```
//////////////////////////////////////////////////////
// Kalkulator (prosty) - wersja agregowalna
import "unknwn.idl";

// Interface ICalc
[
   object,
   uuid(892753ED-D14E-4d2f-B812-041E0C01F5F3),
   helpstring("Kalkulator (prosty), wersja agregowalna"),
   pointer_default(unique)
]
interface ICalc : IUnknown
{
   HRESULT digit([in] int n);
   HRESULT comma();
   HRESULT sign();
   HRESULT op([in] int n);
   HRESULT eq();
   HRESULT c();
   HRESULT ce();
   [propget, id(1), helpstring("wyświetlacz")]
                                HRESULT display([out, retval] double *pVal);
   [propput, id(1), helpstring("wyświetlacz")]
                                        HRESULT display([in] double newVal);
};

// Biblioteka typów + clsid składników
[
   uuid(7FB7B169-2288-4713-98BD-4360E0147DD8),
   version(1.0),
   helpstring("Kalkulator (prosty), wersja agregowalna, bibl.typów")
]
library CalcAggrTypeLib
{
   importlib("stdole32.tlb") ;
   [
      uuid(4D3B9D9B-8DCC-4443-BAC8-1DCAE9955270),
      helpstring("Kalkulator (prosty), wersja agregowalna")
   ]
   coclass CalcSimple
   {
      [default] interface ICalc;
   };
} ;
```

```
// Serwer prostego kalkulatora - wersja agregowalna
// Copyright (c) 2001 by Jarosław Francik

#include "stdafx.h"
#include "calc2.h"      // plik automatycznie utworzony przez MIDL
#include "calc2_i.c" // plik automatycznie utworzony przez MIDL
#include <iostream.h>
#include "registry.h"   // pomocnicze narzędzia do rejestru systemowego...

//////////////////////////////////////////////////////
// Zmienne globalne

static HMODULE g_hModule = NULL ;   // uchwyt modułu DLL
static long g_cComponents = 0 ;     // Licznik aktywnych składników
static long g_cServerLocks = 0 ;    // Licznik dla LockServer

// Dane do rejestru systemowego
const char g_szProgID[] = "Calc.SimpleAggr.2";
const char g_szFriendlyName[] = "Prosty kalkulator COM - wersja agregowalna";
const char g_szVerIndProgID[] = "Calc.SimpleAggr";

//////////////////////////////////////////////////////
// Interfejs INondelegatingUnknown

interface INondelegatingUnknown
{  virtual HRESULT __stdcall
      NondelegatingQueryInterface(const IID&, void**) = 0 ;
   virtual ULONG __stdcall NondelegatingAddRef() = 0 ;
   virtual ULONG __stdcall NondelegatingRelease() = 0 ;
} ;

//////////////////////////////////////////////////////
// Klasa składnika

class CCalc : public ICalc, INondelegatingUnknown
{public:
   // Implementacja interfejsu IUnknown
   virtual HRESULT _stdcall QueryInterface(REFIID iid, void **ppv);
   virtual ULONG _stdcall AddRef();
   virtual ULONG _stdcall Release();

   // Implementacja interfejsu INondelegatingUnknown
   virtual HRESULT _stdcall NondelegatingQueryInterface(REFIID iid, void **ppv);
   virtual ULONG _stdcall NondelegatingAddRef();
   virtual ULONG _stdcall NondelegatingRelease();

   // Implementacja interfejsu ICalc
   virtual HRESULT __stdcall digit(int n);
   virtual HRESULT __stdcall comma();
   virtual HRESULT __stdcall sign();
   virtual HRESULT __stdcall op(int n);
   virtual HRESULT __stdcall eq();
   virtual HRESULT __stdcall c();
   virtual HRESULT __stdcall ce();
   virtual HRESULT __stdcall get_display(double __RPC_FAR *pVal);
   virtual HRESULT __stdcall put_display(double newVal);

   // Implementacja wewnętrzna
   double m_display, m_reg;   // wyświetlacz i drugi rejestr
   int m_op;                  // kod operacji; 0=nic, 1-4 = + - * /
   bool m_bAppendDisp;        // tryb dodawania do wyświetlacza;
public:
   CCalc(IUnknown *pUnknownOuter) : m_nRef(1), m_display(0.0),
                                    m_reg(0.0), m_op(0), m_bAppendDisp(false)
   {  InterlockedIncrement(&g_cComponents);
      if (pUnknownOuter)
         m_pUnknownOuter = pUnknownOuter; // zostaliśmy zagregowani!
      else
         m_pUnknownOuter = (IUnknown*)(INondelegatingUnknown*)this;
   }
   ~CCalc() { InterlockedDecrement(&g_cComponents); }
private:
   long m_nRef;
   IUnknown *m_pUnknownOuter; // obiekt do którego delegujemy odwołania
};
```

```
//////////////////////////////////////////////////////////
//
// Implementacja interfejsu ICalc

HRESULT __stdcall CCalc::digit(int n)
{  if (!m_bAppendDisp)
      m_display = n;
   else
      m_display = m_display * 10 + n;
   m_bAppendDisp = true;
   return S_OK;
}
HRESULT __stdcall CCalc::comma()
{  return E_NOTIMPL;
}
HRESULT __stdcall CCalc::sign()
{  m_display = -m_display;
   return S_OK;
}
HRESULT __stdcall CCalc::op(int n)
{  eq();
   m_op = n;
   return S_OK;
}
HRESULT __stdcall CCalc::eq()
{  switch (m_op)
   {
      case 1: m_display = m_reg + m_display; break;
      case 2: m_display = m_reg - m_display; break;
      case 3: m_display = m_reg * m_display; break;
      case 4: m_display = m_reg / m_display; break;
   }
   m_reg = m_display;
   m_bAppendDisp = false;
   return S_OK;
}
HRESULT __stdcall CCalc::c()
{  m_reg = m_display = m_op = 0;
   m_bAppendDisp = false;
   return S_OK;
}
HRESULT __stdcall CCalc::ce()
{  m_display = 0;
   m_bAppendDisp = false;
   return S_OK;
}
HRESULT __stdcall CCalc::get_display(double __RPC_FAR *pVal)
{
   *pVal = m_display;
   return S_OK;
}
HRESULT __stdcall CCalc::put_display(double newVal)
{
   m_display = newVal;
   return S_OK;
}


//////////////////////////////////////////////////////////
```

```
//////////////////////////////////////////////////////////
//
// Implementacja interfejsu IUnknown

HRESULT _stdcall CCalc::QueryInterface(REFIID iid, void **ppv)
{
   return m_pUnknownOuter->QueryInterface(iid, ppv);
}

ULONG _stdcall CCalc::AddRef()
{
   return m_pUnknownOuter->AddRef();
}

ULONG _stdcall CCalc::Release()
{
   return m_pUnknownOuter->Release();
}

//////////////////////////////////////////////////////////
//
// Implementacja interfejsu INondelegatingUnknown

HRESULT _stdcall CCalc::NondelegatingQueryInterface(REFIID iid, void **ppv)
{
   if (iid == IID_IUnknown)
      *ppv = (INondelegatingUnknown*)this;
   else if (iid == IID_ICalc)
      *ppv = (ICalc*)this;
   else
   {  ppv = NULL;
      return E_NOINTERFACE;
   }
   ((IUnknown*)(*ppv))->AddRef();
   return S_OK;
}

ULONG _stdcall CCalc::NondelegatingAddRef()
{
   return InterlockedIncrement(&m_nRef);
}

ULONG _stdcall CCalc::NondelegatingRelease()
{
   if (InterlockedDecrement(&m_nRef) == 0)
   {
        delete this;
       return 0;
   }
   return m_nRef;
}
```

```
////////////////////////////////////////////////////////
//
// Class Factory

class CFactory : public IClassFactory
{
   // standardowa deklaracja interfejsu IClassFactory
} ;

HRESULT __stdcall CFactory::QueryInterface(const IID& iid, void** ppv)
// standardowa implementacja

ULONG __stdcall CFactory::AddRef()
// standardowa implementacja

ULONG __stdcall CFactory::Release()
// standardowa implementacja

HRESULT __stdcall CFactory::CreateInstance(IUnknown* pUnknownOuter,
                                           const IID& iid, void** ppv)
{
   if (pUnknownOuter != NULL && iid != IID_IUnknown)
      return CLASS_E_NOAGGREGATION;

   // Utworzenie składnika
   CCalc* pCalc = new CCalc(pUnknownOuter);
   if (!pCalc)
      return E_OUTOFMEMORY;

   // Utworzenie żądanego interfejsu
   HRESULT hr = pCalc->NondelegatingQueryInterface(iid, ppv);

   pCalc->NondelegatingRelease();

   return hr;
}

HRESULT __stdcall CFactory::LockServer(BOOL bLock)
// standardowa implementacja


////////////////////////////////////////////////////////
//
// Funkcje eksportowane!

// standardowa implementacja
```

```
////////////////////////////////////////////////////////
// Kalkulator “naukowy” (z wykorzystaniem agregacji)
import "unknwn.idl";
// Interface ICalcScientific
[
   object,
   uuid(22E6B303-CEFE-4f65-9BED-8623AF83903E),
   helpstring("Kalkulator naukowy"),
   pointer_default(unique)
]
interface ICalcScientific : IUnknown
{
   HRESULT sqroot();
};

// Biblioteka typów + clsid składników
[
   uuid(F3B7AD02-3D70-46e5-9313-D479B95E4089),
   version(1.0),
   helpstring("Kalkulator naukowy, biblioteka typów")
]
library CalcSciTypeLib
{
   importlib("stdole32.tlb") ;

   [
      uuid(BA847794-18E2-42ce-B053-754828A6388E),
      helpstring("Kalkulator naukowy - komponent zagregowany")
   ]
   coclass CalcScientific
   {
      [default] interface ICalcScientific;
      // interface ICalc;
   };
} ;
```

```
// Serwer kalkulatora naukowego (z pierwiastkowaniem)
// Moduł agreguje składnik Calc2 - prosty kalkulator
// Copyright (c) 2001 by Jarosław Francik

#include "stdafx.h"
#include "calcsci.h" // plik automatycznie utworzony przez MIDL
#include "calcsci_i.c"  // plik automatycznie utworzony przez MIDL
#include <iostream.h>
#include "registry.h"   // pomocnicze narzędzia do rejestru systemowego...
#include <math.h>

#include "..\\calc2\\calc2.h"
#include "..\\calc2\\calc2_i.c"

////////////////////////////////////////////////////////
//
// Zmienne globalne

// jak w poprzednim przykładzie...
```

```
////////////////////////////////////////////////////////
//
// Klasa składnika

class CCalcScientific : public ICalcScientific
{
public:
   // Implementacja interfejsu IUnknown
   virtual HRESULT _stdcall QueryInterface(REFIID iid, void **ppv);
   virtual ULONG _stdcall AddRef();
   virtual ULONG _stdcall Release();

   // Implementacja interfejsu ICalcScientific
   virtual HRESULT __stdcall sqroot();

   // Funkcja inicjalizująca obiekt wewnętrzny
   // (wywoływana przez fabrykę klas)
   HRESULT Init()
   {
      return CoCreateInstance(CLSID_CalcSimple, (IUnknown*)this,
                CLSCTX_INPROC_SERVER, IID_IUnknown, (void**)&m_pUnknownInner);
   }

public:
   IUnknown *m_pUnknownInner;
   CCalcScientific() : m_nRef(1)    { InterlockedIncrement(&g_cComponents); }
   ~CCalcScientific()               { InterlockedDecrement(&g_cComponents); }
private:
   long m_nRef;
};

////////////////////////////////////////////////////////
//
// Implementacja interfejsu ICalcScientific

HRESULT __stdcall CCalcScientific::sqroot()
{
   ICalc *pCalc = NULL;
   HRESULT hr = m_pUnknownInner->QueryInterface(IID_ICalc, (void**)&pCalc);
   if (SUCCEEDED(hr))
   {
      double dDisp;
      pCalc->get_display(&dDisp);
      dDisp = sqrt(dDisp);
      pCalc->put_display(dDisp);
      return S_OK;
   }
   return hr;
}
```

```
////////////////////////////////////////////////////////
//
// Implementacja interfejsu IUnknown

HRESULT _stdcall CCalcScientific::QueryInterface(REFIID iid, void **ppv)
{
   if (iid == IID_IUnknown)
      *ppv = (ICalcScientific*)this;
   else if (iid == IID_ICalcScientific)
      *ppv = (ICalcScientific*)this;
   else if (iid == IID_ICalc)
      return m_pUnknownInner->QueryInterface(iid, ppv);
   else
   {  ppv = NULL;
      return E_NOINTERFACE;
   }
   ((IUnknown*)(*ppv))->AddRef();
   return S_OK;
}

ULONG _stdcall CCalcScientific::AddRef()
ULONG _stdcall CCalcScientific::Release()
// standardowe implementacje

////////////////////////////////////////////////////////
//
// Class Factory

// standardowa implementacja fabryki klas (jak w pierwszym przykładzie)
// zmiany dotyczą tylko poniższego:
HRESULT __stdcall CFactory::CreateInstance(IUnknown* pUnknownOuter,
                                           const IID& iid, void** ppv)
{
   if (pUnknownOuter != NULL)
      return CLASS_E_NOAGGREGATION;

   // Utworzenie składnika
   CCalcScientific* pCalcScientific = new CCalcScientific;
   if (!pCalcScientific)
      return E_OUTOFMEMORY;

   // Utworzenie wewnętrznego składnika
   HRESULT hr = pCalcScientific->Init();
   if (!SUCCEEDED(hr))
      return CLASS_E_NOAGGREGATION;

   // Utworzenie żądanego interfejsu
   hr = pCalcScientific->QueryInterface(iid, ppv);

   pCalcScientific->Release();

   return hr;
}
////////////////////////////////////////////////////////
//
// Funkcje eksportowane!

// standardowa implementacja
```

```
//////////////////////////////////////////////////////
// Kalkulator (prosty) – wersja z punktami połączeń
import "unknwn.idl";

// Interface ICalc
[
   object,
   uuid(14169CFC-8CC3-45ec-8C46-D080C8DE6D39),
   helpstring("Kalkulator (prosty), wersja z punktami połączenia"),
   pointer_default(unique)
]
interface ICalc : IUnknown
{
   HRESULT digit([in] int n);
   HRESULT comma();
   HRESULT sign();
   HRESULT op([in] int n);
   HRESULT eq();
   HRESULT c();
   HRESULT ce();
   [propget, id(1), helpstring("wyświetlacz")]
                                HRESULT display([out, retval] double *pVal);
   [propput, id(1), helpstring("wyświetlacz")]
                                        HRESULT display([in] double newVal);
};

// Interfejs wychodzący
[
   object,
   uuid(A1284ED8-990E-4885-901A-26DA646245C8),
   helpstring("Kalkulator - Interfejs wychodzący"),
   pointer_default(unique)
]
interface IOutgoingCalc : IUnknown
{
   HRESULT changed([in] double val);
   HRESULT errorB([in] BSTR msg);
};

// Biblioteka typów + clsid składników
[
   uuid(AF933814-60B5-4017-A695-413D21340D47),
   version(1.0),
   helpstring("Kalkulator (prosty), wersja z punktami połączenia, biblioteka
typów")
]
library CalcAggrTypeLib
{
   importlib("stdole32.tlb") ;
   [
      uuid(0D49456B-4A4B-423c-9804-762059B7299A),
      helpstring("Kalkulator (prosty), wersja z punktami połączenia")
   ]
   coclass CalcSimple
   {
      [default] interface ICalc;
      [source] interface IOutgoingCalc;
   };
} ;
```

```
// Serwer prostego kalkulatora - wersja z punktami połączenia
// Copyright (c) 2001 by Jarosław Francik

#include "stdafx.h"
#include "calcCP.h"     // plik automatycznie utworzony przez MIDL
#include "calcCP_i.c"   // plik automatycznie utworzony przez MIDL
#include <iostream.h>
#include "registry.h"   // pomocnicze narzędzia do rejestru systemowego...

#include <ocidl.h>      // IConnectionPoint, IConnectionPointContainer

//////////////////////////////////////////////////////
//
// Zmienne globalne

static HMODULE g_hModule = NULL ;   // uchwyt modułu DLL
static long g_cComponents = 0 ;     // Licznik aktywnych składników
static long g_cServerLocks = 0 ;    // Licznik dla LockServer

// Dane do rejestru systemowego
const char g_szProgID[] = "Calc.SimpleCP.2";
const char g_szFriendlyName[] = "Prosty kalkulator - wersja z CP”;
const char g_szVerIndProgID[] = "Calc.SimpleCP";

// Wskaźnik do interfejsu ujścia
IOutgoingCalc *g_pOutgoingCalc;

//////////////////////////////////////////////////////
//
// Interfejs INondelegatingUnknown

// jak w pierwszym przykładzie

//////////////////////////////////////////////////////
//
// Klasa składnika

class CCalc : public ICalc, INondelegatingUnknown,
                                        IConnectionPoint, IConnectionPointContainer
{
public:
   // Implementacja interfejsu IUnknown
   virtual HRESULT _stdcall QueryInterface(REFIID iid, void **ppv);
   virtual ULONG _stdcall AddRef();
   virtual ULONG _stdcall Release();

   // Implementacja interfejsu INondelegatingUnknown
   virtual HRESULT _stdcall NondelegatingQueryInterface(REFIID iid, void **ppv);
   virtual ULONG _stdcall NondelegatingAddRef();
   virtual ULONG _stdcall NondelegatingRelease();
```

```
   // Implementacja interfejsu IConnectionPoint
   virtual HRESULT _stdcall GetConnectionInterface(IID __RPC_FAR *pIID);
   virtual HRESULT _stdcall GetConnectionPointContainer(IConnectionPointContainer
                                                 __RPC_FAR *__RPC_FAR *ppCPC);
   virtual HRESULT _stdcall Advise(IUnknown __RPC_FAR *pUnkSink,
                                   DWORD __RPC_FAR *pdwCookie);
   virtual HRESULT _stdcall Unadvise(DWORD dwCookie);
   virtual HRESULT _stdcall EnumConnections(IEnumConnections
                                                 __RPC_FAR *__RPC_FAR *ppEnum);

   // Implementacja interfejsu IConnectionPointContainer
   virtual HRESULT _stdcall EnumConnectionPoints(IEnumConnectionPoints
                                                 __RPC_FAR *__RPC_FAR *ppEnum);
   virtual HRESULT _stdcall FindConnectionPoint(REFIID riid,
                               IConnectionPoint __RPC_FAR *__RPC_FAR *ppCP);

   // dalej jak w pierwszym przykładzie
   ....
};

//////////////////////////////////////////////////////
//
// Implementacja interfejsu Icalc

// jak w pierwszym przykładzie

//////////////////////////////////////////////////////
//
// Implementacja interfejsu IUnknown

// jak w pierwszym przykładzie

//////////////////////////////////////////////////////
//
// Implementacja interfejsu INondelegatingUnknown

HRESULT _stdcall CCalc::NondelegatingQueryInterface(REFIID iid, void **ppv)
{  if (iid == IID_IUnknown)
      *ppv = (INondelegatingUnknown*)this;
   else if (iid == IID_ICalc)
      *ppv = (ICalc*)this;
   else if (iid == IID_IConnectionPoint)
      *ppv = (IConnectionPoint*)this;
   else if (iid == IID_IConnectionPointContainer)
      *ppv = (IConnectionPointContainer*)this;
   else
   {  ppv = NULL;
      return E_NOINTERFACE;
   }
   ((IUnknown*)(*ppv))->AddRef();
   return S_OK;
}

ULONG _stdcall CCalc::NondelegatingAddRef()
// standardowa implementacja

ULONG _stdcall CCalc::NondelegatingRelease()
// standardowa implementacja
```

```
//////////////////////////////////////////////////////
//
// Implementacja interfejsu IConnectionPoint

HRESULT _stdcall CCalc::GetConnectionInterface(IID __RPC_FAR *pIID)
{  return E_NOTIMPL;
}

HRESULT _stdcall CCalc::GetConnectionPointContainer(IConnectionPointContainer
__RPC_FAR *__RPC_FAR *ppCPC)
{  return E_NOTIMPL;
}

HRESULT _stdcall CCalc::Advise(IUnknown __RPC_FAR *pUnkSink, DWORD __RPC_FAR
*pdwCookie)
{
   // Wartość cookie jest nieistotna przy jednym połączeniu
   *pdwCookie = 1;

   // oto wskaźnik do interfejsu ujścia
   return pUnkSink->QueryInterface(IID_IOutgoingCalc,
                                                   (void**)&g_pOutgoingCalc);
}

HRESULT _stdcall CCalc::Unadvise(DWORD dwCookie)
{
   g_pOutgoingCalc->Release();
   g_pOutgoingCalc = NULL;
   return S_OK;
}

HRESULT _stdcall CCalc::EnumConnections(IEnumConnections __RPC_FAR
                                                    *__RPC_FAR *ppEnum)
{  return E_NOTIMPL;
}

//////////////////////////////////////////////////////
//
// Implementacja interfejsu IConnectionPointContainer

HRESULT _stdcall CCalc::EnumConnectionPoints(IEnumConnectionPoints __RPC_FAR
*__RPC_FAR *ppEnum)
{  return E_NOTIMPL;
}

HRESULT _stdcall CCalc::FindConnectionPoint(REFIID riid, IConnectionPoint
__RPC_FAR *__RPC_FAR *ppCP)
{
   if (riid == IID_IOutgoingCalc)
      return QueryInterface(IID_IConnectionPoint, (void**)ppCP);
   return E_NOINTERFACE;
}

//////////////////////////////////////////////////////
//
// Class Factory I funkcje eksportowane

// standardowa implementacja fabryki klas (jak w pierwszym przykładzie)
```