Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Институт информационных технологий

Факультет повышения квалификации и переподготовки

Кафедра МПСС

**ОТЧЕТ**

По лабораторной работе №1-2 и самостоятельной работе

по дисциплине ТКП

Тема: «Разработка простого сервера и клиента СОМ средствами С++»

Минск 2021

**Цель работы:** Изучить технологию создания простого компонента и его использования. Создать динамически загружаемую библиотеку, реализующую сервер c интерфейсом IMath и зарегистрировать сервер в реестре и создать клиентское приложение.

**Задание.**

**Общая часть**

1. Изучить технологию создания простого компонента и его использования .

2. Создать динамически загружаемую библиотеку, реализующую сервер c

интерфейсом IMath в соответствии с описанием к работе (файл . " Simple COM

Server\_&\_Client\_1.htm") и зарегистрировать сервер в реестре (как указано в описании).

3. Создать клиентское приложение (файл . "Simple COM Server\_&\_Client\_2.htm"). ,

демонстрирующее использование сервера СОМ . Выполнить приложение, убедиться в корректности взаимодействия с сервером.

4. Переместить библиотеку сервера в другой каталог. Внести необходимые изменения в реестр. Запустить приложение клиента и убедиться, что оно взаимодействует с сервером.

**Индивидуальное задание**

5. Создать сервер, реализующий интерфейс IFN\_nn, где nn - номер варианта. Создать

клиентское приложение, вызывающее методы интерфейса IFN\_nn .

6. Добавить к серверу реализацию второго интерфейса - IVer c единственным методом которого будет HRESULT GetAutor(wchar\_t \*\* author) , возвращающий фамилию и

группу автора сервера.

7. Добавить в библиотеку сервера функции регистрации и отрегистрации сервера.

8. Оформить отчет о работе.

**Ход работы.**

**1 Изучение технологии создания простого компонента и его использование**

Описание примера

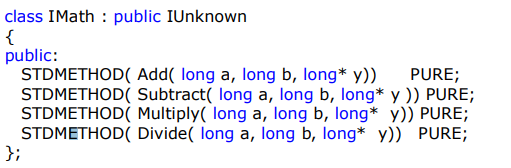
long у = (long) a + (long) b; // метод Add

long y = (long) a - (long) b; // метод Subtract

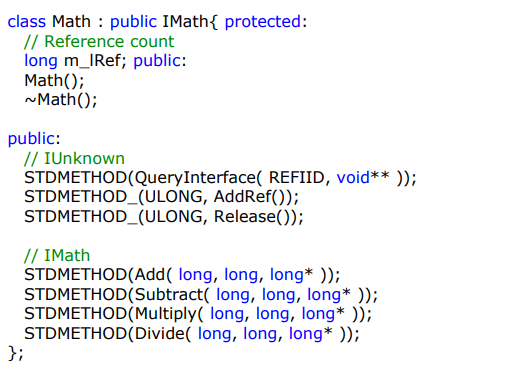
long y = (long) a \* (long) b; // метод Multiply

long y = (long) a / (long) b; // метод Divide

Определение интерфейса



Класс реализации компонента

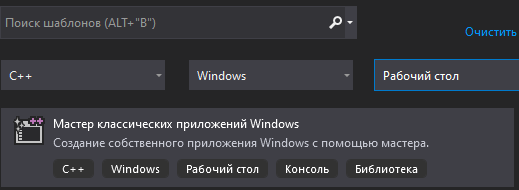


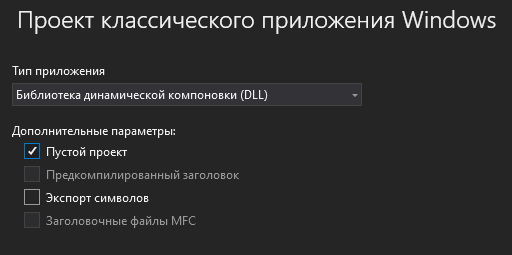
Типы серверов COM

1. Внутрипроцессный IProc
2. Локальный
3. Удаленный

Для клиента это совершенно не имеет значения.

Создание проекта динамически компануемой библиотеки

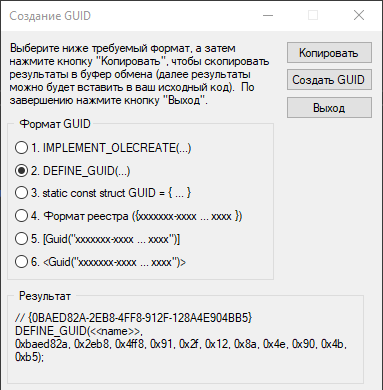




Далее объявим абстрактный класс интерфейса и класс компонента, их IID и CLSID. Чтобы обеспечить возможность предоставить программе клиента информацию об определениях интерфейса и идентификаторов CLSID и IID, отделим их от собственно текста программы и поместим в отдельный файл.

Добавим заголовочный файл, в котором будет определен тип интерфейса c уникальным именем, которое формируется специальным образом. GUID (Globally Unique Identifier) формир. из мак-адреса сетевой карты и отметки времени + псевдослучайное значение.

Далее создадим GUID (средства - создать GUID). Используем форму DEFINE\_GUID



|  |
| --- |
| #pragma once  #include <Windows.h> //для наших макро DEFINE\_GUID, STDMETHOD...  //  // imath.h  //  //Идентификатор класса компонента  // {0BAED82A-2EB8-4FF8-912F-128A4E904BB5}  DEFINE\_GUID(CLSID\_MATH ,  0xbaed82a, 0x2eb8, 0x4ff8, 0x91, 0x2f, 0x12, 0x8a, 0x4e, 0x90, 0x4b, 0xb5);  //Идентификатор интерфейса  // {56D7AF4F-C48A-4190-8880-19962537A9AE}  DEFINE\_GUID(IID\_IMATH ,  0x56d7af4f, 0xc48a, 0x4190, 0x88, 0x80, 0x19, 0x96, 0x25, 0x37, 0xa9, 0xae);  class IMath : public IUnknown  {  public:  STDMETHOD( Add( long, long, long\* )) PURE;  STDMETHOD( Subtract( long, long, long\* )) PURE;  STDMETHOD( Multiply( long, long, long\* )) PURE;  STDMETHOD( Divide( long, long, long\* )) PURE;  }; |

Далее необходимо объявить класс компонента и фабрику классов для него.

|  |
| --- |
| #pragma once  //  // сmath.h  //  #include "IMath.h"  extern long g\_lObjs; //счетчик компонентов, созданных на базе сервера; память для него будет выделена в другом файле  extern long g\_lLocks; //счетчик блокировок  class СMath : public IMath  {  protected:  // Reference count  long m\_lRef;  public:  СMath();  ~СMath();  public:  // IUnknown  STDMETHOD(QueryInterface(REFIID, void\*\*));  STDMETHOD\_(ULONG, AddRef());  STDMETHOD\_(ULONG, Release());  // IMath  STDMETHOD(Add(long, long, long\*));  STDMETHOD(Subtract(long, long, long\*));  STDMETHOD(Multiply(long, long, long\*));  STDMETHOD(Divide(long, long, long\*));  };  class MathClassFactory : public IClassFactory  {  protected:  long m\_lRef;  public:  MathClassFactory();  ~MathClassFactory();  // IUnknown  STDMETHOD(QueryInterface(REFIID, void\*\*));  STDMETHOD\_(ULONG, AddRef());  STDMETHOD\_(ULONG, Release());  // IClassFactory  STDMETHOD(CreateInstance(LPUNKNOWN, REFIID, void\*\*));  STDMETHOD(LockServer(BOOL));  }; |

Класс СMath является производным по отношению к классу интерфейсов IMath, который в свою очередь является производным для IUnknown. Объявим методы IUnknown и IMath. Отслеживание общего количества экземпляров компонента в DLL-файле и количества вызовов IClassFactory::LockServer возлагается на две глобальные переменные. Затем объявляем класс для фабрики классов компонента Math.

|  |
| --- |
| //  // СMath.cpp  //  #include <windows.h>  #include "CMath.h"  //  // Math class implementation  //  // Constructors  CMath::CMath()  {  m\_lRef = 0;  // Увеличить значение внешнего счетчика объектов  InterlockedIncrement(&g\_lObjs);  }  // The destructor  CMath::~CMath()  {  // Уменьшить значение внешнего счетчика объектов  InterlockedDecrement(&g\_lObjs);  }  STDMETHODIMP CMath::QueryInterface(REFIID riid, void\*\* ppv)  {  \*ppv = 0;  if (riid == IID\_IUnknown || riid == IID\_IMATH)  \*ppv = this;  if (\*ppv)  {  AddRef();  return(S\_OK);  }  return (E\_NOINTERFACE);  }  STDMETHODIMP\_(ULONG) CMath::AddRef()  {  return InterlockedIncrement(&m\_lRef);  }  STDMETHODIMP\_(ULONG) CMath::Release()  {  if (InterlockedDecrement(&m\_lRef) == 0)  {  delete this;  return 0;  }  return m\_lRef;  }  STDMETHODIMP CMath::Add(long lOp1, long lOp2, long\* pResult)  {  \*pResult = lOp1 + lOp2;  return S\_OK;  }  STDMETHODIMP CMath::Subtract(long lOp1, long lOp2, long\*  pResult)  {  \*pResult = lOp1 - lOp2;  return S\_OK;  }  STDMETHODIMP CMath::Multiply(long lOp1, long lOp2, long\*  pResult)  {  \*pResult = lOp1 \* lOp2;  return S\_OK;  }  STDMETHODIMP CMath::Divide(long lOp1, long lOp2, long\* pResult  )  {  \*pResult = lOp1 / lOp2;  return S\_OK;  }  //Фабрика класса  MathClassFactory::MathClassFactory()  {  m\_lRef = 0;  }  MathClassFactory::~MathClassFactory()  {  }  STDMETHODIMP MathClassFactory::QueryInterface(REFIID riid,  void\*\* ppv)  {  \*ppv = 0;  if (riid == IID\_IUnknown || riid == IID\_IClassFactory)  \*ppv = this;  if (\*ppv)  {  AddRef();  return S\_OK;  }  return(E\_NOINTERFACE);  }  STDMETHODIMP\_(ULONG) MathClassFactory::AddRef()  {  return InterlockedIncrement(&m\_lRef);  }  STDMETHODIMP\_(ULONG) MathClassFactory::Release()  {  if (InterlockedDecrement(&m\_lRef) == 0)  {  delete this;  return 0;  }  return m\_lRef;  }  STDMETHODIMP MathClassFactory::CreateInstance  (LPUNKNOWN pUnkOuter, REFIID riid, void\*\* ppvObj)  {  CMath\* pMath;  HRESULT hr;  \*ppvObj = 0;  pMath = new CMath;  if (pMath == 0)  return(E\_OUTOFMEMORY);  hr = pMath->QueryInterface(riid, ppvObj);  if (FAILED(hr))  delete pMath;  return hr;  }  STDMETHODIMP MathClassFactory::LockServer(BOOL fLock)  {  if (fLock)  InterlockedIncrement(&g\_lLocks);  else  InterlockedDecrement(&g\_lLocks);  return S\_OK;  } |

Главная программа реализации хранилища компонента.

|  |
| --- |
| //  // server.cpp : Defines the initialization routines for the DLL.  //  #include <windows.h>  #include <initguid.h>  #include "CMath.h"  long g\_lObjs = 0;  long g\_lLocks = 0;  STDAPI DllGetClassObject(REFCLSID rclsid, REFIID riid, void\*\*  ppv)  {  HRESULT hr;  MathClassFactory\* pCF;  pCF = 0;  // Make sure the CLSID is for our Expression component  if (rclsid != CLSID\_MATH)  return(E\_FAIL);  pCF = new MathClassFactory;  if (pCF == 0)  return(E\_OUTOFMEMORY);  hr = pCF->QueryInterface(riid, ppv);  // Check for failure of QueryInterface  if (FAILED(hr))  {  delete pCF;  pCF = 0;  }  return hr;  }  STDAPI DllCanUnloadNow(void)  {  if (g\_lObjs || g\_lLocks)  return(S\_FALSE);  else  return(S\_OK);  } |

Регистрация в реестре

Ключ класса компонента Math\_CLSID32.reg

|  |
| --- |
| Windows Registry Editor Version 5.00  [HKEY\_LOCAL\_MACHINE\SOFTWARE\WOW6432Node\Classes\CLSID\{0BAED82A-2EB8-4FF8-912F-128A4E904BB5}]  @="TCP\_Lab1 Math Component 00321"  [HKEY\_LOCAL\_MACHINE\SOFTWARE\WOW6432Node\Classes\CLSID\{0BAED82A-2EB8-4FF8-912F-128A4E904BB5}\InprocServer32]  @="F:\\!bguir 4 tr\\TKP\\labs1-2dkotsuba\\labs1-2\_00321\_dkotsuba\_prj\\Server00321Sol\\Debug\\Server00321Sol.dll"  "ThreadingModel"="Apartment"  [HKEY\_LOCAL\_MACHINE\SOFTWARE\WOW6432Node\Classes\CLSID\{0BAED82A-2EB8-4FF8-912F-128A4E904BB5}\ProgID]  @="Math.Component00321.1"  [HKEY\_LOCAL\_MACHINE\SOFTWARE\WOW6432Node\Classes\CLSID\{0BAED82A-2EB8-4FF8-912F-128A4E904BB5}\VersionIndependentProgID]  @="Math.Component00321" |

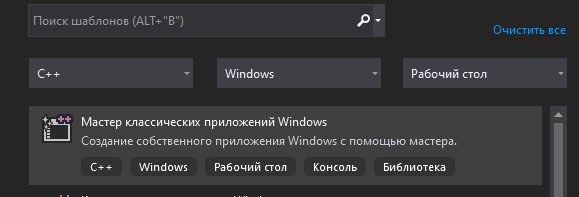
Math\_ProgId.reg

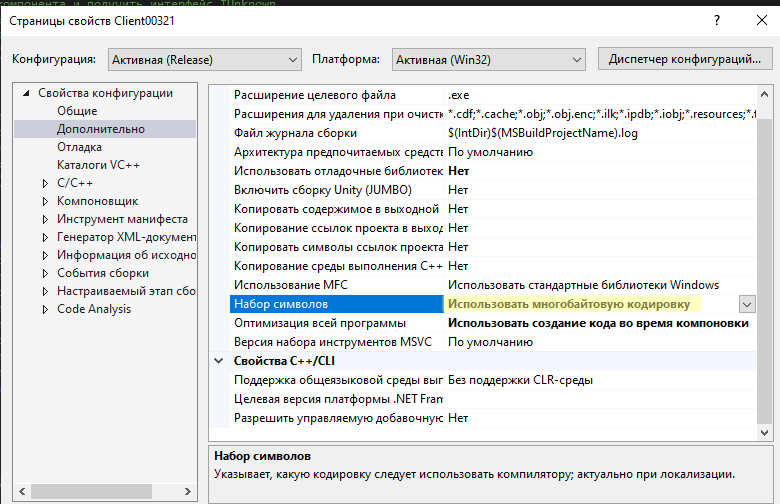
|  |
| --- |
| Windows Registry Editor Version 5.00  [HKEY\_CLASSES\_ROOT\Math.Component00321.1]  @="TCP\_Lab1 Math Component 00321"  [HKEY\_CLASSES\_ROOT\Math.Component00321.1\CLSID]  @="{0BAED82A-2EB8-4FF8-912F-128A4E904BB5}" |

Math\_ProgIdVerInd.reg

|  |
| --- |
| Windows Registry Editor Version 5.00  [HKEY\_CLASSES\_ROOT\Math.Component0321]  @="TCP\_Lab1 Math Component 00321"  [HKEY\_CLASSES\_ROOT\Math.Component00321\CLSID]  @="{0BAED82A-2EB8-4FF8-912F-128A4E904BB5}"  [HKEY\_CLASSES\_ROOT\Math.Component00321\CurVer]  @="Math.Component00321.1" |

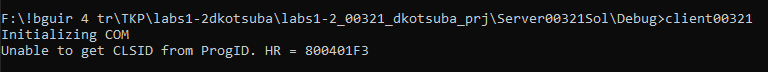
Создание клиента

Добавим к проекту приложение и назначим в качестве запускаемого. Шаблон тот же :

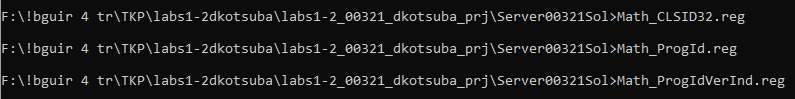


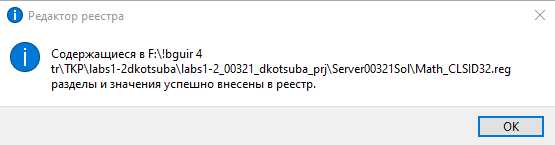
|  |
| --- |
| //  // Client.cpp  //  #include <windows.h>  #include <tchar.h>  #include <stdio.h>  #include <initguid.h>  #include "../Server00321/IMath.h"  int main(int argc, char\* argv[])  {  printf("Initializing COM \n");  if (FAILED(CoInitialize(NULL)))  {  printf("Unable to initialize COM \n"); return -1;  }  const char\* szProgID = "Math.Component00321.1";  WCHAR szWideProgID[128];  CLSID clsid;  long lLen = MultiByteToWideChar(CP\_ACP, 0, szProgID,  strlen(szProgID), szWideProgID, sizeof(szWideProgID));  szWideProgID[lLen] = '\0';  HRESULT hr = ::CLSIDFromProgID(szWideProgID, &clsid);  if (FAILED(hr))  {  printf("Unable to get CLSID from ProgID. HR = %X \n", hr); return -1;  }  IClassFactory\* pCF;  // Получить фабрику класса для CMath  hr = CoGetClassObject(clsid, CLSCTX\_INPROC, NULL,  IID\_IClassFactory, (void\*\*)&pCF);  if (FAILED(hr))  {  printf("Failed to GetClassObject server instance. HR = %X \n", hr); return -1;  }  // с помощью фабрики класса создать экземпляр  // компонента и получить интерфейс IUnknown.  IUnknown\* pUnk; hr = pCF->CreateInstance(NULL, IID\_IUnknown, (void\*\*)&pUnk);  // Release the class factory pCF->Release();  if (FAILED(hr))  {  printf("Failed to create server instance. HR =%X \n", hr); return -1;  }  printf("Instance created \n");  IMath\* pMath = NULL;  hr = pUnk->QueryInterface(IID\_IMATH, (LPVOID\*)&pMath);  pUnk->Release();  if (FAILED(hr))  {  printf("QueryInterface() for IMath failed \n"); return -1;  }  long result;  pMath->Multiply(20, 7, &result);  printf("20 \* 7 is %d \n", result);  pMath->Subtract(200, 123, &result);  printf("200 - 123 is %d \n", result);  printf("Releasing instance \n");  pMath->Release();  printf("Shuting down COM\n");  CoUninitialize();  return 0;  } |

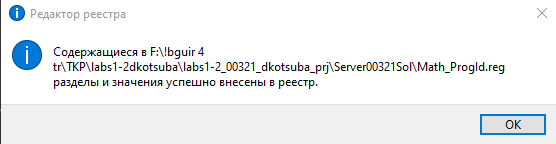
Пересоборем проект и запустим клиент до регистрации сервера COM

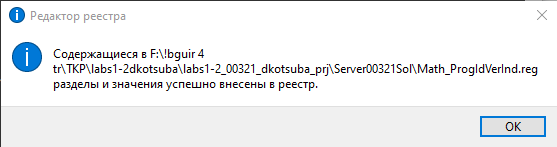


Запускаем консоль от имени администратора для регистрации в реестре

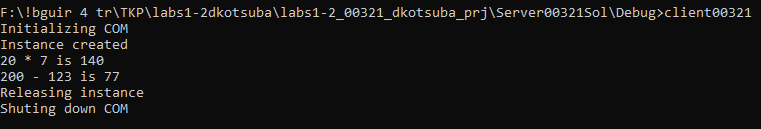




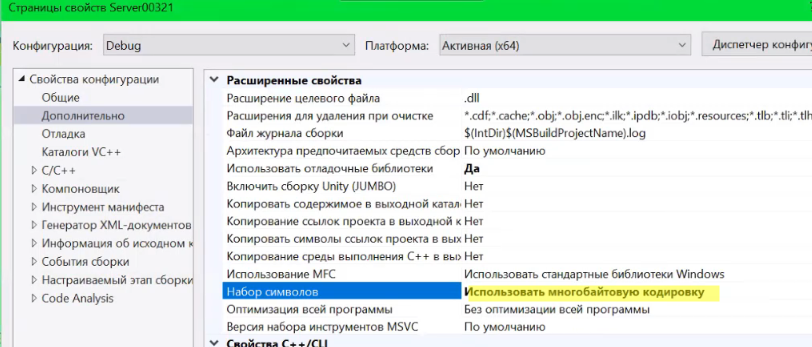




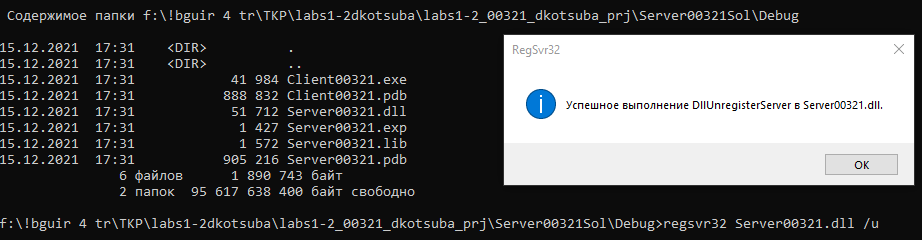
Проверка корректности взаимодействия с сервером



**2 Создание динамически загружаемую библиотеку, реализующую сервер c интерфейсом IMath.**



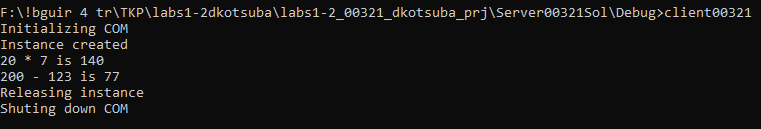
Пересоберем решение и отрегистрируем сервер regsvr32 Server00321.dll /u



Запустим клиент



Зарегистрируем сервер (regsvr32 Server00321.dll) и запустим клиент еще **раз**



Добавим реализацию второго интерфейса. Сгенерируем для него идентификатор, опишем интерфейс.

IMath.H

|  |
| --- |
| // {2F85142B-F184-4106-B73A-767845260D1F}  DEFINE\_GUID(IID\_IVer ,  0x2f85142b, 0xf184, 0x4106, 0xb7, 0x3a, 0x76, 0x78, 0x45, 0x26, 0xd, 0x1f);  […] |
| class IVer : public IUnknown {  public:  STDMETHOD(GetAuthor(wchar\_t\*\*autor)) PURE;  }; |

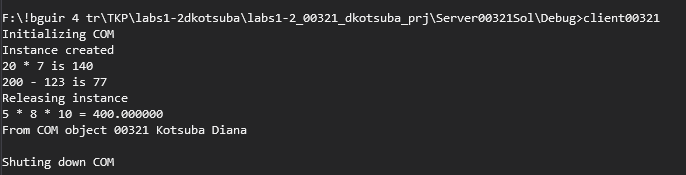
CMath.cpp

|  |
| --- |
| STDMETHODIMP CMath::GetAuthor(wchar\_t\*\* author)  {  wchar\_t\* p = (wchar\_t\*)CoTaskMemAlloc(200);  lstrcpyW(p, L"From COM object 00321 Kotsuba Diana\n");  \*author = p;  return S\_OK;  } |

Client00321.cpp

|  |
| --- |
| IVer\* pVer = NULL;  hr = pMath->QueryInterface(IID\_IVer, (LPVOID\*)&pVer);  pMath->Release();  wchar\_t str\_author[200];  wchar\_t\* pstr;  hr = pVer->GetAuthor(&pstr);  lstrcpyW(str\_author, pstr);  CoTaskMemFree(pstr);  pVer->Release();  wprintf(L"%s", str\_author); |

Скомпилируем и запустим



**Самостоятельная работа. Вариант 14**

IFN\_14

double Fun141 (int, int)

int Fun142 (int, int, int)

void Fun143 (double in, double \*out )

IMath.h

|  |
| --- |
| […]  // {ED809CF3-DEE3-47F4-B8A3-BF5660B21104}  DEFINE\_GUID(IID\_IFN\_14,  0xed809cf3, 0xdee3, 0x47f4, 0xb8, 0xa3, 0xbf, 0x56, 0x60, 0xb2, 0x11, 0x4);  […]  public:  STDMETHOD(Fun0(int, int, float\*)) PURE;  virtual double Fun141(int, int) PURE;  virtual int Fun142(int, int, int) PURE;  STDMETHOD(Fun143(double, double\*)) PURE;  }; |

CMath.h

|  |
| --- |
| //IFN\_14  STDMETHOD(Fun0(int, int, float\*));  double Fun141(int, int);  int Fun142(int, int, int);  STDMETHOD(Fun143(double, double\*)); |

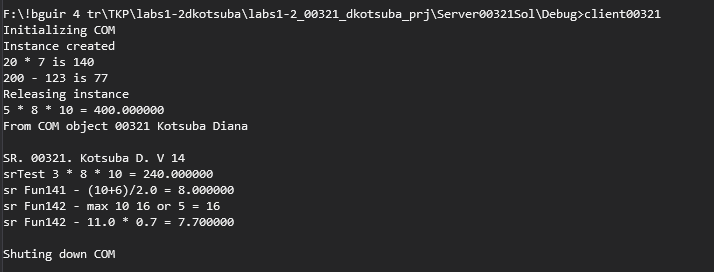
CMath.cpp

|  |
| --- |
| double CMath::Fun141(int lOp1, int lOp2)  {  double r = ((double)lOp1 + (double)lOp2)/2.0;  return r;  }  int CMath::Fun142(int lOp1, int lOp2, int lOp3)  {  int r;  if (lOp1 > lOp2)  r = lOp1;  else  r = lOp2;  if (lOp3 > r)  r = lOp3;  return r;  }  STDMETHODIMP CMath::Fun143(double lOp1, double\* pResult)  {  \*pResult = lOp1 \* 0.7;  return S\_OK;  } |

Client00321.cpp

|  |
| --- |
| /\*---------SR----------------\*/  printf("\nSR. 00321. Kotsuba D. V 14 \n");  IUnknown\* pUnkSR;  hr = pCF->CreateInstance(NULL, IID\_IUnknown, (void\*\*)&pUnkSR);  // Release the class factory pCF->Release();  if (FAILED(hr))  {  printf("Failed to create server instance. HR =%X \n", hr); return -1;  }  // printf("Instance created \n");  IFN\_14\* pIFN\_14SR = NULL;  hr = pUnkSR->QueryInterface(IID\_IFN\_14, (LPVOID\*)&pIFN\_14SR);  if (FAILED(hr))  {  printf("QueryInterface() for SR\_14 failed \n"); return -1;  }  pUnkSR->Release();  float floatResult1 = 0;  pIFN\_14SR->Fun0(3, 8, &floatResult1);  printf("srTest 3 \* 8 \* 10 = %lf \n", floatResult1);    printf("sr Fun141 - (10+6)/2.0 = %lf \n", pIFN\_14SR->Fun141(10, 6));    printf("sr Fun142 - max 10 16 or 5 = %li \n", pIFN\_14SR->Fun142(10, 16, 5));  double doubleResult;  pIFN\_14SR->Fun143(11.0, &doubleResult);  printf("sr Fun142 - 11.0 \* 0.7 = %lf \n", doubleResult);  pIFN\_14SR->Release();      printf("\nShuting down COM\n"); |

В результате получим:



**Выводы:**

В ходе выполнения лабораторной работы были получены практические навыки по создания простого компонента и его использования, созданию динамически загружаемую библиотеку, реализующую сервер c интерфейсом IMath.

Код программы.

Проект Server00321

|  |
| --- |
| #pragma once  //  // сmath.h  //  #include "IMath.h"  extern long g\_lObjs; //счетчик компонентов, созданных на базе сервера; память для него будет выделена в другом файле  extern long g\_lLocks; //счетчик блокировок  class CMath : public IMath, IVer, IFN\_14  {  protected:  // Reference count  long m\_lRef;  public:  CMath();  ~CMath();  public:  // IUnknown  STDMETHOD(QueryInterface(REFIID, void\*\*));  STDMETHOD\_(ULONG, AddRef());  STDMETHOD\_(ULONG, Release());    // IMath  STDMETHOD(Add(long, long, long\*));  STDMETHOD(Subtract(long, long, long\*));  STDMETHOD(Multiply(long, long, long\*));  STDMETHOD(Divide(long, long, long\*));  STDMETHOD(FunTest(int, int, float\*));    //IVer  STDMETHOD(GetAuthor(wchar\_t\*\* autor));    //IFN\_14  STDMETHOD(Fun0(int, int, float\*));  double Fun141(int, int);  int Fun142(int, int, int);  STDMETHOD(Fun143(double, double\*));    };  class MathClassFactory : public IClassFactory  {  protected:  long m\_lRef;  public:  MathClassFactory();  ~MathClassFactory();  // IUnknown  STDMETHOD(QueryInterface(REFIID, void\*\*));  STDMETHOD\_(ULONG, AddRef());  STDMETHOD\_(ULONG, Release());  // IClassFactory  STDMETHOD(CreateInstance(LPUNKNOWN, REFIID, void\*\*));  STDMETHOD(LockServer(BOOL));  }; |

|  |
| --- |
| //  //REGISTRY.h  //  #ifndef \_\_Registry\_H\_\_  #define \_\_Registry\_H\_\_  //  // Registry.h  // - Helper functions registering and unregistering a component.  //  // This function will register a component in the Registry.  // The component calls this function from its DllRegisterServer function.  HRESULT RegisterServer(HMODULE hModule,  const CLSID& clsid,  const char\* szFriendlyName,  const char\* szVerIndProgID,  const char\* szProgID) ;  // This function will unregister a component. Components  // call this function from their DllUnregisterServer function.  HRESULT UnregisterServer(const CLSID& clsid,  const char\* szVerIndProgID,  const char\* szProgID) ;  #endif |

|  |
| --- |
| #pragma once  #include <Windows.h> //для наших макро DEFINE\_GUID, STDMETHOD...  //  // imath.h  //  //Идентификатор класса компонента  // {0BAED82A-2EB8-4FF8-912F-128A4E904BB5}  DEFINE\_GUID(CLSID\_MATH ,  0xbaed82a, 0x2eb8, 0x4ff8, 0x91, 0x2f, 0x12, 0x8a, 0x4e, 0x90, 0x4b, 0xb5);  //Идентификатор интерфейса  // {56D7AF4F-C48A-4190-8880-19962537A9AE}  DEFINE\_GUID(IID\_IMATH ,  0x56d7af4f, 0xc48a, 0x4190, 0x88, 0x80, 0x19, 0x96, 0x25, 0x37, 0xa9, 0xae);  // {2F85142B-F184-4106-B73A-767845260D1F}  DEFINE\_GUID(IID\_IVer ,  0x2f85142b, 0xf184, 0x4106, 0xb7, 0x3a, 0x76, 0x78, 0x45, 0x26, 0xd, 0x1f);  // {ED809CF3-DEE3-47F4-B8A3-BF5660B21104}  DEFINE\_GUID(IID\_IFN\_14,  0xed809cf3, 0xdee3, 0x47f4, 0xb8, 0xa3, 0xbf, 0x56, 0x60, 0xb2, 0x11, 0x4);  class IMath : public IUnknown  {  public:  STDMETHOD( Add( long, long, long\* )) PURE;  STDMETHOD( Subtract( long, long, long\* )) PURE;  STDMETHOD( Multiply( long, long, long\* )) PURE;  STDMETHOD( Divide( long, long, long\* )) PURE;  STDMETHOD(FunTest(int, int, float\*)) PURE;  };  class IVer : public IUnknown {  public:  STDMETHOD(GetAuthor(wchar\_t\*\*autor)) PURE;  };  class IFN\_14 : public IUnknown  {  public:  STDMETHOD(Fun0(int, int, float\*)) PURE;  virtual double Fun141(int, int) PURE;  virtual int Fun142(int, int, int) PURE;  STDMETHOD(Fun143(double, double\*)) PURE;  }; |

|  |
| --- |
| ; Server00321.def  LIBRARY  EXPORTS  ; Имена точек входа для внешнего пользования помещаются здесь  DllGetClassObject PRIVATE  DllCanUnloadNow PRIVATE  DllRegisterServer PRIVATE  DllUnregisterServer PRIVATE |

|  |
| --- |
| //  // server.cpp : Defines the initialization routines for the DLL.  //  #include <windows.h>  #include <initguid.h>  #include "CMath.h"  #include "REGISTRY.H"  ///////////////////////////////////////////////////////////  //  // Global variables  //  static HMODULE g\_hModule = NULL; // DLL module handle  // Friendly name of component  const char g\_szFriendlyName[] = "TCP\_Lab1 Math Component 00321";  // Version-independent ProgID  const char g\_szVerIndProgID[] = "Math.Component";  // ProgID  const char g\_szProgID[] = "Math.Component.1";  long g\_lObjs = 0; //счетчик объектов  long g\_lLocks = 0; //счетчик блокировок  STDAPI DllGetClassObject(REFCLSID rclsid, REFIID riid, void\*\*  ppv)  {  HRESULT hr;  MathClassFactory\* pCF;  pCF = 0;  // Make sure the CLSID is for our Expression component  if (rclsid != CLSID\_MATH)  return(E\_FAIL);  pCF = new MathClassFactory;  if (pCF == 0)  return(E\_OUTOFMEMORY);  hr = pCF->QueryInterface(riid, ppv);  // Check for failure of QueryInterface  if (FAILED(hr))  {  delete pCF;  pCF = 0;  }  return hr;  }  STDAPI DllCanUnloadNow(void)  {  if (g\_lObjs || g\_lLocks)  return(S\_FALSE);  else  return(S\_OK);  }  //  // Server registration  //  STDAPI DllRegisterServer()  {  return RegisterServer(g\_hModule,  CLSID\_MATH,  g\_szFriendlyName,  g\_szVerIndProgID,  g\_szProgID);  }  //  // Server unregistration  //  STDAPI DllUnregisterServer()  {  return UnregisterServer(CLSID\_MATH,  g\_szVerIndProgID,  g\_szProgID);  }  ///////////////////////////////////////////////////////////  //  // DLL module information  //  BOOL APIENTRY DllMain(HANDLE hModule,  DWORD dwReason,  void\* lpReserved)  {  if (dwReason == DLL\_PROCESS\_ATTACH)  {  g\_hModule = (HMODULE)hModule;  }  return TRUE;  } |

|  |
| --- |
| //  // Registry.cpp  //  #define \_CRT\_SECURE\_NO\_WARNINGS  #include <objbase.h>  #include <assert.h>  #include "Registry.h"  ////////////////////////////////////////////////////////  //  // Internal helper functions prototypes  //  // Set the given key and its value.  BOOL setKeyAndValue(const char\* pszPath,  const char\* szSubkey,  const char\* szValue) ;  // Convert a CLSID into a char string.  void CLSIDtochar(const CLSID& clsid,  char\* szCLSID,  int length) ;  // Delete szKeyChild and all of its descendents.  LONG recursiveDeleteKey(HKEY hKeyParent, const char\* szKeyChild) ;  ////////////////////////////////////////////////////////  //  // Constants  //  // Size of a CLSID as a string  const int CLSID\_STRING\_SIZE = 39 ;  /////////////////////////////////////////////////////////  //  // Public function implementation  //  //  // Register the component in the registry.  //  HRESULT RegisterServer(HMODULE hModule, // DLL module handle  const CLSID& clsid, // Class ID  const char\* szFriendlyName, // Friendly Name  const char\* szVerIndProgID, // Programmatic  const char\* szProgID) // IDs  {  // Get server location.  char szModule[512] ;  DWORD dwResult =  ::GetModuleFileName(hModule,  szModule,  sizeof(szModule)/sizeof(char)) ;  assert(dwResult != 0) ;  // Convert the CLSID into a char.  char szCLSID[CLSID\_STRING\_SIZE] ;  CLSIDtochar(clsid, szCLSID, sizeof(szCLSID)) ;  // Build the key CLSID\\{...}  char szKey[64] ;  strcpy(szKey, "CLSID\\") ;  strcat(szKey, szCLSID) ;    // Add the CLSID to the registry.  setKeyAndValue(szKey, NULL, szFriendlyName) ;  // Add the server filename subkey under the CLSID key.  setKeyAndValue(szKey, "InprocServer32", szModule) ;  // Add the ProgID subkey under the CLSID key.  setKeyAndValue(szKey, "ProgID", szProgID) ;  // Add the version-independent ProgID subkey under CLSID key.  setKeyAndValue(szKey, "VersionIndependentProgID",  szVerIndProgID) ;  // Add the version-independent ProgID subkey under HKEY\_CLASSES\_ROOT.  setKeyAndValue(szVerIndProgID, NULL, szFriendlyName) ;  setKeyAndValue(szVerIndProgID, "CLSID", szCLSID) ;  setKeyAndValue(szVerIndProgID, "CurVer", szProgID) ;  // Add the versioned ProgID subkey under HKEY\_CLASSES\_ROOT.  setKeyAndValue(szProgID, NULL, szFriendlyName) ;  setKeyAndValue(szProgID, "CLSID", szCLSID) ;  return S\_OK ;  }  //  // Remove the component from the registry.  //  LONG UnregisterServer(const CLSID& clsid, // Class ID  const char\* szVerIndProgID, // Programmatic  const char\* szProgID) // IDs  {  // Convert the CLSID into a char.  char szCLSID[CLSID\_STRING\_SIZE] ;  CLSIDtochar(clsid, szCLSID, sizeof(szCLSID)) ;  // Build the key CLSID\\{...}  char szKey[64] ;  strcpy(szKey, "CLSID\\") ;  strcat(szKey, szCLSID) ;  // Delete the CLSID Key - CLSID\{...}  LONG lResult = recursiveDeleteKey(HKEY\_CLASSES\_ROOT, szKey) ;  assert((lResult == ERROR\_SUCCESS) ||  (lResult == ERROR\_FILE\_NOT\_FOUND)) ; // Subkey may not exist.  // Delete the version-independent ProgID Key.  lResult = recursiveDeleteKey(HKEY\_CLASSES\_ROOT, szVerIndProgID) ;  assert((lResult == ERROR\_SUCCESS) ||  (lResult == ERROR\_FILE\_NOT\_FOUND)) ; // Subkey may not exist.  // Delete the ProgID key.  lResult = recursiveDeleteKey(HKEY\_CLASSES\_ROOT, szProgID) ;  assert((lResult == ERROR\_SUCCESS) ||  (lResult == ERROR\_FILE\_NOT\_FOUND)) ; // Subkey may not exist.  return S\_OK ;  }  ///////////////////////////////////////////////////////////  //  // Internal helper functions  //  // Convert a CLSID to a char string.  void CLSIDtochar(const CLSID& clsid,  char\* szCLSID,  int length)  {  assert(length >= CLSID\_STRING\_SIZE) ;  // Get CLSID  LPOLESTR wszCLSID = NULL ;  HRESULT hr = StringFromCLSID(clsid, &wszCLSID) ;  assert(SUCCEEDED(hr)) ;  // Covert from wide characters to non-wide.  wcstombs(szCLSID, wszCLSID, length) ;  // Free memory.  CoTaskMemFree(wszCLSID) ;  }  //  // Delete a key and all of its descendents.  //  LONG recursiveDeleteKey(HKEY hKeyParent, // Parent of key to delete  const char\* lpszKeyChild) // Key to delete  {  // Open the child.  HKEY hKeyChild ;  LONG lRes = RegOpenKeyEx(hKeyParent, lpszKeyChild, 0,  KEY\_ALL\_ACCESS, &hKeyChild) ;  if (lRes != ERROR\_SUCCESS)  {  return lRes ;  }  // Enumerate all of the decendents of this child.  FILETIME time ;  char szBuffer[256] ;  DWORD dwSize = 256 ;  while (RegEnumKeyEx(hKeyChild, 0, szBuffer, &dwSize, NULL,  NULL, NULL, &time) == S\_OK)  {  // Delete the decendents of this child.  lRes = recursiveDeleteKey(hKeyChild, szBuffer) ;  if (lRes != ERROR\_SUCCESS)  {  // Cleanup before exiting.  RegCloseKey(hKeyChild) ;  return lRes;  }  dwSize = 256 ;  }  // Close the child.  RegCloseKey(hKeyChild) ;  // Delete this child.  return RegDeleteKey(hKeyParent, lpszKeyChild) ;  }  //  // Create a key and set its value.  // - This helper function was borrowed and modifed from  // Kraig Brockschmidt's book Inside OLE.  //  BOOL setKeyAndValue(const char\* szKey,  const char\* szSubkey,  const char\* szValue)  {  HKEY hKey;  char szKeyBuf[1024] ;  // Copy keyname into buffer.  strcpy(szKeyBuf, szKey) ;  // Add subkey name to buffer.  if (szSubkey != NULL)  {  strcat(szKeyBuf, "\\") ;  strcat(szKeyBuf, szSubkey ) ;  }  // Create and open key and subkey.  long lResult = RegCreateKeyEx(HKEY\_CLASSES\_ROOT ,  szKeyBuf,  0, NULL, REG\_OPTION\_NON\_VOLATILE,  KEY\_ALL\_ACCESS, NULL,  &hKey, NULL) ;  if (lResult != ERROR\_SUCCESS)  {  return FALSE ;  }  // Set the Value.  if (szValue != NULL)  {  RegSetValueEx(hKey, NULL, 0, REG\_SZ,  (BYTE \*)szValue,  strlen(szValue)+1) ;  }  RegCloseKey(hKey) ;  return TRUE ;  } |

|  |
| --- |
| //  // СMath.cpp  //  #include <windows.h>  #include "CMath.h"  //  // Math class implementation  //  // Constructors  CMath::CMath()  {  m\_lRef = 0;  // Увеличить значение внешнего счетчика объектов  InterlockedIncrement(&g\_lObjs);  }  // The destructor  CMath::~CMath()  {  // Уменьшить значение внешнего счетчика объектов  InterlockedDecrement(&g\_lObjs);  }  STDMETHODIMP CMath::QueryInterface(REFIID riid, void\*\* ppv)  {  /\*\*ppv = 0;  if (riid == IID\_IUnknown || riid == IID\_IMATH)  \*ppv = this;\*/  \*ppv = 0;  if (riid == IID\_IUnknown)  \*ppv = (IMath\*)this;  else if (riid == IID\_IMATH)  \*ppv = (IMath\*)this;  else if (riid == IID\_IVer)  \*ppv = (IVer\*)this;  else if (riid == IID\_IFN\_14)  \*ppv = (IFN\_14\*)this;    if (\*ppv)  {  AddRef();  return(S\_OK);  }  return (E\_NOINTERFACE);  }  STDMETHODIMP\_(ULONG) CMath::AddRef()  {  return InterlockedIncrement(&m\_lRef);  }  STDMETHODIMP\_(ULONG) CMath::Release()  {  if (InterlockedDecrement(&m\_lRef) == 0)  {  delete this;  return 0;  }  return m\_lRef;  }  STDMETHODIMP CMath::Add(long lOp1, long lOp2, long\* pResult)  {  \*pResult = lOp1 + lOp2;  return S\_OK;  }  STDMETHODIMP CMath::Subtract(long lOp1, long lOp2, long\*  pResult)  {  \*pResult = lOp1 - lOp2;  return S\_OK;  }  STDMETHODIMP CMath::Multiply(long lOp1, long lOp2, long\*  pResult)  {  \*pResult = lOp1 \* lOp2;  return S\_OK;  }  STDMETHODIMP CMath::Divide(long lOp1, long lOp2, long\* pResult  )  {  \*pResult = lOp1 / lOp2;  return S\_OK;  }  STDMETHODIMP CMath::FunTest(int lOp1, int lOp2, float\* pResult)  {  \*pResult = lOp1 \* lOp2 \* 10;  return S\_OK;  }  STDMETHODIMP CMath::GetAuthor(wchar\_t\*\* author)  {  wchar\_t\* p = (wchar\_t\*)CoTaskMemAlloc(200);  lstrcpyW(p, L"From COM object 00321 Kotsuba Diana\n");  \*author = p;  return S\_OK;  }  STDMETHODIMP CMath::Fun0(int lOp1, int lOp2, float\* pResult)  {  \*pResult = lOp1 \* lOp2 \* 10;  return S\_OK;  }  double CMath::Fun141(int lOp1, int lOp2)  {  double r = ((double)lOp1 + (double)lOp2)/2.0;  return r;  }  int CMath::Fun142(int lOp1, int lOp2, int lOp3)  {  int r;  if (lOp1 > lOp2)  r = lOp1;  else  r = lOp2;  if (lOp3 > r)  r = lOp3;  return r;  }  STDMETHODIMP CMath::Fun143(double lOp1, double\* pResult)  {  \*pResult = lOp1 \* 0.7;  return S\_OK;  }  //Фабрика класса  MathClassFactory::MathClassFactory()  {  m\_lRef = 0;  }  MathClassFactory::~MathClassFactory()  {  }  STDMETHODIMP MathClassFactory::QueryInterface(REFIID riid,  void\*\* ppv)  {  \*ppv = 0;  if (riid == IID\_IUnknown || riid == IID\_IClassFactory)  \*ppv = this;  if (\*ppv)  {  AddRef();  return S\_OK;  }  return(E\_NOINTERFACE);  }  STDMETHODIMP\_(ULONG) MathClassFactory::AddRef()  {  return InterlockedIncrement(&m\_lRef);  }  STDMETHODIMP\_(ULONG) MathClassFactory::Release()  {  if (InterlockedDecrement(&m\_lRef) == 0)  {  delete this;  return 0;  }  return m\_lRef;  }  STDMETHODIMP MathClassFactory::CreateInstance  (LPUNKNOWN pUnkOuter, REFIID riid, void\*\* ppvObj)  {  CMath\* pMath;  HRESULT hr;  \*ppvObj = 0;  pMath = new CMath;  if (pMath == 0)  return(E\_OUTOFMEMORY);  hr = pMath->QueryInterface(riid, ppvObj);  if (FAILED(hr))  delete pMath;  return hr;  }  STDMETHODIMP MathClassFactory::LockServer(BOOL fLock)  {  if (fLock)  InterlockedIncrement(&g\_lLocks);  else  InterlockedDecrement(&g\_lLocks);  return S\_OK;  } |

Проект Client00321

|  |
| --- |
| //kotsuba  // Client.cpp  //  #include <windows.h>  #include <tchar.h>  #include <stdio.h>  #include <initguid.h>  #include "..\Server00321\IMath.h"  int main(int argc, char\* argv[])  {  printf("Initializing COM \n");  if (FAILED(CoInitialize(NULL)))  {  printf("Unable to initialize COM \n"); return -1;  }  const char\* szProgID = "Math.Component.1"; // - Program ID  WCHAR szWideProgID[128];  CLSID clsid;  long lLen = MultiByteToWideChar(CP\_ACP, 0, szProgID,  strlen(szProgID), szWideProgID, sizeof(szWideProgID));  szWideProgID[lLen] = '\0';  HRESULT hr = ::CLSIDFromProgID(szWideProgID, &clsid);  if (FAILED(hr))  {  printf("Unable to get CLSID from ProgID. HR = %X \n", hr); return -1;  }  IClassFactory\* pCF;  // Получить фабрику класса для CMath  hr = CoGetClassObject(clsid, CLSCTX\_INPROC, NULL,  IID\_IClassFactory, (void\*\*)&pCF);  if (FAILED(hr))  {  printf("Failed to GetClassObject server instance. HR = %X \n", hr); return -1;  }  // с помощью фабрики класса создать экземпляр  // компонента и получить интерфейс IUnknown.  IUnknown\* pUnk; hr = pCF->CreateInstance(NULL, IID\_IUnknown, (void\*\*)&pUnk);  // Release the class factory pCF->Release();  if (FAILED(hr))  {  printf("Failed to create server instance. HR =%X \n", hr); return -1;  }  printf("Instance created \n");  IMath\* pMath = NULL;  hr = pUnk->QueryInterface(IID\_IMATH, (LPVOID\*)&pMath);  if (FAILED(hr))  {  printf("QueryInterface() for IMath failed \n"); return -1;  }  pUnk->Release();  long result;  pMath->Multiply(20, 7, &result);  printf("20 \* 7 is %d \n", result);  pMath->Subtract(200, 123, &result);  printf("200 - 123 is %d \n", result);  printf("Releasing instance \n");  float floatResult0 = 0;  pMath->FunTest(5, 8, &floatResult0);  printf("5 \* 8 \* 10 = %lf \n", floatResult0);  IVer\* pVer = NULL;  hr = pMath->QueryInterface(IID\_IVer, (LPVOID\*)&pVer);  pMath->Release();  wchar\_t str\_author[200];  wchar\_t\* pstr;  hr = pVer->GetAuthor(&pstr);  lstrcpyW(str\_author, pstr);  CoTaskMemFree(pstr);  pVer->Release();  wprintf(L"%s", str\_author);    /\*---------SR----------------\*/  printf("\nSR. 00321. Kotsuba D. V 14 \n");  IUnknown\* pUnkSR;  hr = pCF->CreateInstance(NULL, IID\_IUnknown, (void\*\*)&pUnkSR);  // Release the class factory pCF->Release();  if (FAILED(hr))  {  printf("Failed to create server instance. HR =%X \n", hr); return -1;  }  // printf("Instance created \n");  IFN\_14\* pIFN\_14SR = NULL;  hr = pUnkSR->QueryInterface(IID\_IFN\_14, (LPVOID\*)&pIFN\_14SR);  if (FAILED(hr))  {  printf("QueryInterface() for SR\_14 failed \n"); return -1;  }  pUnkSR->Release();  float floatResult1 = 0;  pIFN\_14SR->Fun0(3, 8, &floatResult1);  printf("srTest 3 \* 8 \* 10 = %lf \n", floatResult1);    printf("sr Fun141 - (10+6)/2.0 = %lf \n", pIFN\_14SR->Fun141(10, 6));    printf("sr Fun142 - max 10 16 or 5 = %li \n", pIFN\_14SR->Fun142(10, 16, 5));  double doubleResult;  pIFN\_14SR->Fun143(11.0, &doubleResult);  printf("sr Fun142 - 11.0 \* 0.7 = %lf \n", doubleResult);  pIFN\_14SR->Release();      printf("\nShuting down COM\n");  CoUninitialize();  return 0;  } |