Wykład 6

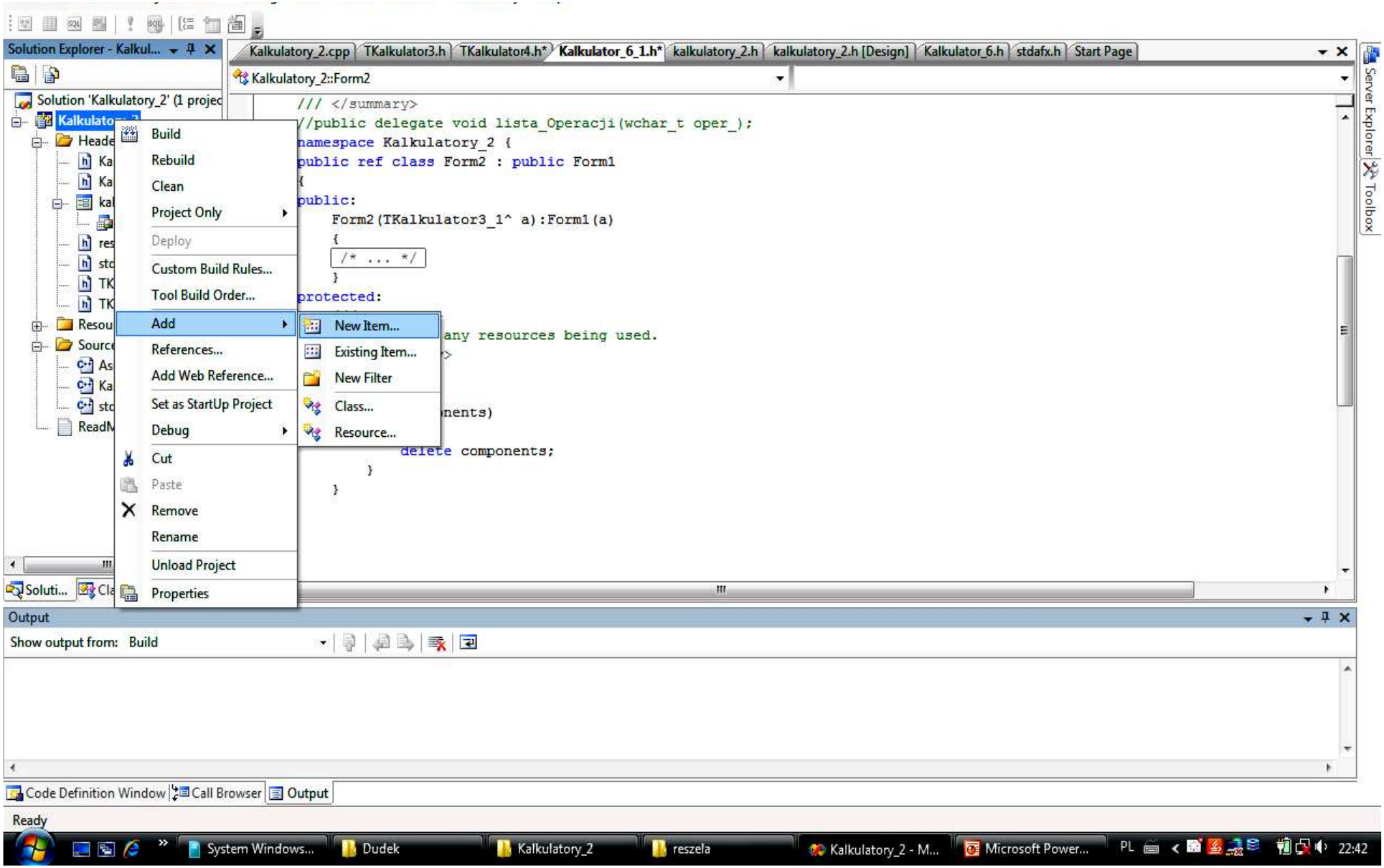
Dziedziczenie – cd., pliki

Autor: Zofia Kruczkiewicz

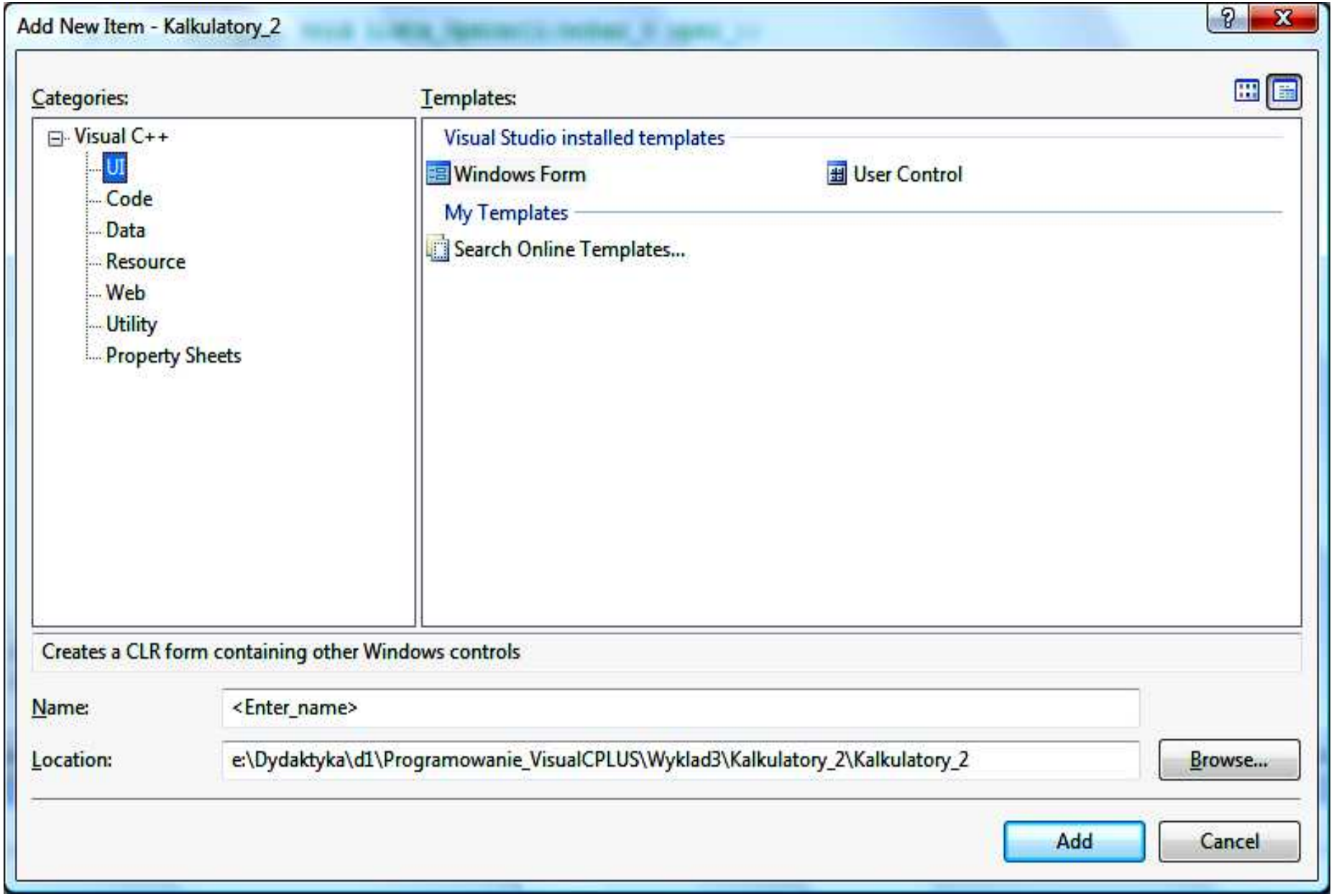
Zagadnienia

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

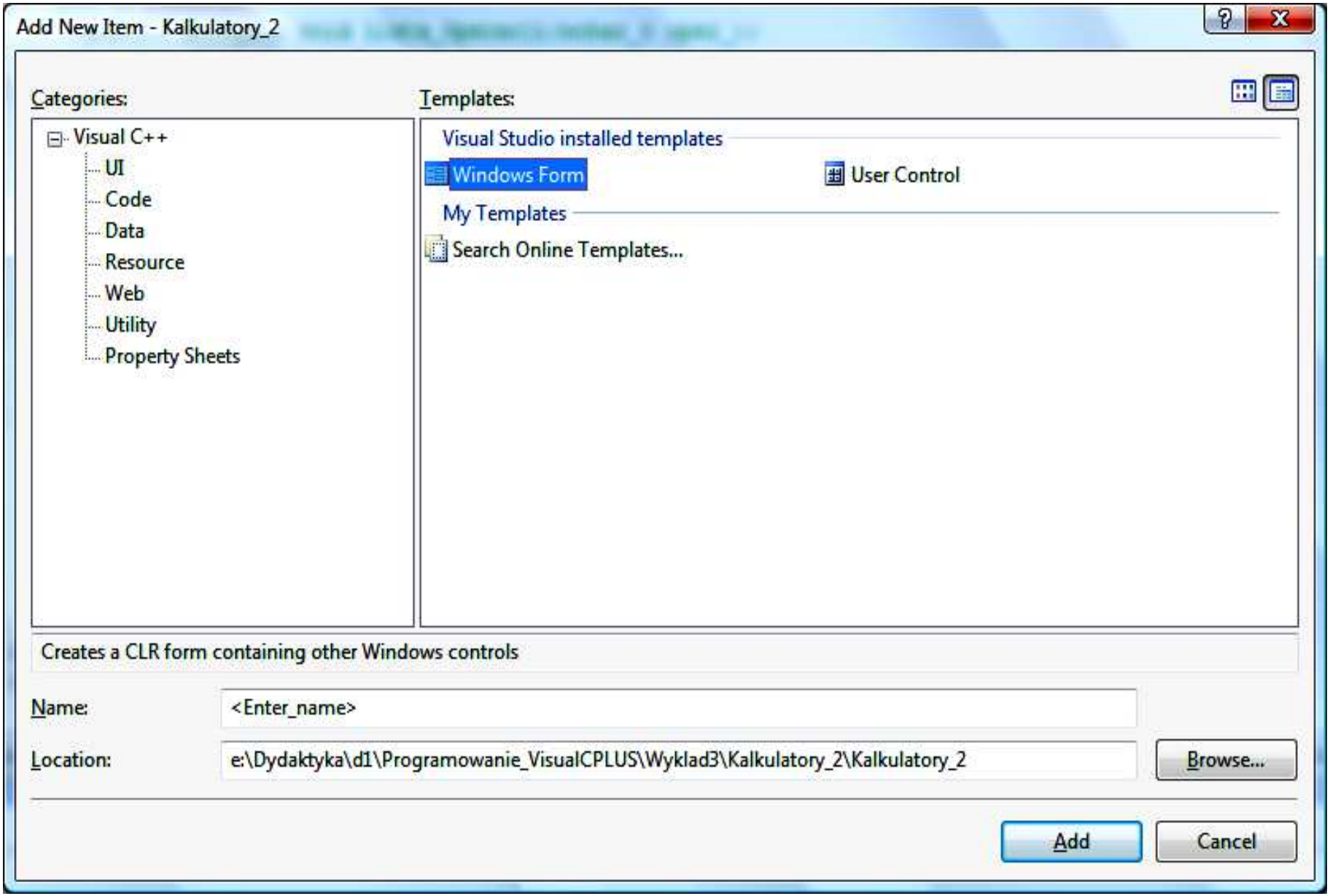
1. Dziedziczenie – aplikacja Kalkultory_2 typu Windows Forms prezentująca dziedziczenie formularzy obsługujących rodzinę kalkulatorów TKalkulator_3 oraz TKalkulator3_1



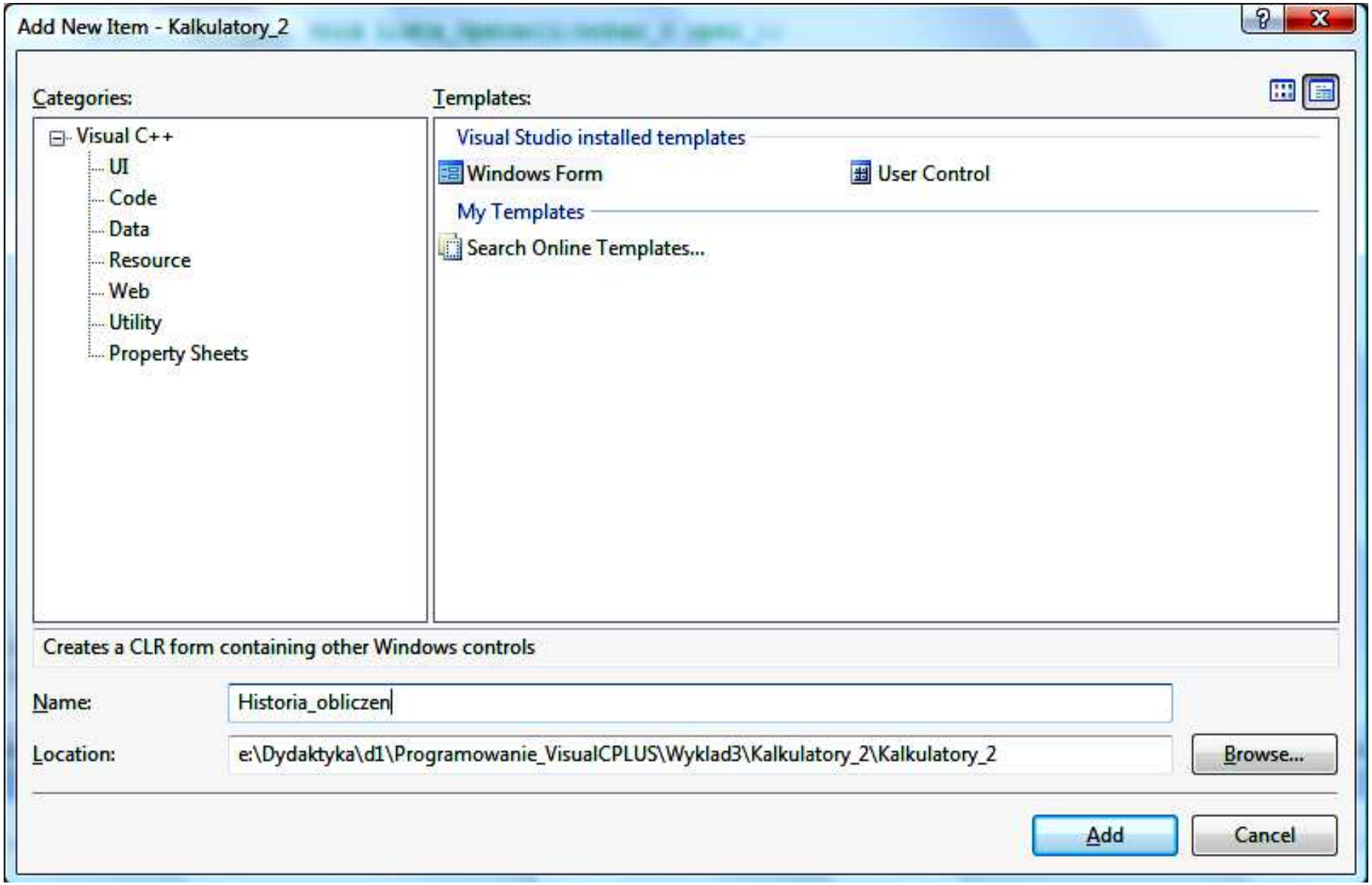
Wstawianie nowego formularza typu Windows Forms (1)



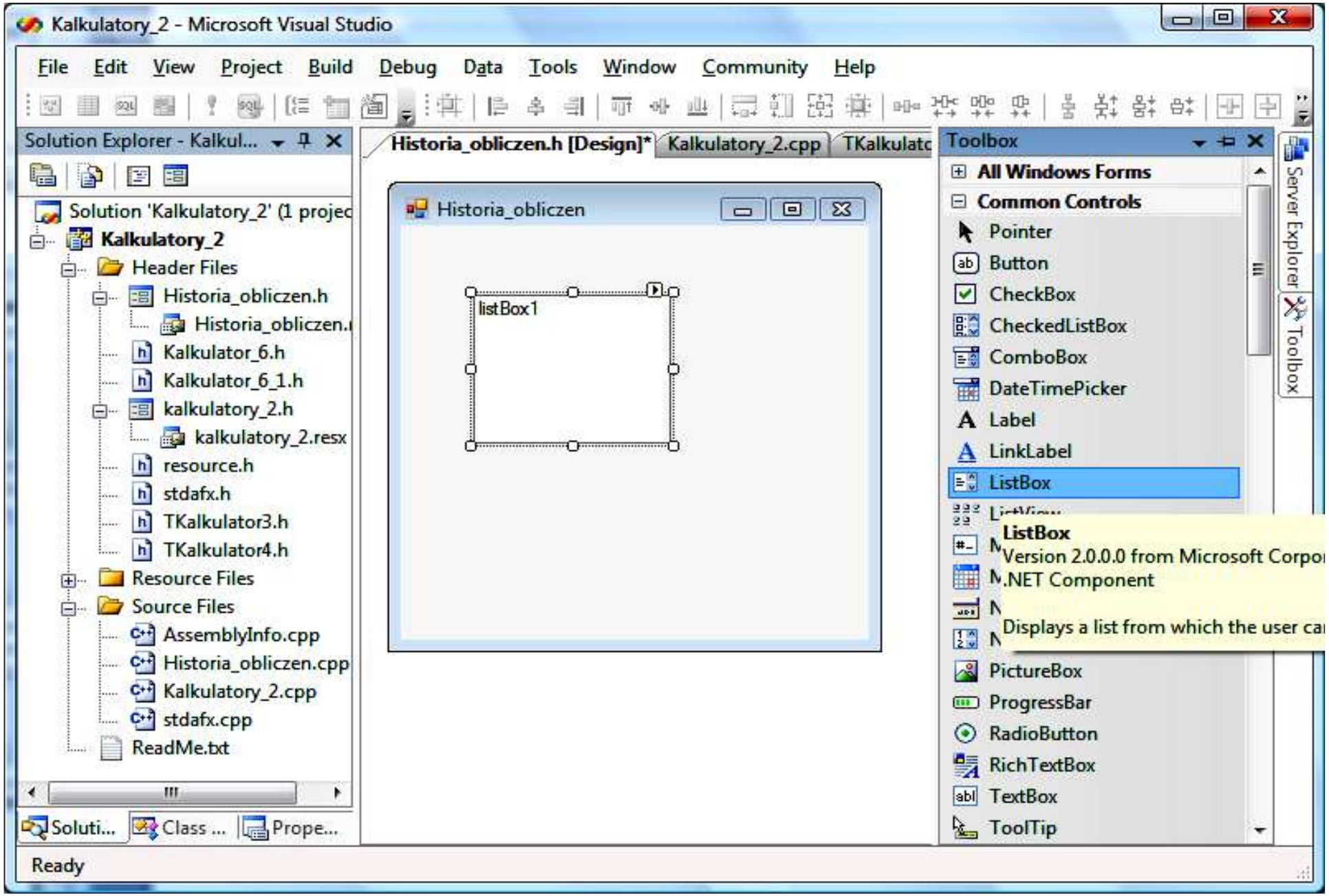
Wstawianie nowego formularza typu Windows Forms (2)



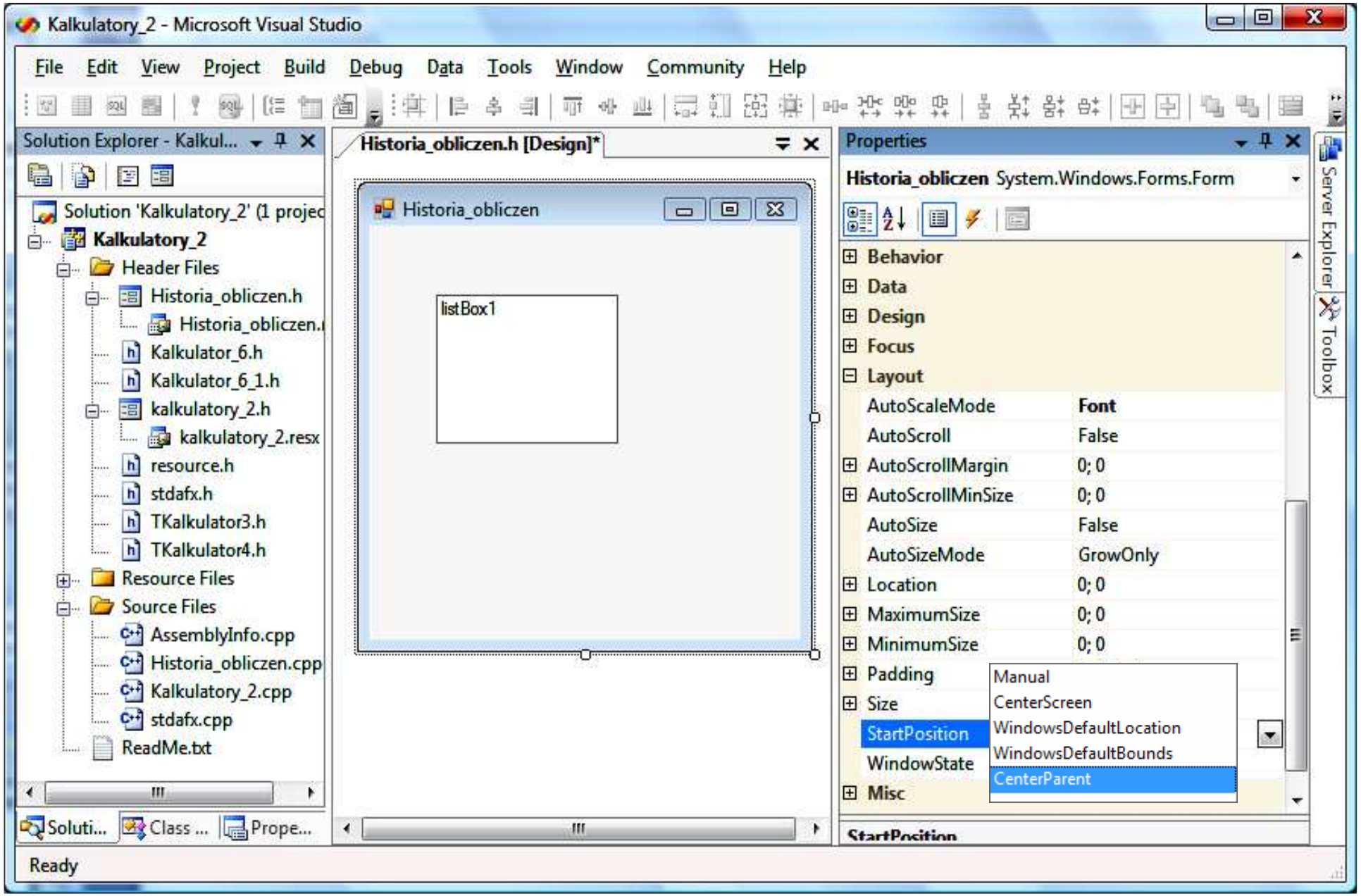
Wstawianie nowego formularza typu Windows Forms (3) w celu przekształcenia go w okno dialogowe *Historia_obliczen* wywoływane z formularza kalkulatora pochodnego typu TKalkulator3_1



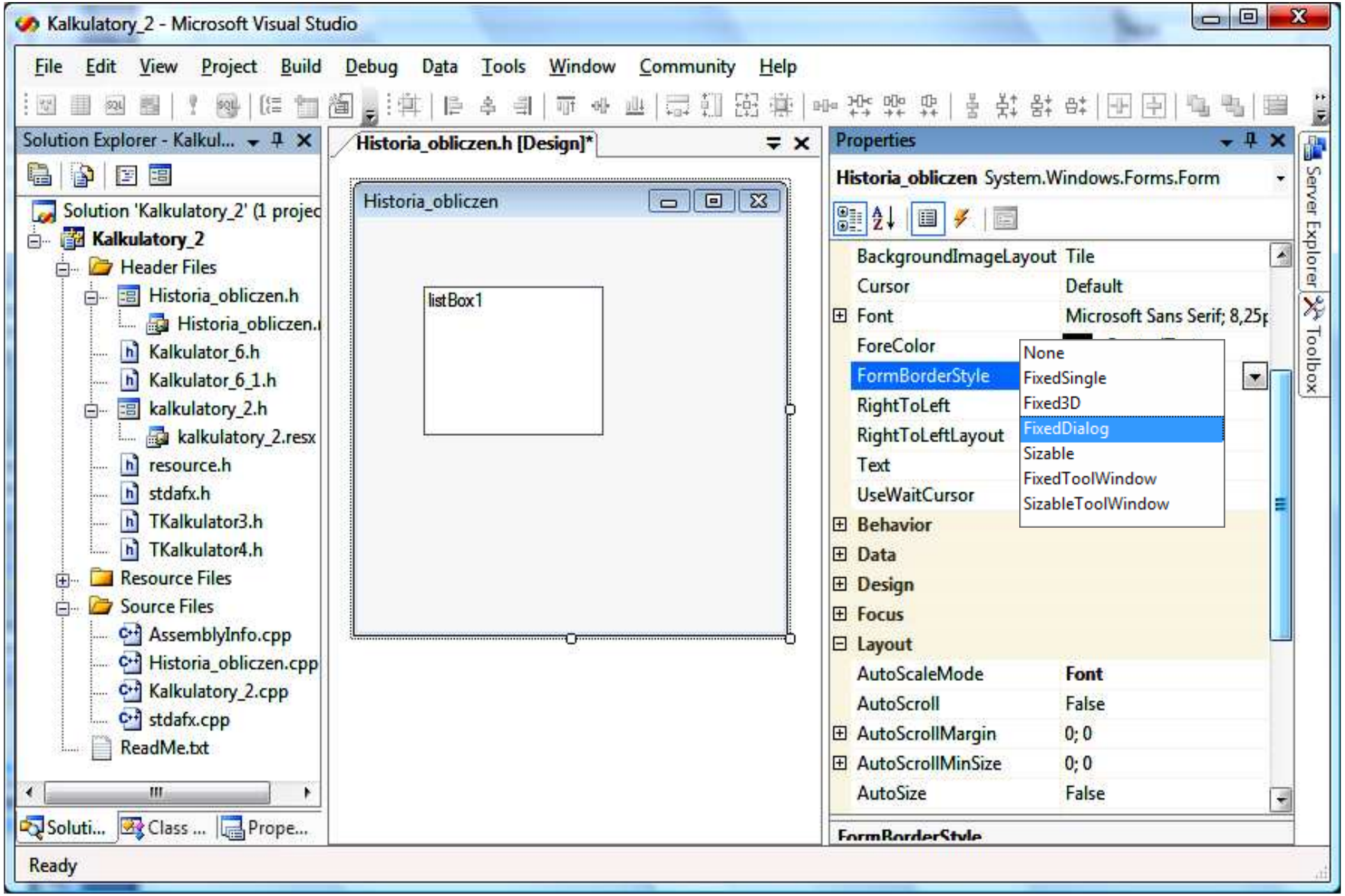
Wstawianie do formularza komponentu typu **ListBox**



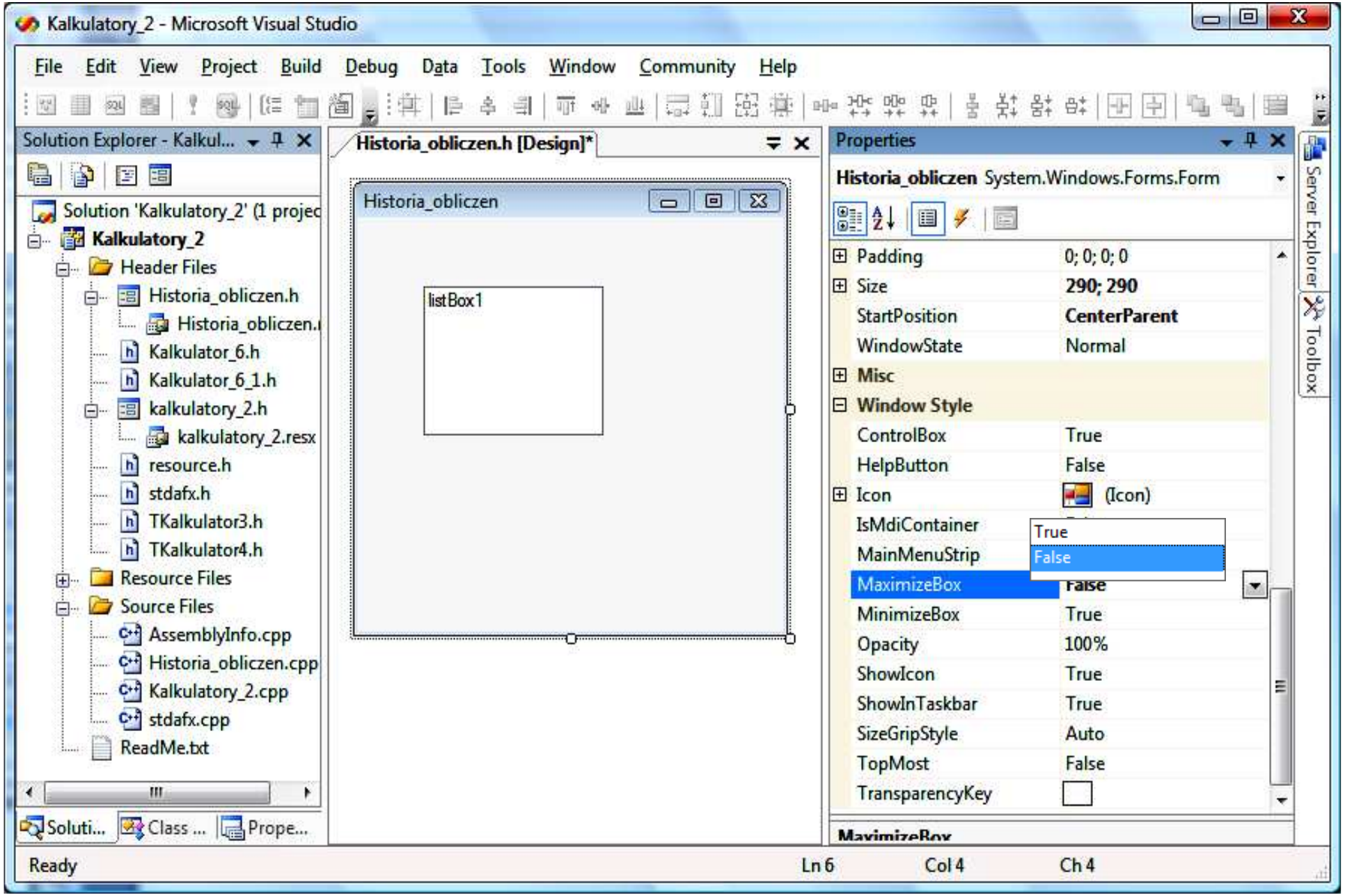
Tworzenie okna dialogowego typu ***Historia_obliczen*** wywoływanego z formularza kalkulatora pochodnego **TKalkulator3_1 (1)**



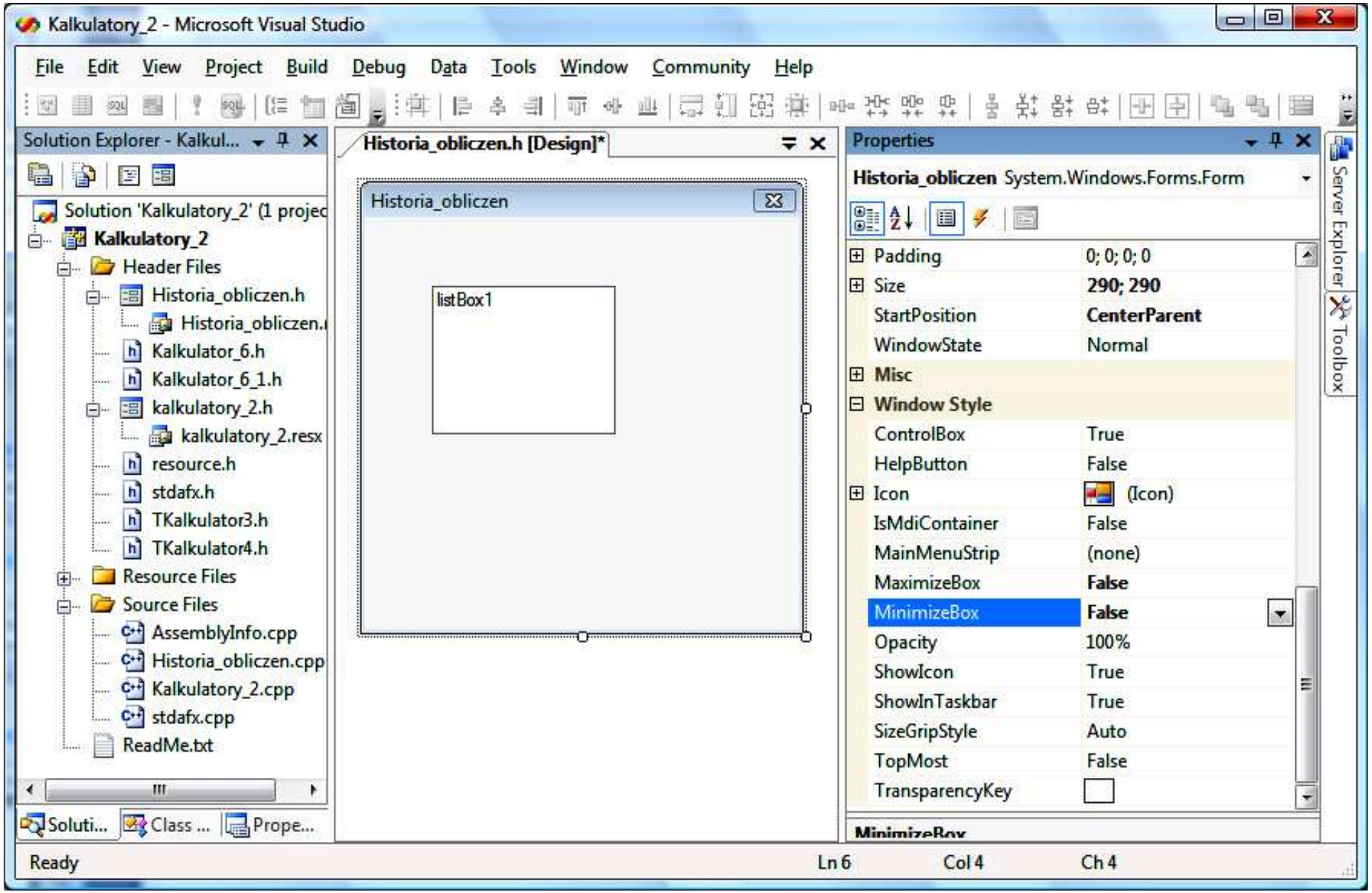
Tworzenie okna dialogowego typu ***Historia_obliczen*** wywoływanego z formularza kalkulatora pochodnego **TKalkulator3_1 (2)**



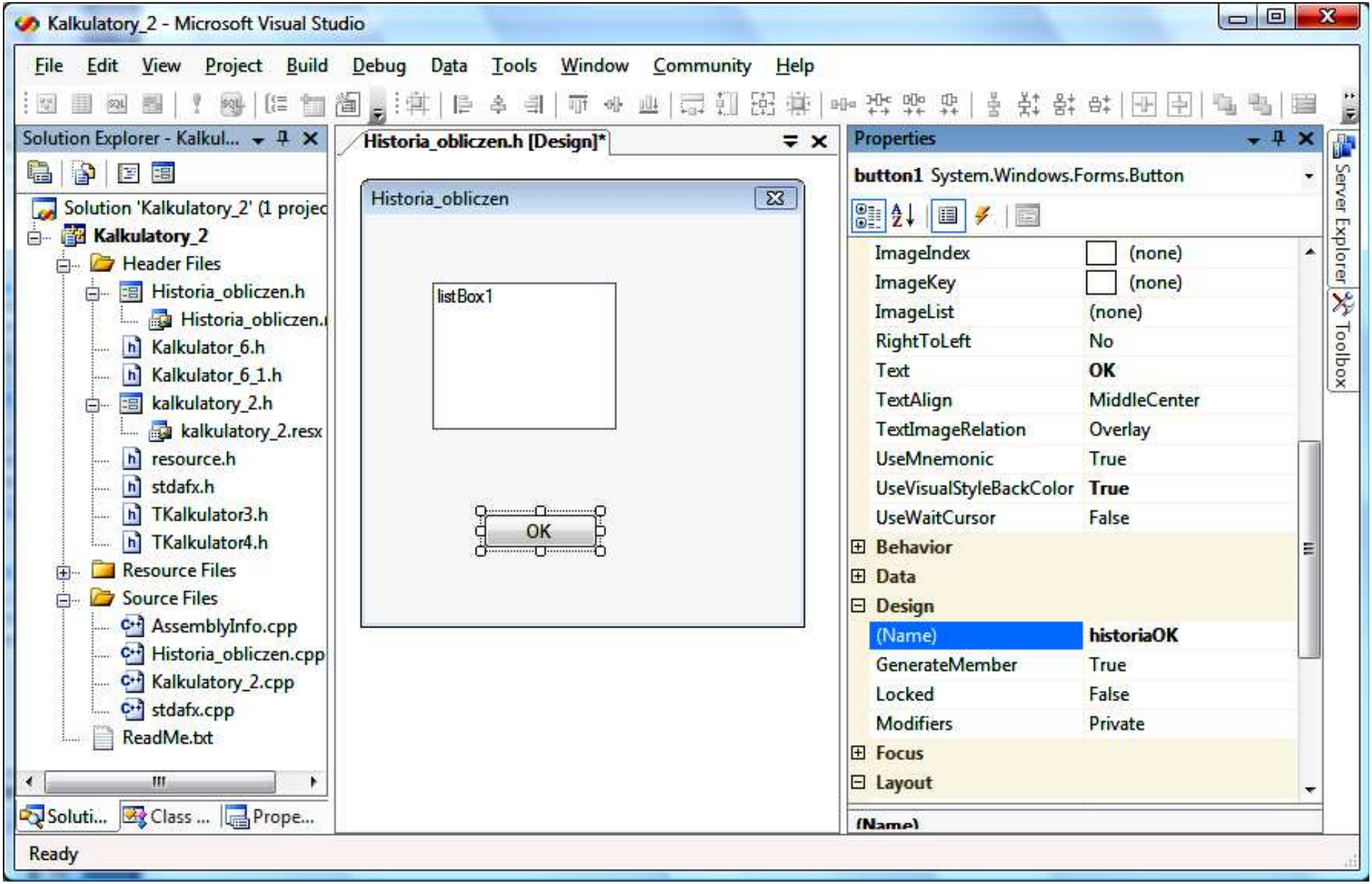
Tworzenie okna dialogowego typu ***Historia_obliczen*** wywoływanego z formularza kalkulatora pochodnego **TKalkulator3_1 (3)**



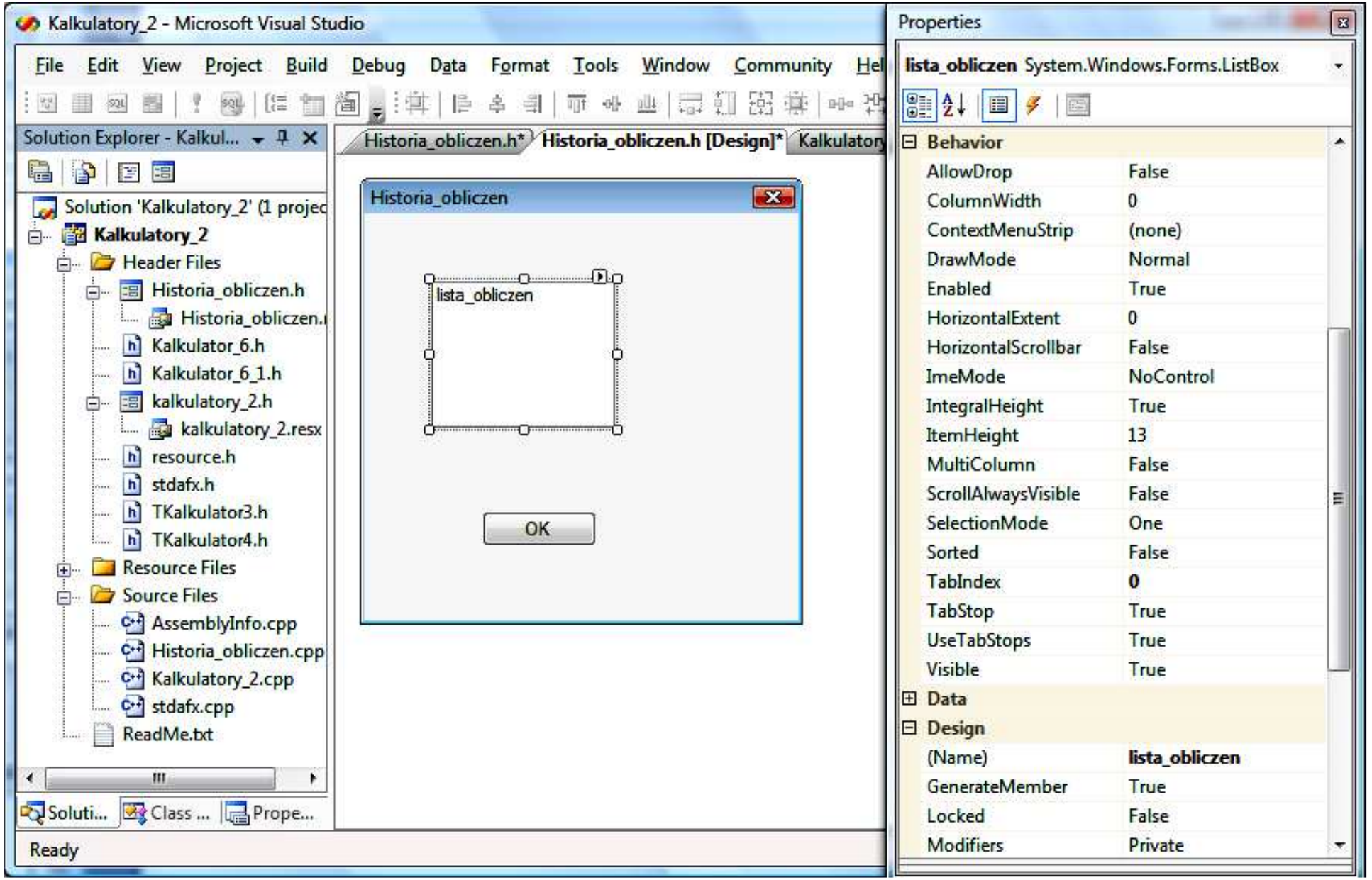
Tworzenie okna dialogowego typu ***Historia_obliczen*** wywoływanego z formularza kalkulatora pochodnego **TKalkulator3_1 (4)**



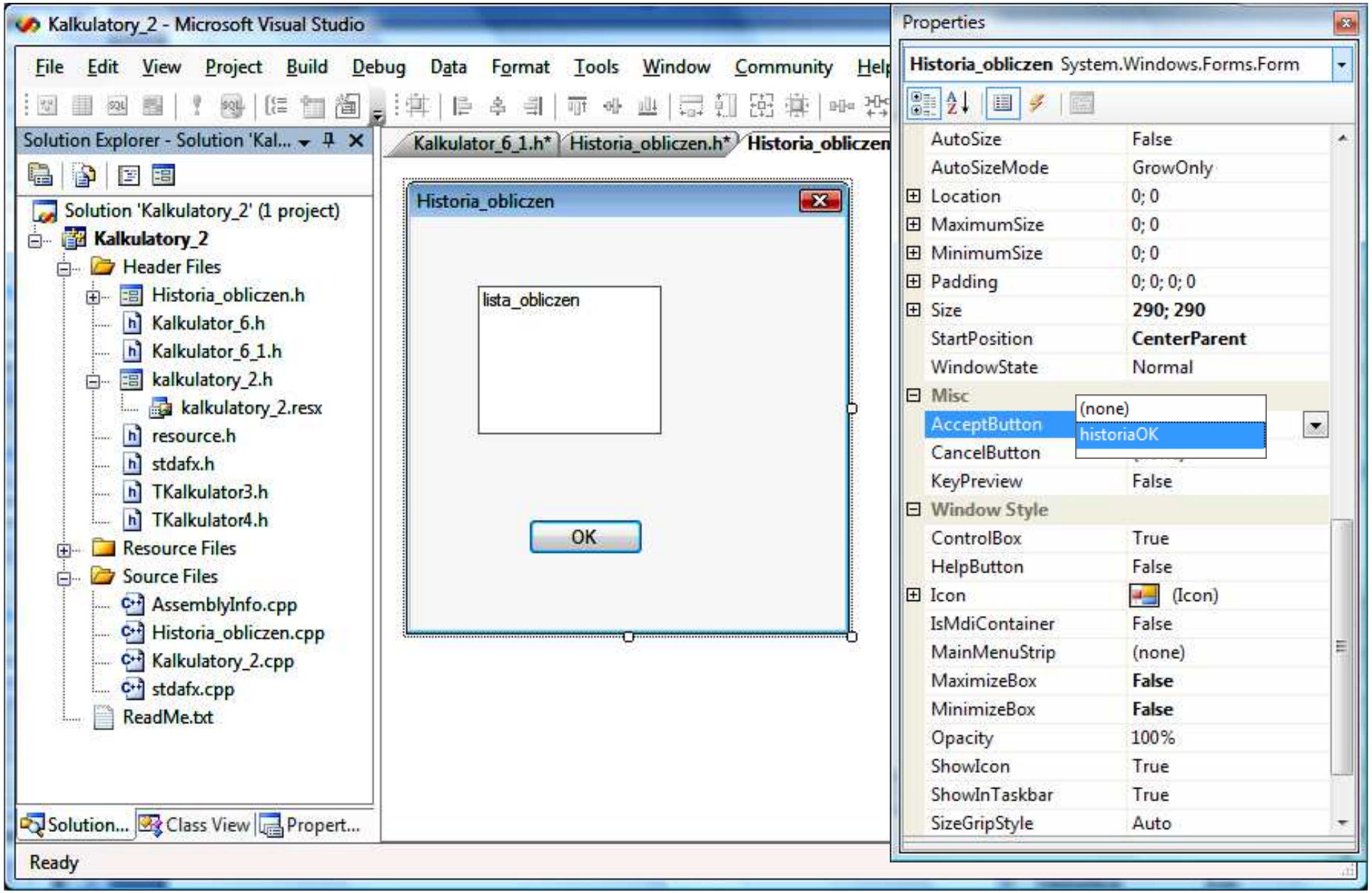
Tworzenie okna dialogowego typu ***Historia_obliczen*** wywoływanego z formularza kalkulatora pochodnego **TKalkulator3_1 (5)**



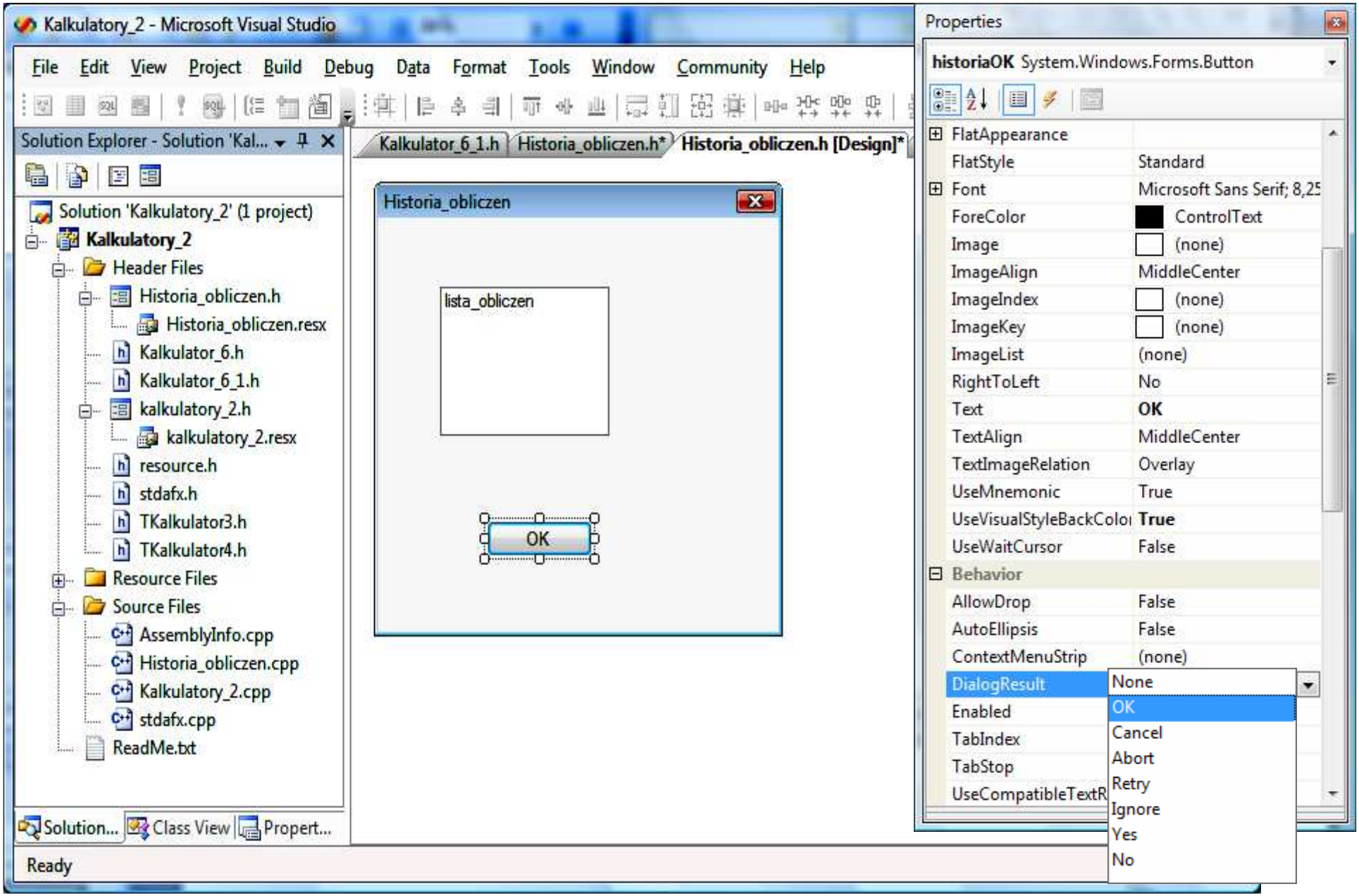
Tworzenie okna dialogowego typu ***Historia_obliczen*** wywoływanego z formularza kalkulatora pochodnego **TKalkulator3_1 (6)**



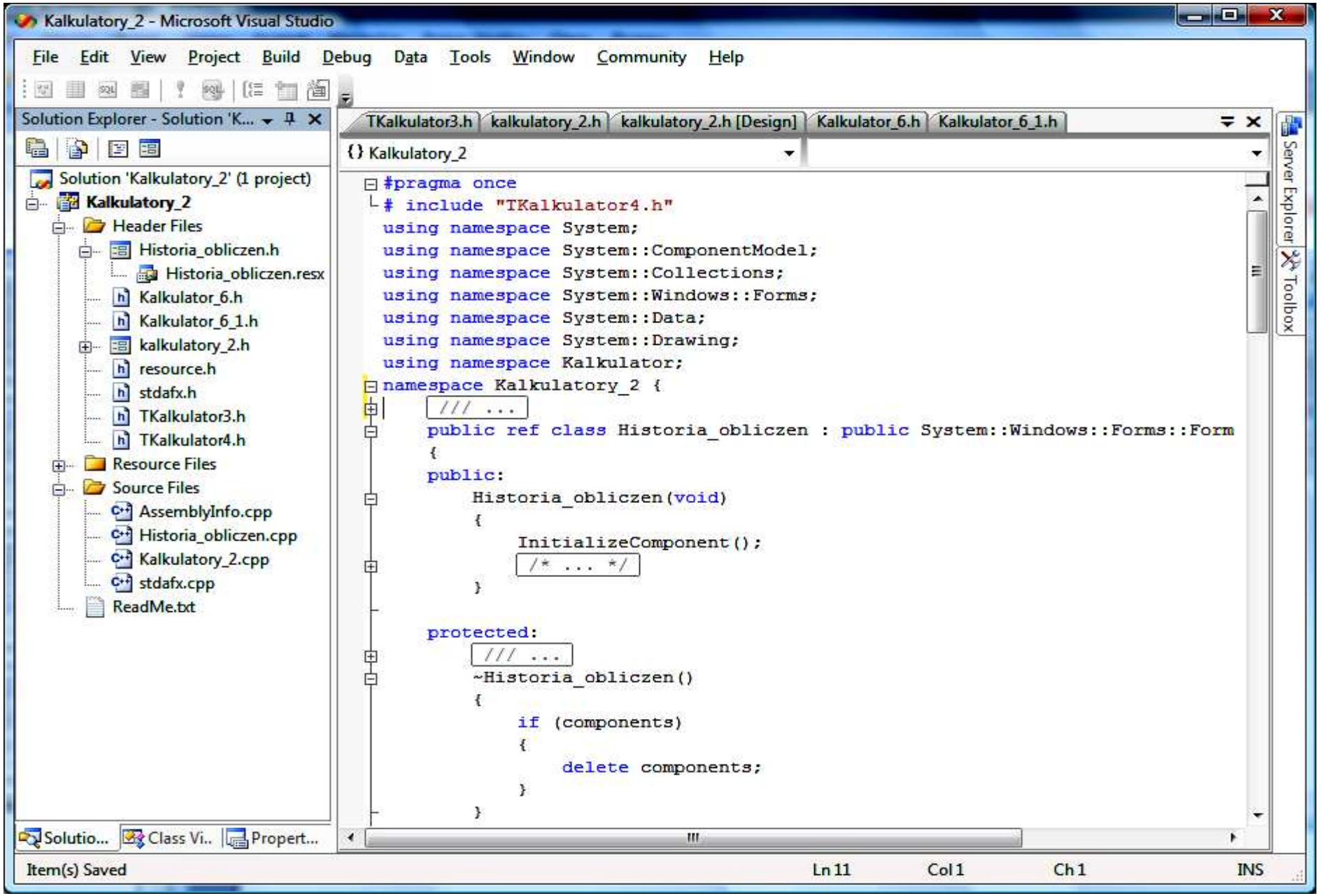
Tworzenie okna dialogowego typu ***Historia_obliczen*** wywoływanego z formularza kalkulatora pochodnego **TKalkulator3_1 (7)**



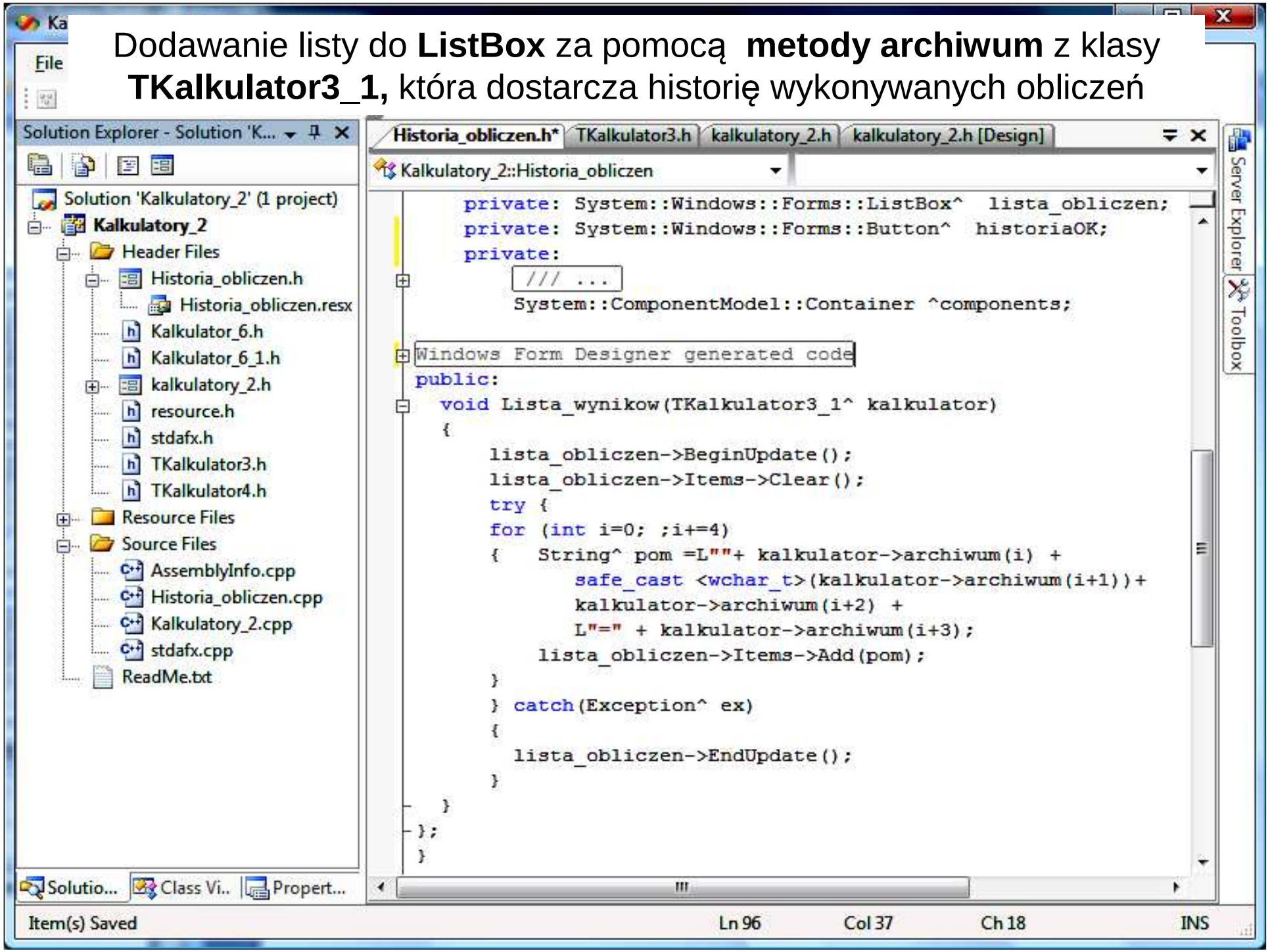
Tworzenie okna dialogowego typu ***Historia_obliczen*** wywoływanego z formularza kalkulatora pochodnego **TKalkulator3_1 (8)**



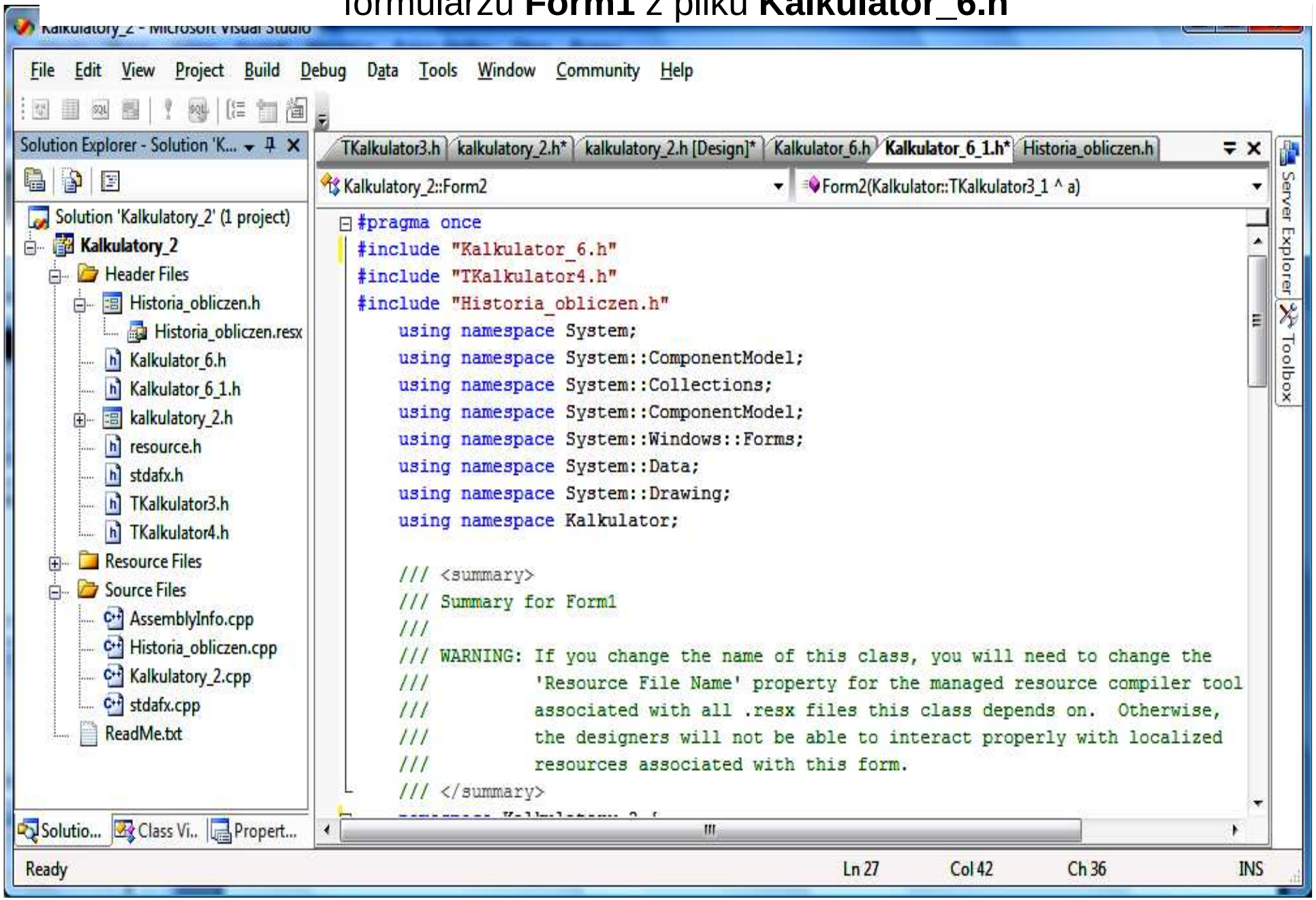
Programowanie okna dialogowego typu **Historia_obliczen**



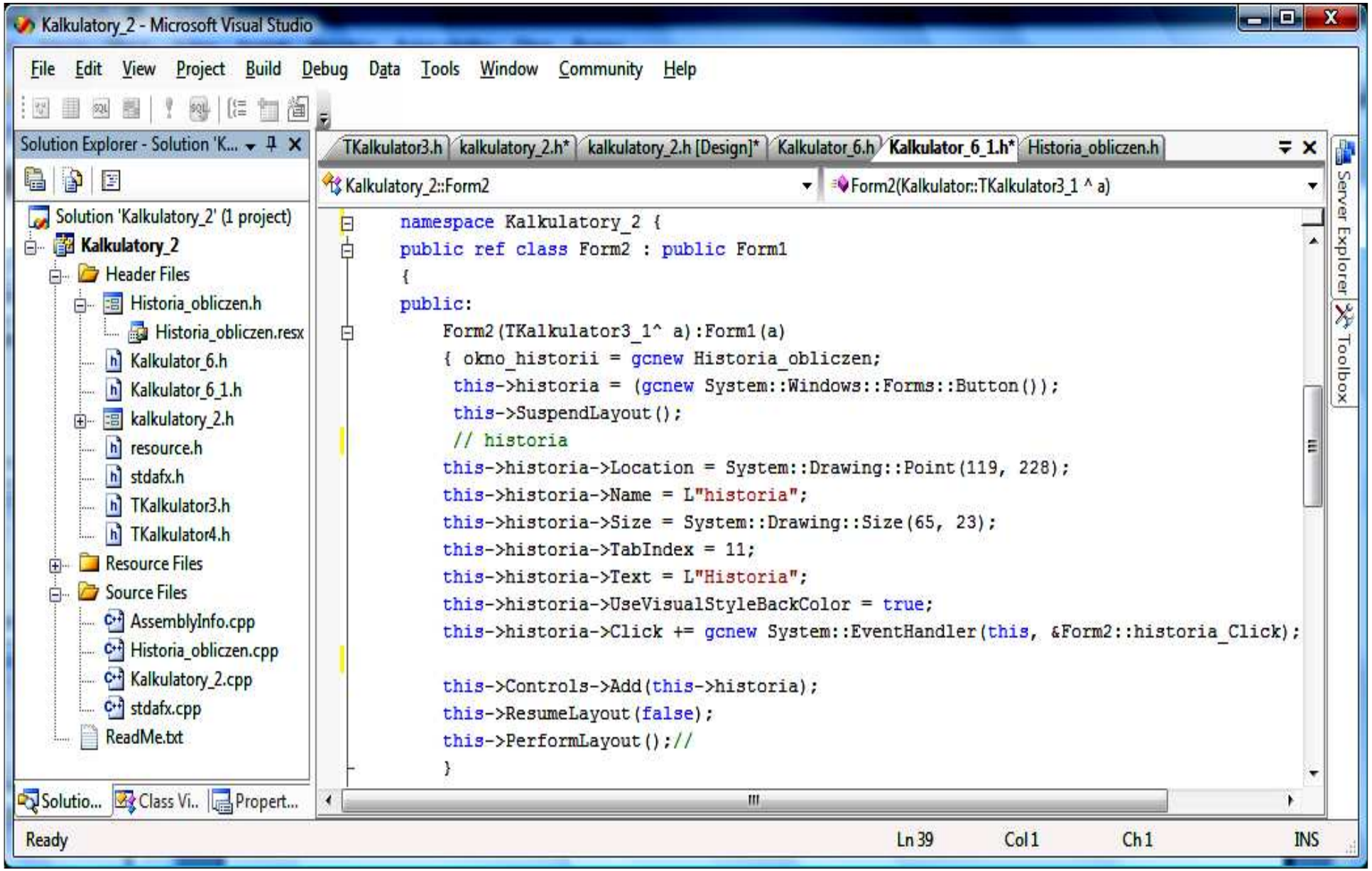
Dodawanie listy do **ListBox** za pomocą **metody archiwum** z klasy **TKalkulator3_1,** która dostarcza historię wykonywanych obliczeń



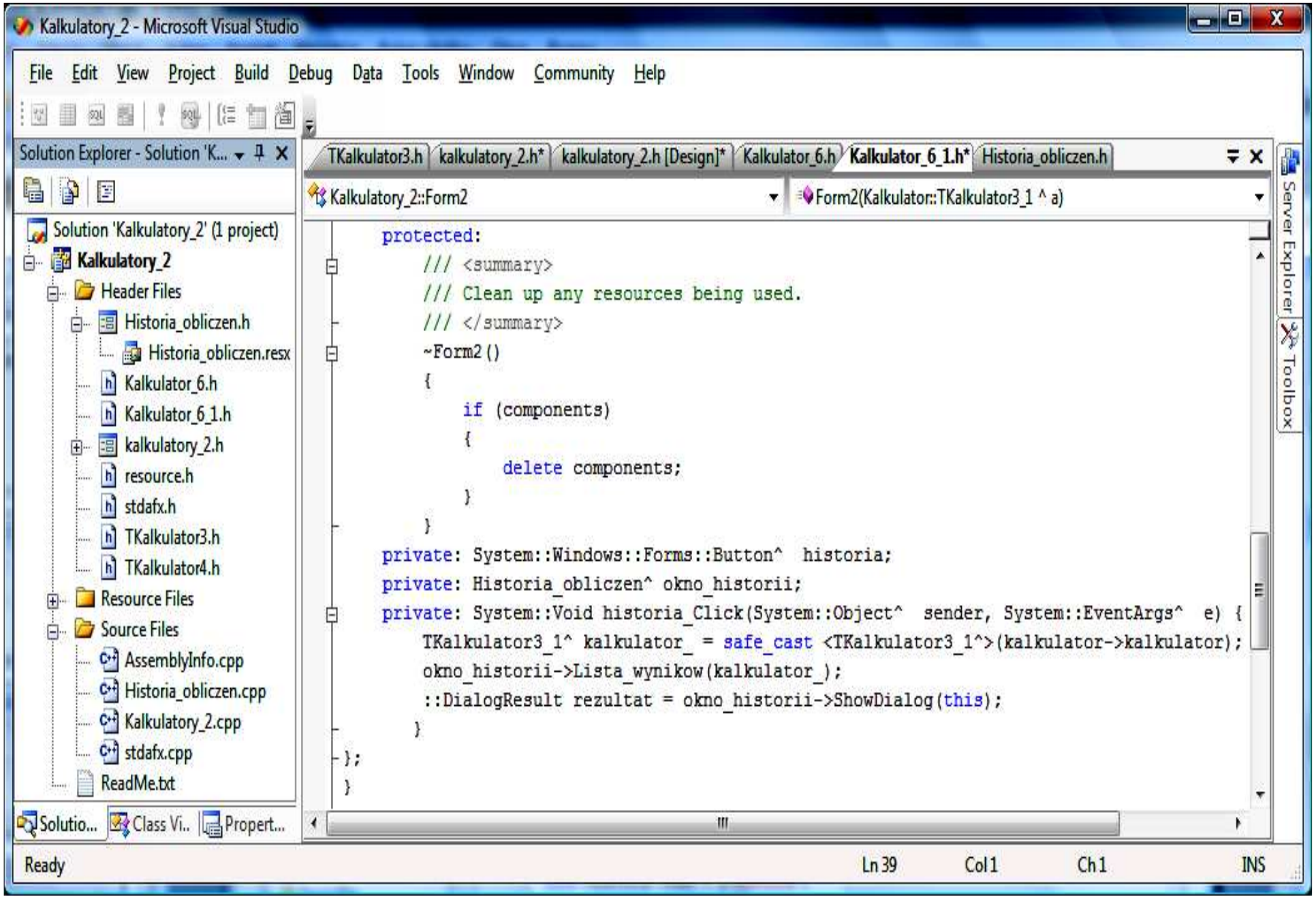
Formularz **Form2** w pliku **Kalkulator_6_1.h** dziedziczący składowe po formularzu **Form1** z pliku **Kalkulator_6.h**



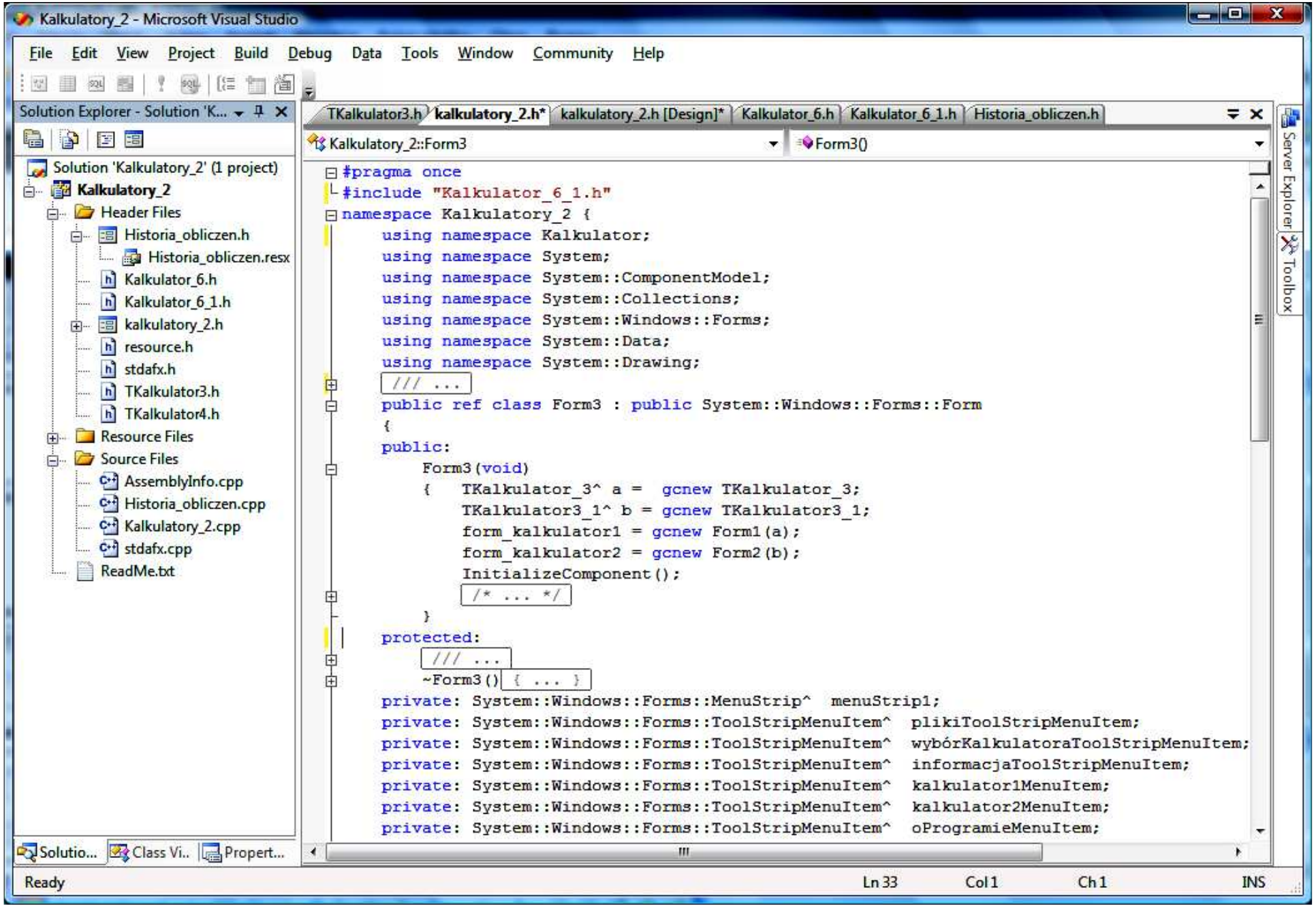
Uzupełnienie inicjalizacji nowego formularza w konstruktorze (ręczne dodanie w kodzie przycisku **historia** typu **Button** wywołującego okno dialogowe ***okno_historii*** z historią obliczeń dla drugiego kalkulatora)



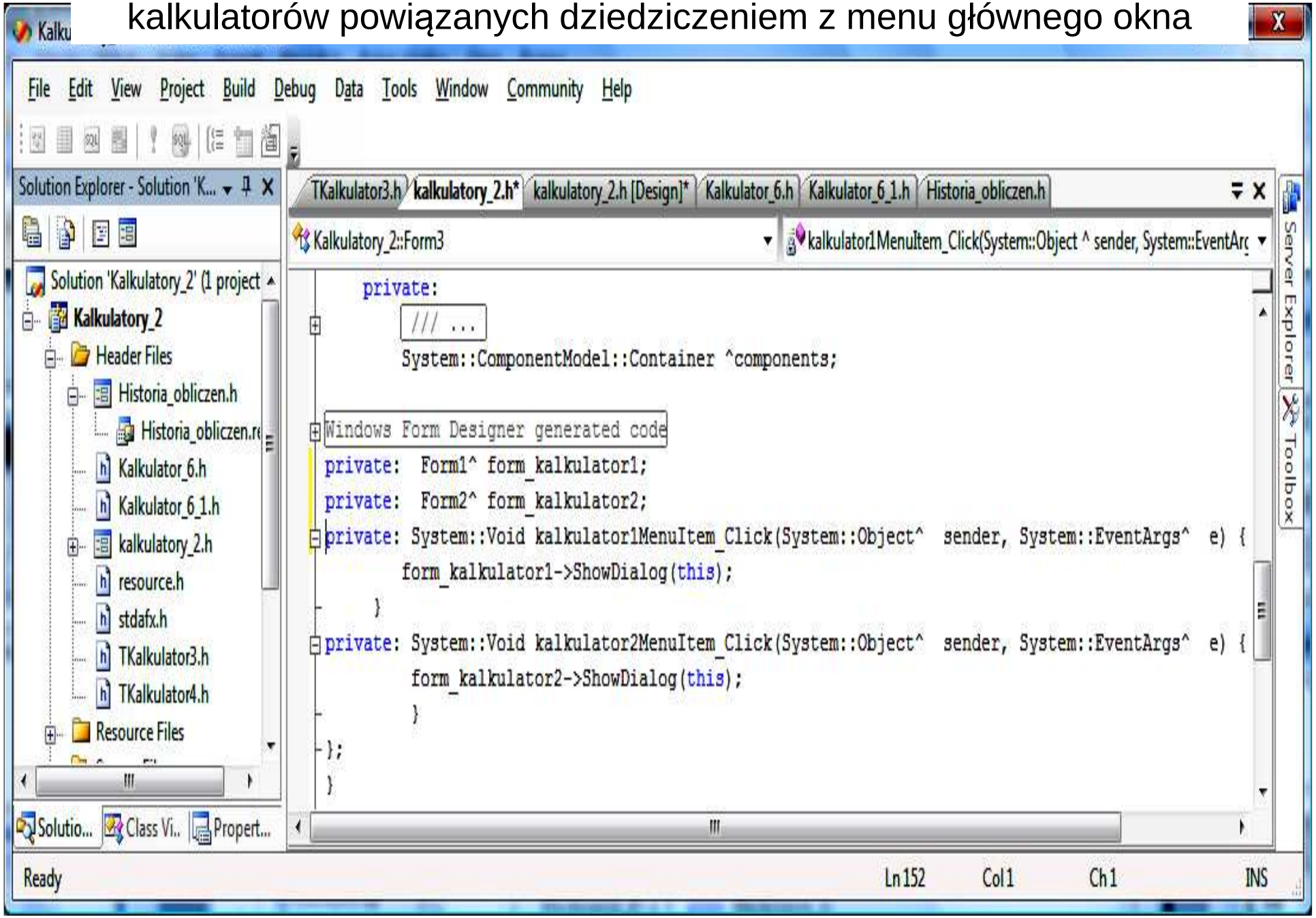
Obsługa zdarzenia klikania na przycisk **historia** w formularzu **Form2**



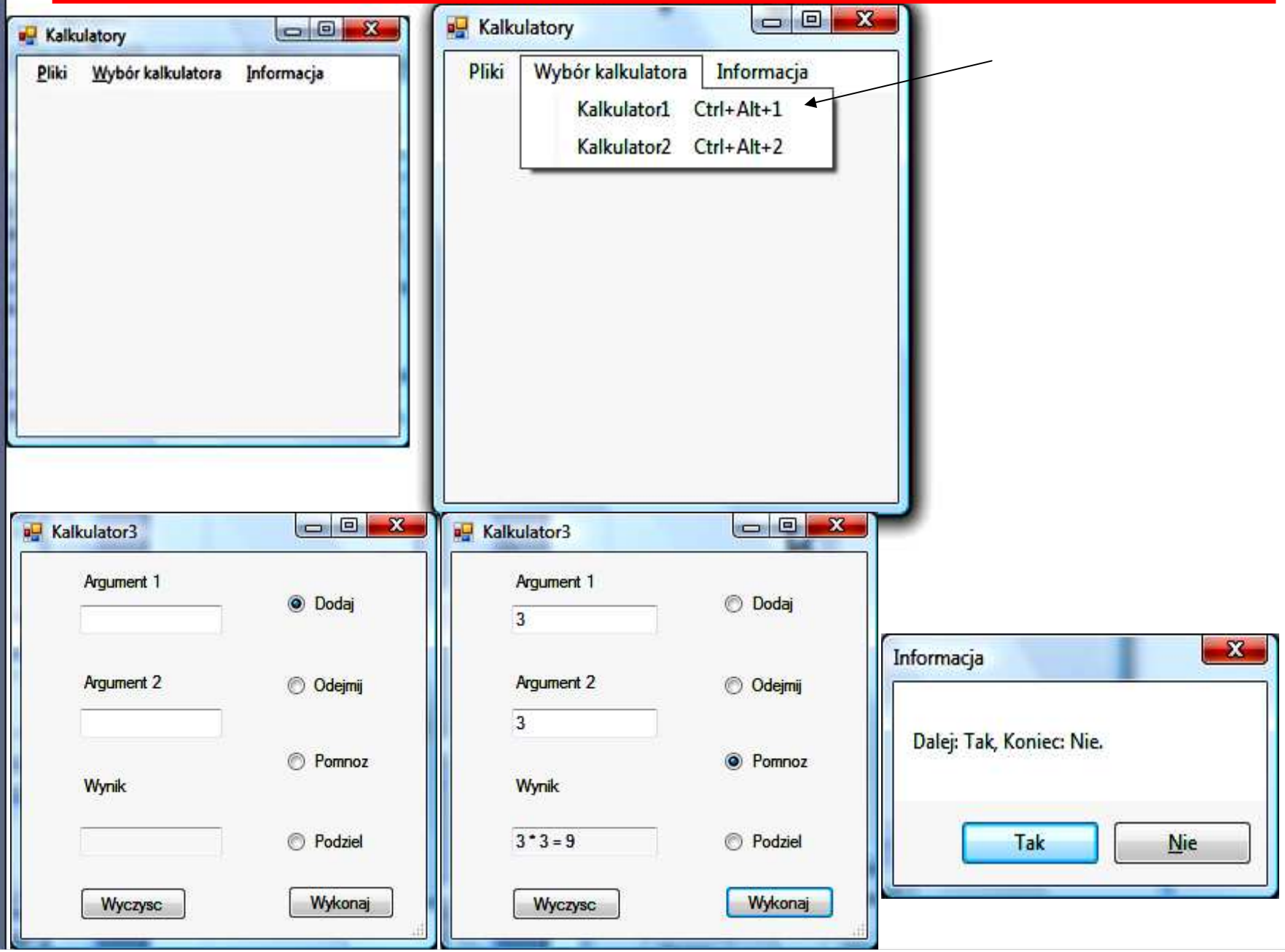
Aplikacja **Kalkulatory_2** z formularzem typu **Form3** z dwoma kalkulatorami



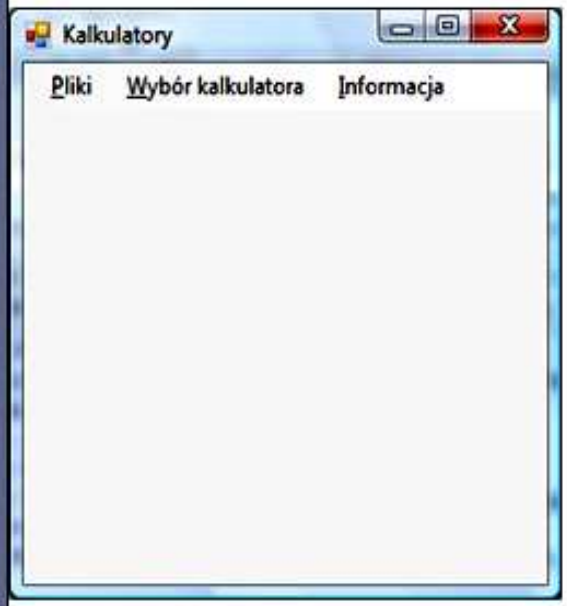
Obsługa zdarzeń listy podmenu – wywołanie formularzy dwóch kalkulatorów powiązanych dziedziczeniem z menu głównego okna

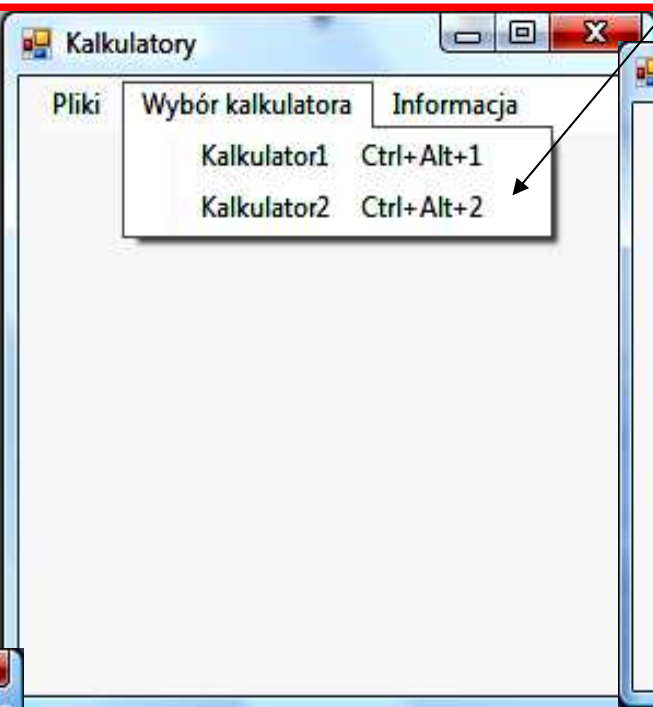


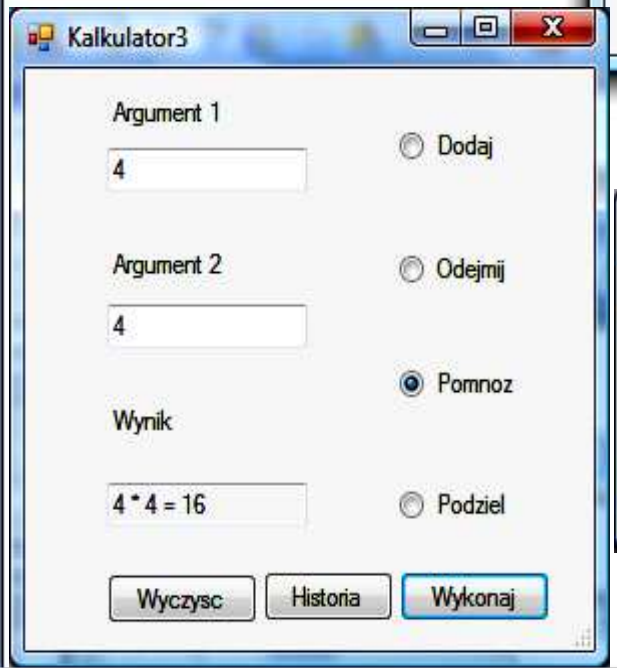
Działanie programu - Wywołanie kalkukatora bez pamięci (Kalkulator1)

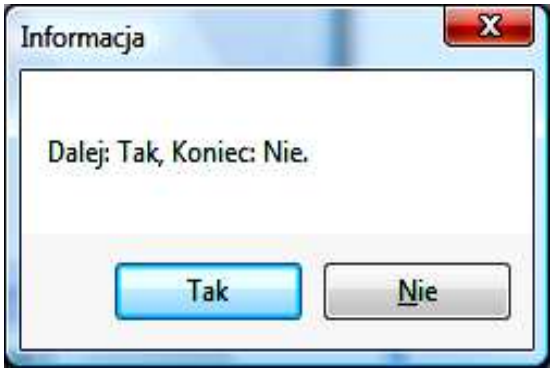


Działanie programu - Wywołanie kalkukatora z pamięcią (Kalkulator2)

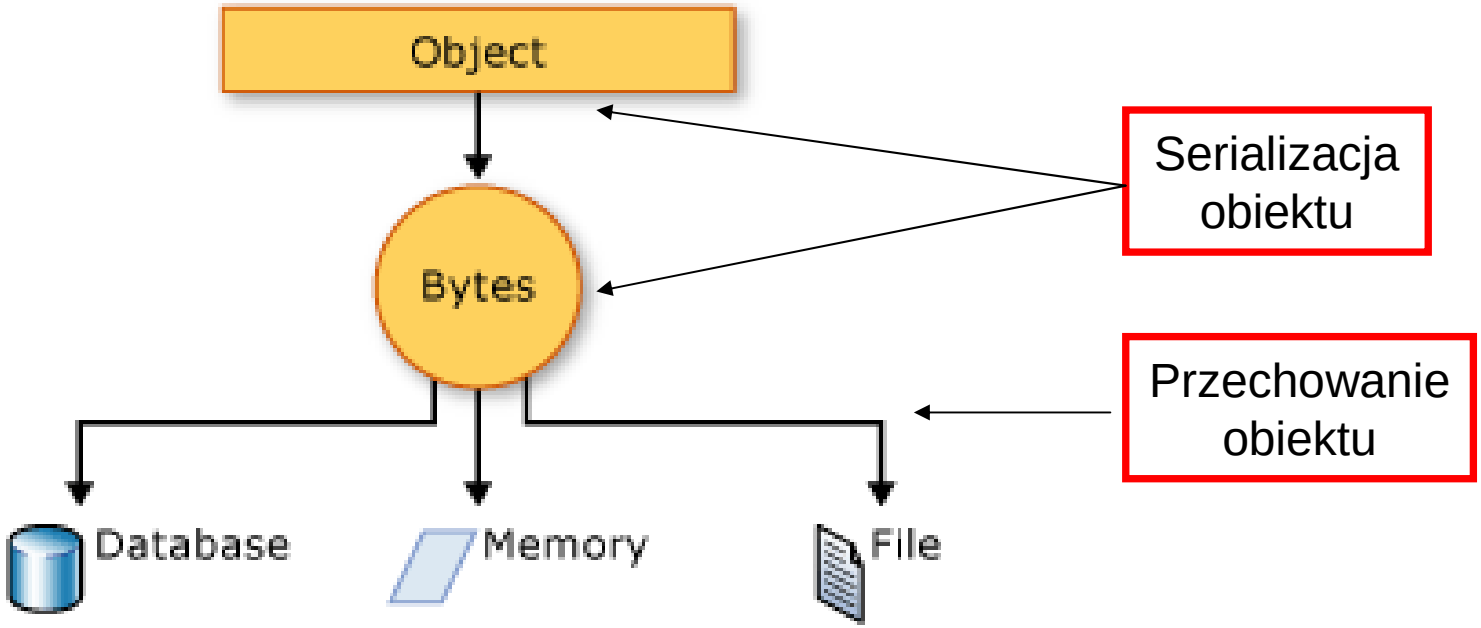








2. Pliki – serializacja (1)



- **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**.

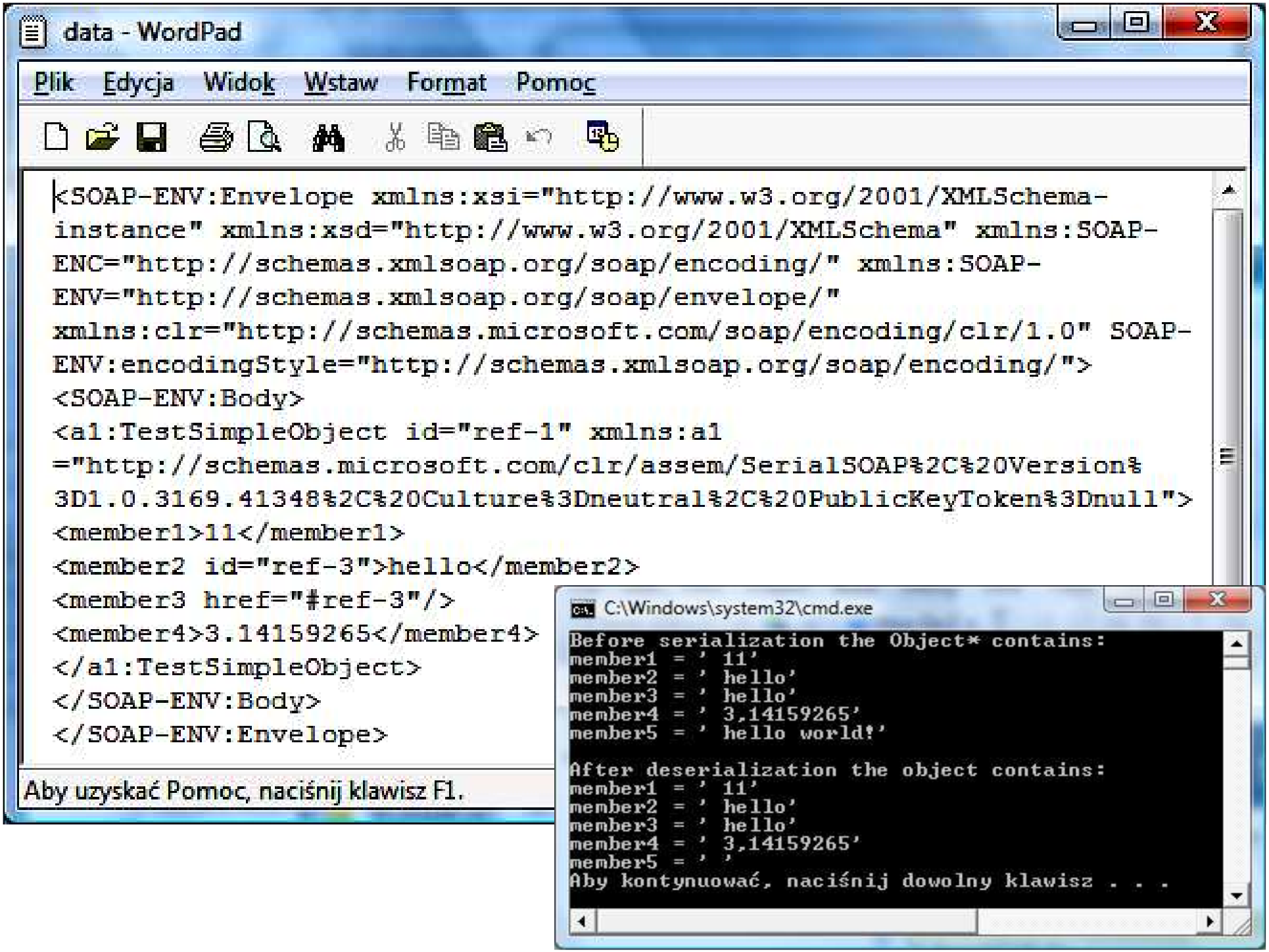
Serializacja SOAP

- 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();
}
```

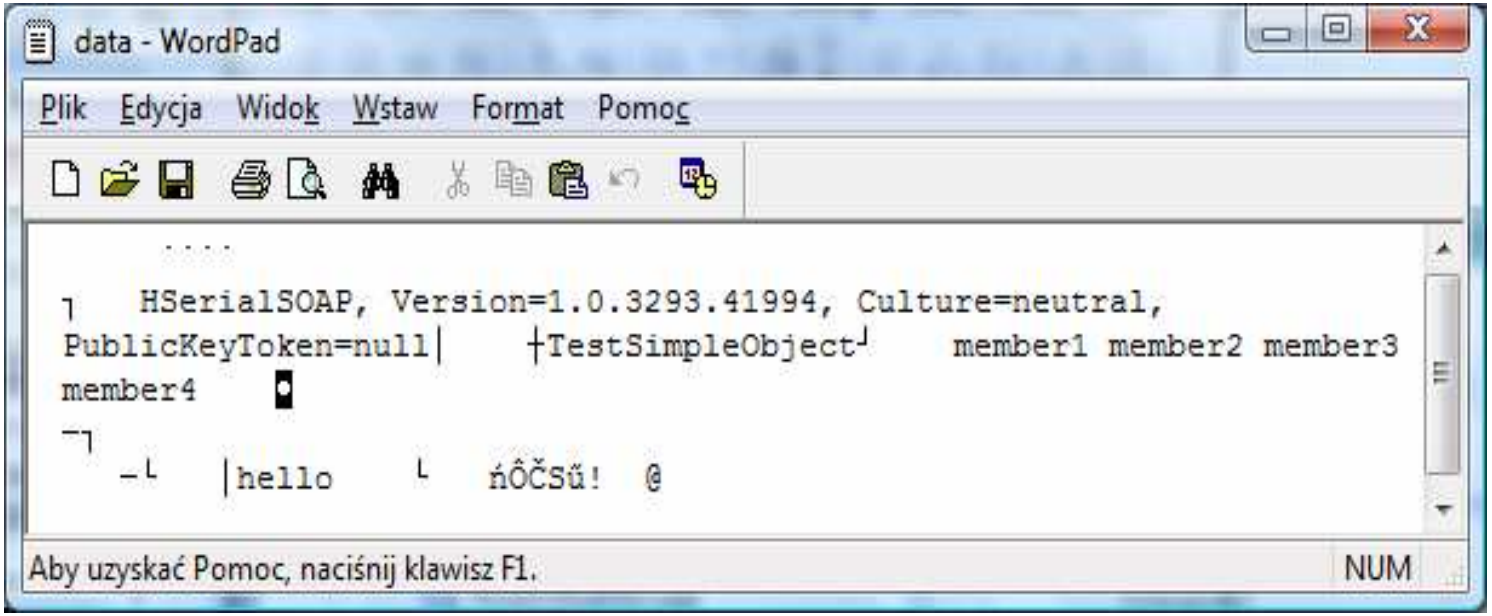

data - WordPad
Plik Edycja Widok Wstaw Format Pomoc
<SOAP-ENV:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:clr="http://schemas.microsoft.com/soap/encoding/clr/1.0" 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 . . .

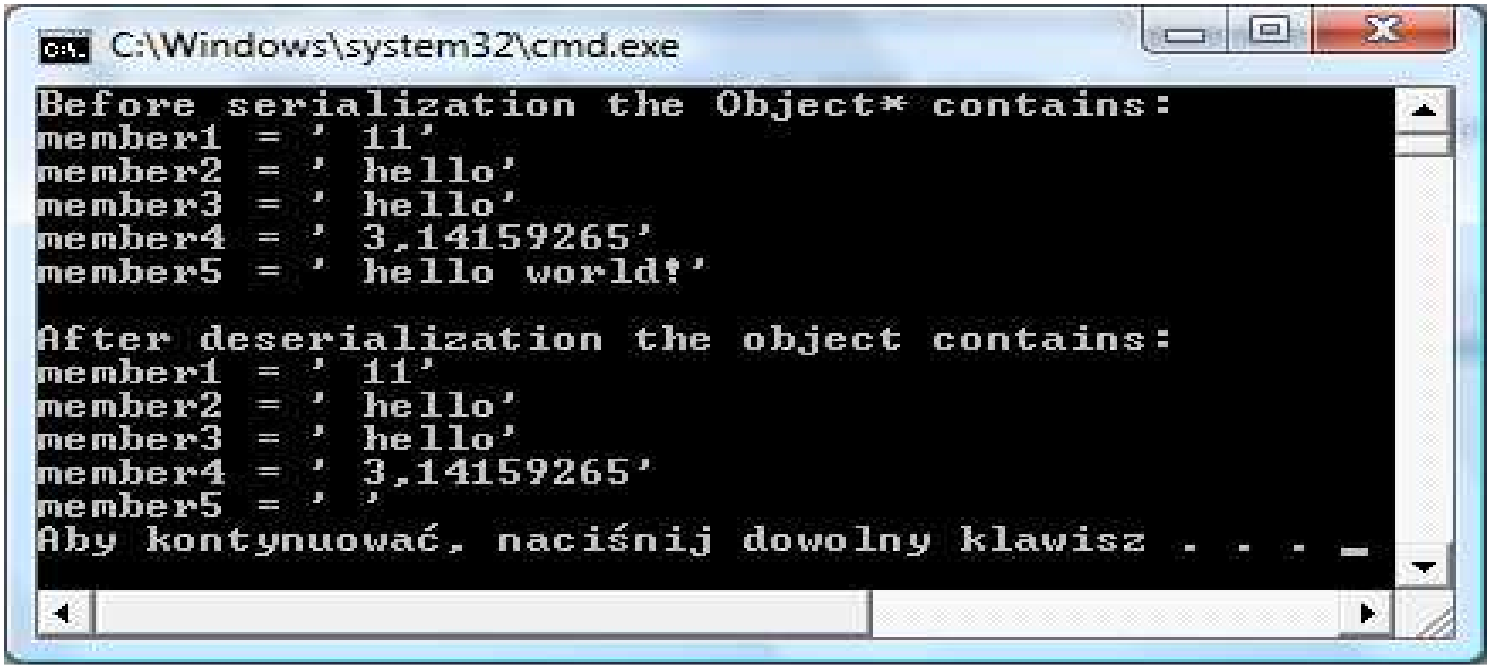
Przykład 2 – na podstawie MSDN (strumienie binarne)

```
#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 ); } };
```

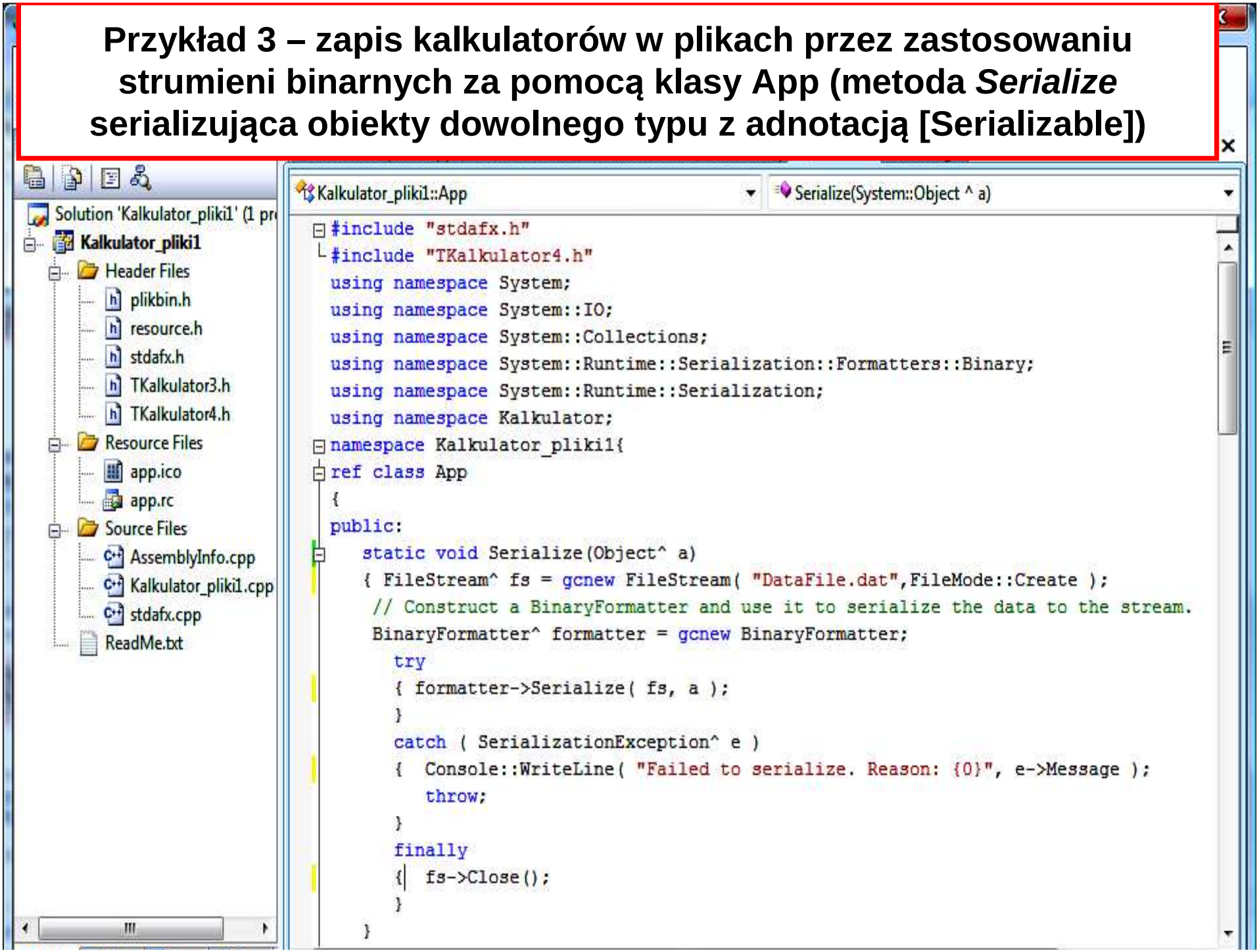
```
int main()
{
 //Creates a new TestSimpleObject object.
    TestSimpleObject^ obj = gcnew TestSimpleObject;
    Console::WriteLine( "Before serialization the Object* contains: " );
    obj->Print();
//Opens a file and serializes the object into it in binary format.
    Stream^ stream = File::Open( "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

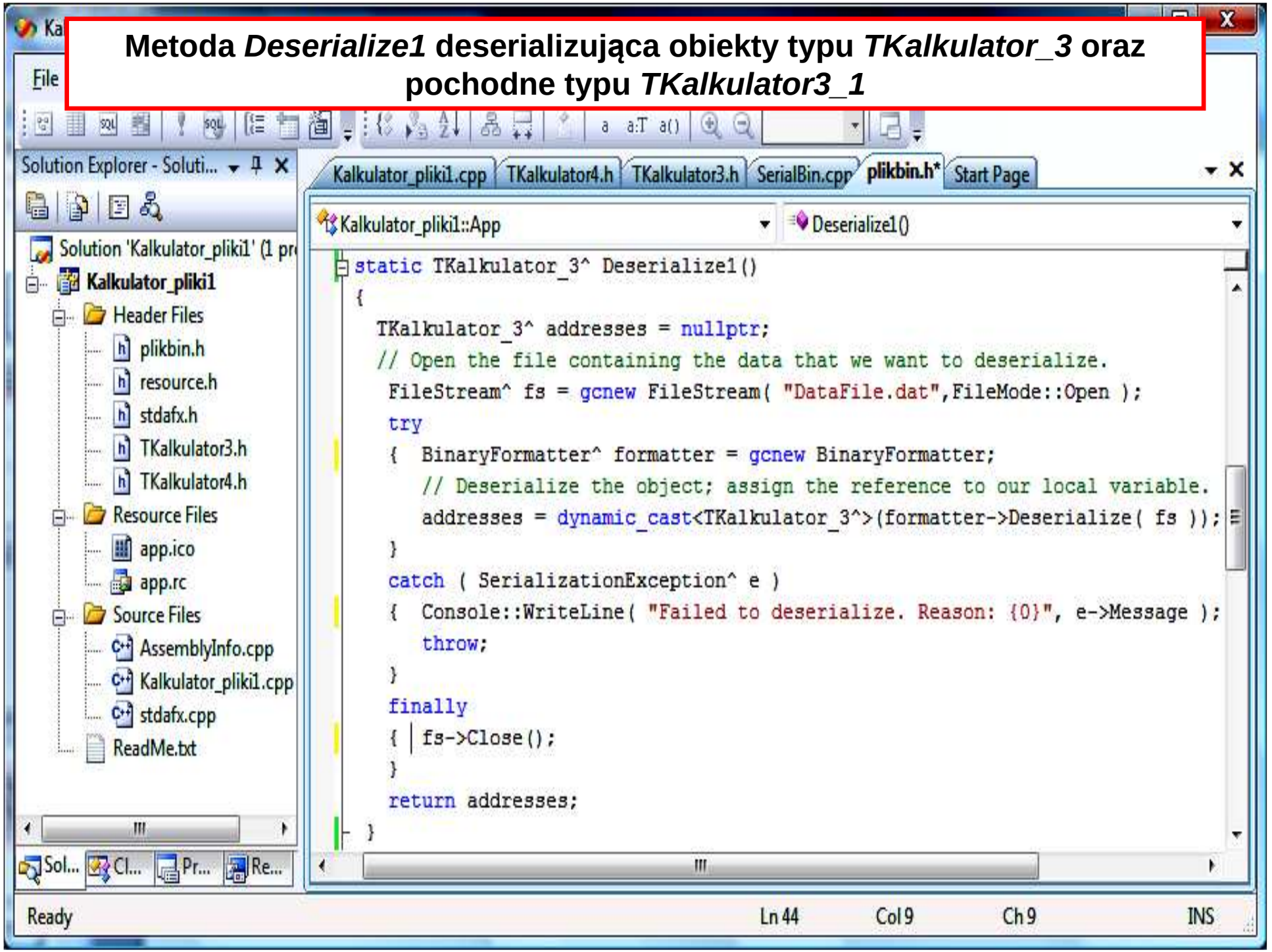




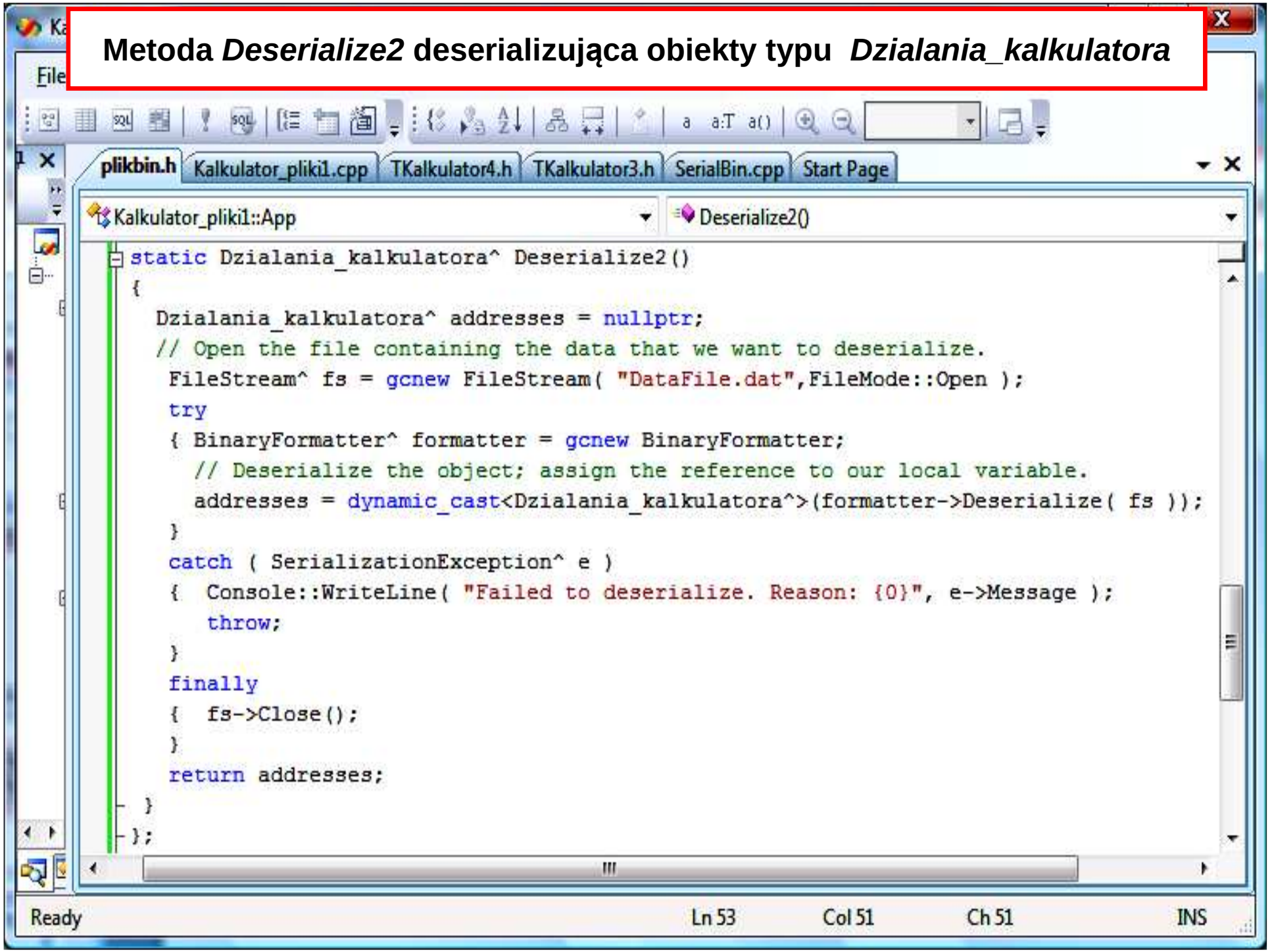
Przykład 3 – zapis kalkulatorów w plikach przez zastosowaniu strumieni binarnych za pomocą klasy App (metoda *Serialize* serializująca obiekty dowolnego typu z adnotacją [Serializable])

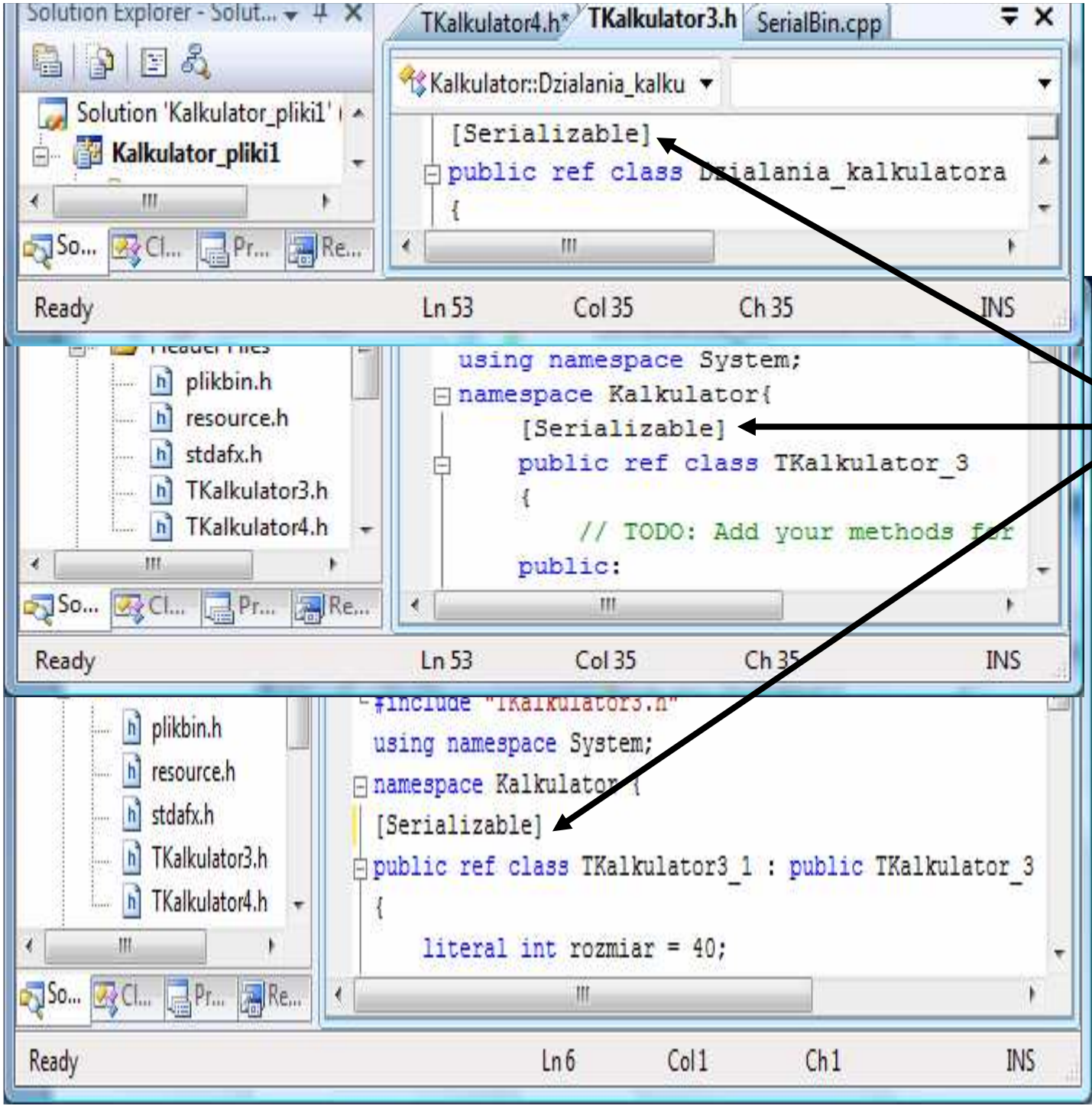


Metoda *Deserialize1* deserializująca obiekty typu *TKalkulator_3* oraz pochodne typu *TKalkulator3_1*

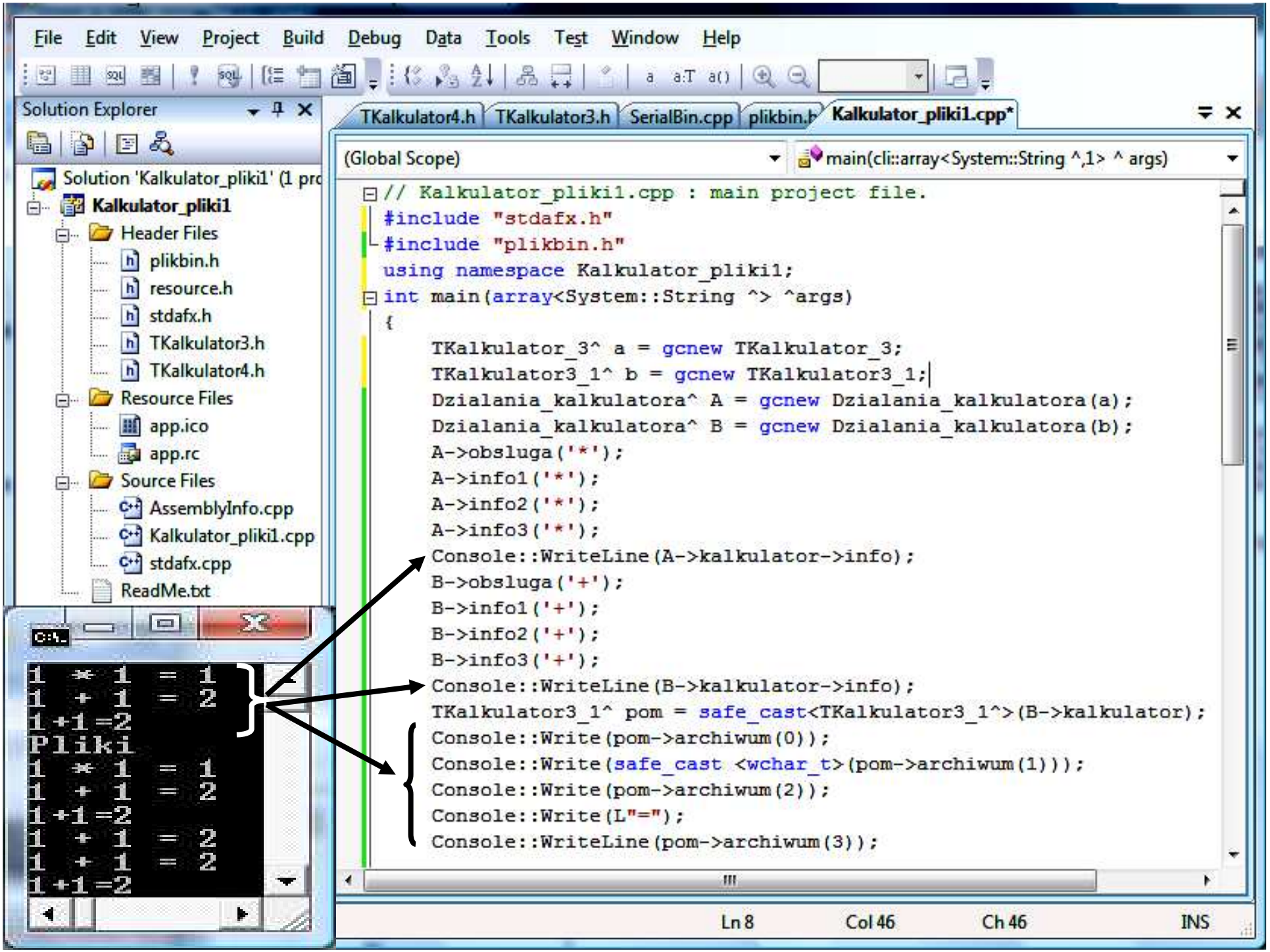


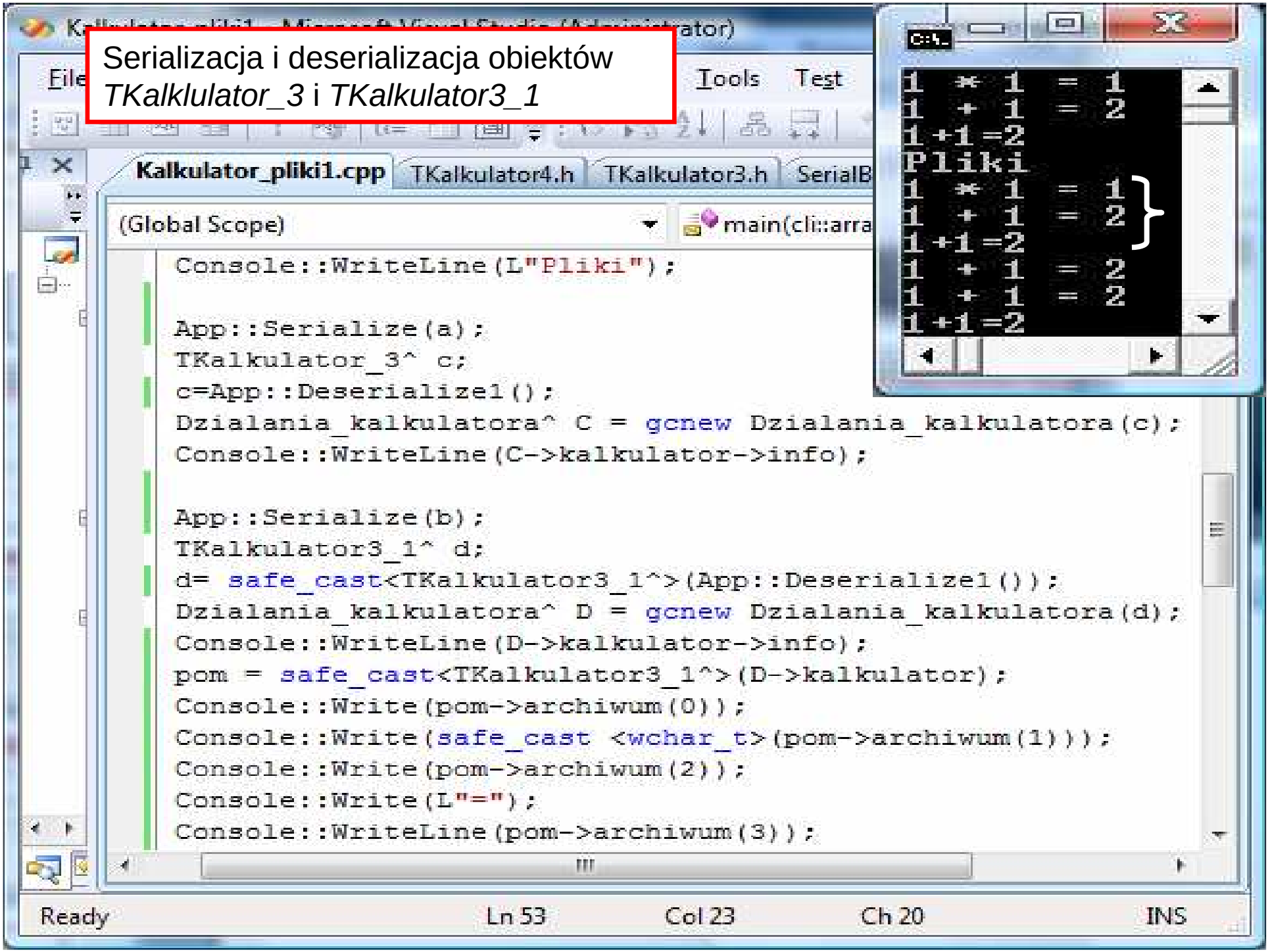
Metoda *Deserialize2* deserializująca obiekty typu *Dzialania_kalkulatora*



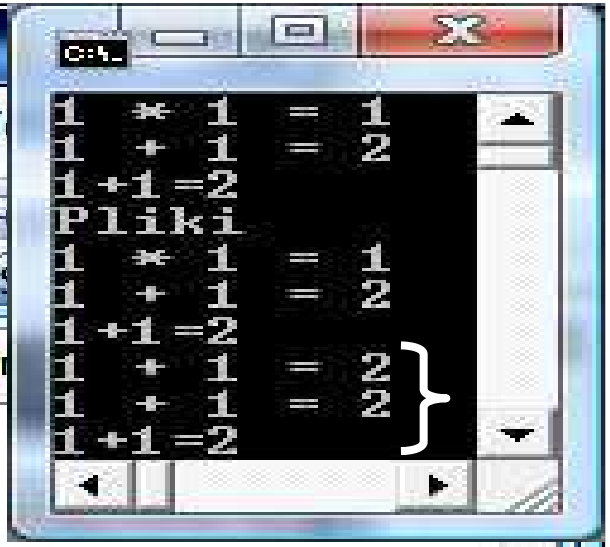


Adnotacje dołączone do definicji klas, niezbędne do serializacji i deserializacji obiektów klas TKalkulator_3 i TKalkulator3_1 oraz Dzialania_kalkulatora


File Edit View Project Build Debug Data Tools Test Window Help
Solution Explorer
Solution 'Kalkulator_pliki1' (1 pro
Kalkulator_pliki1
Header Files
plikbin.h
resource.h
stdafx.h
TKalkulator3.h
TKalkulator4.h
Resource Files
app.ico
app.rc
Source Files
AssemblyInfo.cpp
Kalkulator_pliki1.cpp
stdafx.cpp
ReadMe.txt
TKalkulator4.h
TKalkulator3.h
SerialBin.cpp
plikbin.h
Kalkulator_pliki1.cpp*
(Global Scope)
main(cli::array<System::String ^,1> ^ args)
// Kalkulator_pliki1.cpp : main project file.
#include "stdafx.h"
#include "plikbin.h"
using namespace Kalkulator_pliki1;
int main(array<System::String ^> ^args)
{
TKalkulator_3^ a = gcnew TKalkulator_3;
TKalkulator3_1^ b = gcnew TKalkulator3_1;
Dzialania_kalkulatora^ A = gcnew Dzialania_kalkulatora(a);
Dzialania_kalkulatora^ B = gcnew Dzialania_kalkulatora(b);
A->obsluga('*');
A->info1('*');
A->info2('*');
A->info3('*');
Console::WriteLine(A->kalkulator->info);
B->obsluga('+');
B->info1('+');
B->info2('+');
B->info3('+');
Console::WriteLine(B->kalkulator->info);
TKalkulator3_1^ pom = safe_cast<TKalkulator3_1^>(B->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));
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
1+1=2
Ln 8
Col 46
Ch 46
INS


Serializacja i deserializacja obiektów
TKalklulator_3 i *TKalkulator3_1*
Kalkulator_pliki1.cpp
TKalkulator4.h
TKalkulator3.h
(Global Scope)
main(cli::arra
Console::WriteLine(L"Pliki");
App::Serialize(a);
TKalkulator_3^ c;
c=App::Deserialize1();
Dzialania_kalkulatora^ C = gcnew Dzialania_kalkulatora(c);
Console::WriteLine(C->kalkulator->info);
App::Serialize(b);
TKalkulator3_1^ d;
d= safe_cast<TKalkulator3_1^>(App::Deserialize1());
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));
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
1 +1 =2
Ready
Ln 53
Col 23
Ch 20
INS

Serializacja i deserializacja obiektu typu *Dzialania_kalkulatora* z obiektem typuTKalklulator_3 oraz obiektu typu *Dzialania_kalkulatora* z obiektem typu*TKalkulator3_1*



(Global Scope) main(cli::ar

```
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<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));
return 0;
}
```

Ready Ln 64 Col 5 Ch 5