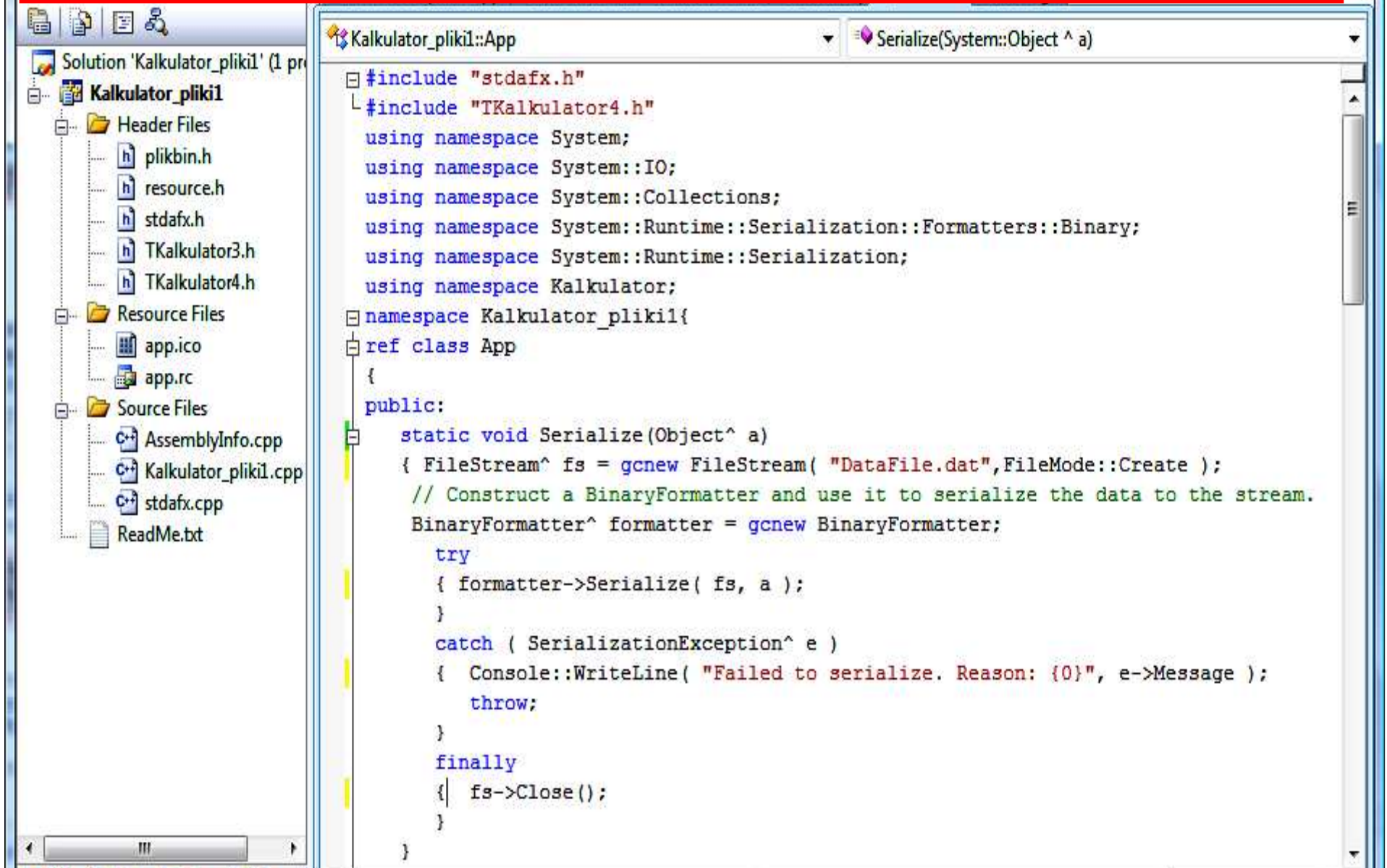
At the bottom of the window, there is a status bar with the text "Aby uzyskać Pomoc, naciśnij klawisz F1." and a "NUM" button.



A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window displays the output of a program that demonstrates binary serialization and deserialization. The text is as follows:

```
Before serialization the Object* contains:  
member1 = ' 11'  
member2 = ' hello'  
member3 = ' hello'  
member4 = ' 3,14159265'  
member5 = ' hello world!'  
  
After deserialization the object contains:  
member1 = ' 11'  
member2 = ' hello'  
member3 = ' hello'  
member4 = ' 3,14159265'  
member5 = ' '  
Aby kontynuować, naciśnij dowolny klawisz . . . -
```

Przykład 3 – zapis kalkulatorów w plikach przez zastosowaniu strumieni binarnych za pomocą klasy App (metoda *Serialize* serializująca obiekty dowolnego typu z adnotacją [Serializable])



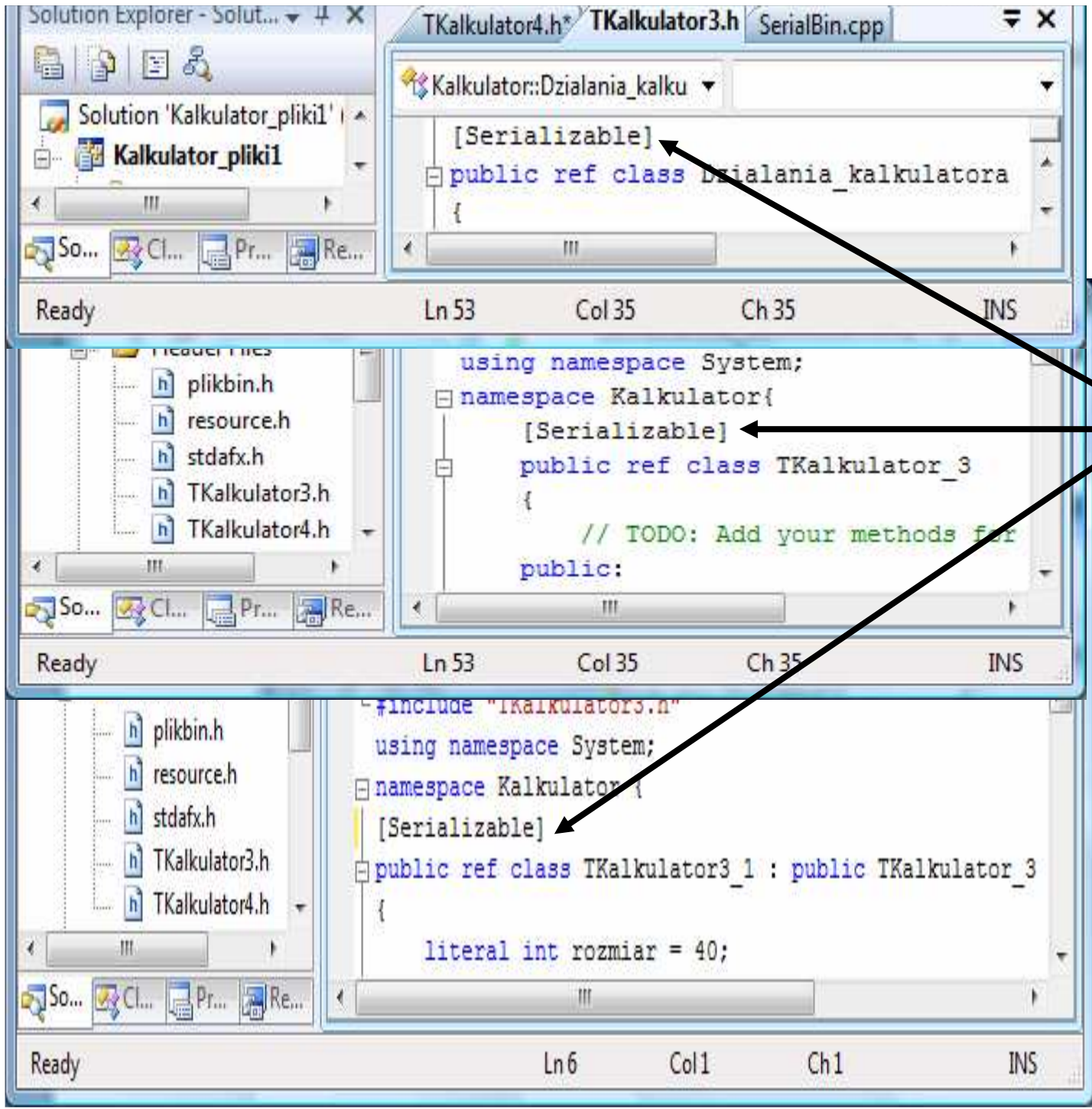
```
Kalkulator_plik1::App Serialize(System::Object ^ a)
#include "stdafx.h"
#include "TKalkulator4.h"
using namespace System;
using namespace System::IO;
using namespace System::Collections;
using namespace System::Runtime::Serialization::Formatters::Binary;
using namespace System::Runtime::Serialization;
using namespace Kalkulator;
namespace Kalkulator_plik1{
ref class App
{
public:
static void Serialize(Object^ a)
{ FileStream^ fs = gcnew FileStream( "DataFile.dat", FileMode::Create );
// Construct a BinaryFormatter and use it to serialize the data to the stream.
BinaryFormatter^ formatter = gcnew BinaryFormatter;
try
{ formatter->Serialize( fs, a );
}
catch ( SerializationException^ e )
{ Console::WriteLine( "Failed to serialize. Reason: {0}", e->Message );
throw;
}
finally
{ fs->Close();
}
}
}
```

Metoda *Deserialize1* deserializująca obiekty typu *TKalkulator_3* oraz pochodne typu *TKalkulator3_1*

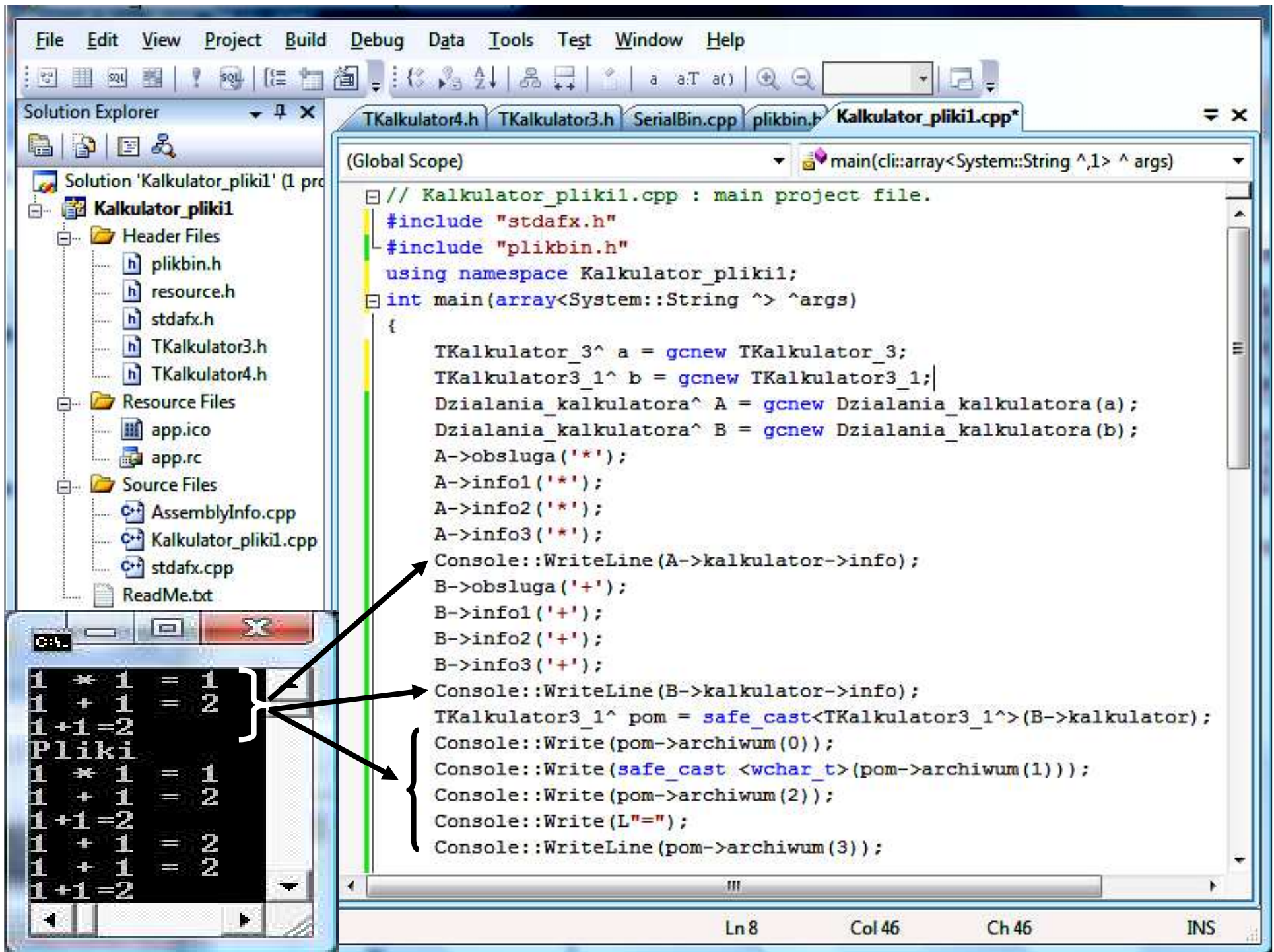
```
Kalkulator_plik1.cpp  TKalkulator4.h  TKalkulator3.h  SerialBin.cpp  plikbin.h*  Start Page
Kalkulator_plik1::App  Deserialize1()
static TKalkulator_3^ Deserialize1()
{
    TKalkulator_3^ addresses = nullptr;
    // Open the file containing the data that we want to deserialize.
    FileStream^ fs = gcnew FileStream( "DataFile.dat", FileMode::Open );
    try
    { BinaryFormatter^ formatter = gcnew BinaryFormatter;
      // Deserialize the object; assign the reference to our local variable.
      addresses = dynamic_cast<TKalkulator_3^>(formatter->Deserialize( fs ));
    }
    catch ( SerializationException^ e )
    { Console::WriteLine( "Failed to deserialize. Reason: {0}", e->Message );
      throw;
    }
    finally
    { | fs->Close();
    }
    return addresses;
}
```

Metoda *Deserialize2* deserializująca obiekty typu *Dzialania_kalkulatora*

```
static Dzialania_kalkulatora^ Deserialize2()
{
    Dzialania_kalkulatora^ addresses = nullptr;
    // Open the file containing the data that we want to deserialize.
    FileStream^ fs = gcnew FileStream( "DataFile.dat", FileMode::Open );
    try
    { BinaryFormatter^ formatter = gcnew BinaryFormatter;
      // Deserialize the object; assign the reference to our local variable.
      addresses = dynamic_cast<Dzialania_kalkulatora^>(formatter->Deserialize( fs ));
    }
    catch ( SerializationException^ e )
    { Console::WriteLine( "Failed to deserialize. Reason: {0}", e->Message );
      throw;
    }
    finally
    { fs->Close();
    }
    return addresses;
};
```



Adnotacje dołączone do definicji klas, niezbędne do serializacji i deserializacji obiektów klas `TKalkulator_3` i `TKalkulator3_1` oraz `Dzialania_kalkulatora`



Serializacja i deserializacja obiektów
TKalkulator_3 i *TKalkulator3_1*

```
Console::WriteLine(L"Pliki");

App::Serialize(a);
TKalkulator_3^ c;
c=App::Deserializel();
Dzialania_kalkulatora^ C = gcnew Dzialania_kalkulatora(c);
Console::WriteLine(C->kalkulator->info);

App::Serialize(b);
TKalkulator3_1^ d;
d= safe_cast<TKalkulator3_1^>(App::Deserializel());
Dzialania_kalkulatora^ D = gcnew Dzialania_kalkulatora(d);
Console::WriteLine(D->kalkulator->info);
pom = safe_cast<TKalkulator3_1^>(D->kalkulator);
Console::Write(pom->archiwum(0));
Console::Write(safe_cast <wchar_t>(pom->archiwum(1)));
Console::Write(pom->archiwum(2));
Console::Write(L"=");
Console::WriteLine(pom->archiwum(3));
```

```
C:\>
1 * 1 = 1
1 + 1 = 2
1 + 1 = 2
Pliki
1 * 1 = 1
1 + 1 = 2
1 + 1 = 2
1 + 1 = 2
1 + 1 = 2
```

Serializacja i deserializacja obiektu typu *Dzialania_kalkulatora* z obiektem typu *TKalkulador_3* oraz obiektu typu *Dzialania_kalkulatora* z obiektem typu *TKalkulador3_1*

(Global Scope)

```
App::Serialize (A) ;  
C = nullptr ;  
C = App::Deserialize2 () ;  
Console::WriteLine (C->kalkulator->info) ;  
  
App::Serialize (B) ;  
D = nullptr ;  
D = App::Deserialize2 () ;  
Console::WriteLine (D->kalkulator->info) ;  
pom = safe_cast<TKalkulador3_1^> (D->kalkulator) ;  
Console::Write (pom->archiwum (0) ) ;  
Console::Write (safe_cast <wchar_t> (pom->archiwum (1) ) ) ;  
Console::Write (pom->archiwum (2) ) ;  
Console::Write (L"=") ;  
Console::WriteLine (pom->archiwum (3) ) ;  
return 0 ;  
}
```

```
1 * 1 = 1  
1 + 1 = 2  
Pliki  
1 * 1 = 1  
1 + 1 = 2  
1 + 1 = 2  
1 + 1 = 2  
1 + 1 = 2  
1 + 1 = 2
```

Ready

Ln 64

Col 5

Ch 5