УО «ВГУ им. П.М. Машерова»
Кафедра прикладной математики и механики

В.В. Новый

Лабораторная работа по теме «Знакомство со средствами разработки для COM/COM+»

Практикум по дисциплине «Компонентные технологии Microsoft» для студентов 4 курса специальности Прикладная математика (1-31 03 03)

Витебск, 2010

Лабораторная работа №1

Знакомство со средствами разработки для COM/COM+

Основной сущностью в технологии COM является COM-объект. Так же как и объекты в C++ и Java он представляет собой некоторую сущность, имеющую методы доступа и свое состояние. COM-объекты можно создавать вызовом специальных функции, но напрямую уничтожить – нельзя. Для удаления объекта используется механизм самоуничтожения, основанный на reference counting – подсчете ссылок на объект. Класс объекта в COM называется CoClass. CoClass поддерживает набор методов и свойств, с помощью которых можно взаимодействовать с объектами этого класса. Этот набор называется интерфейсом.

Каждый CoClass имеет двоичный идентификатор класса – CLSID и текстовый идентификатор – ProgID. Генерируются любые GUID с помощью функции CoCreateGuid.

Для описания интерфейсов в COM используется язык IDL, точнее его реализация MIDL (MS Interface Definition Language). Для C++ он используется часто, в Object Pascal (Delphi) и VB не используется совсем.

MIDL позволяет не только описать интерфейс, но и задать необходимые атрибуты. Атрибуты заключаются в [ ]. Основным атрибутом является uuid, который задает IID интерфейса. Например:

```
[
   uuid(A3992B80-10CF-1111-AB3C-254500DDA18D),
   helpstring("Этот интерфейс управляет запуском и остановкой приложения")
]
interface IControl : IUnknown
{
   HRESULT Start();
   HRESULT Stop();
}
```

Главным по важности интерфейсом является IUnknown. Как вы знаете он определяет 3 метода. У программиста есть две возможности: либо реализовать эти методы самостоятельно, либо воспользоваться какой-либо программной библиотекой (чаще всего ATL – библиотекой шаблонов на C++).

Чтобы использовать COM-объект необходимо создать его экземпляр. Это можно выполнить с помощью функции `CoCreateInstance`, `CoCreateInstanceEx`, `CoGetObject`, `CoGetClassObject`, `CoGetInstanceFromFile`, `CoGetInstanceFromIStorage`, `OleLoadFromStream` и других. Чаще всего применяются функции `CoCreateInstance`, `CoCreateInstanceEx`, `CreateObject` и оператор `new`.

Чтобы создать экземпляр объекта при помощи CoCreateInstance, ей необходимо передать: CLSID требуемого объекта, контекст создаваемого объекта, IID интерфейса, и указатель, в который и будет возвращен указатель на интерфейс созданного объекта.

Контекст создаваемого объекта – это информация о том, где должен создаваться объект: в том же процессе (адресном пространстве EXE-модуля), в другом процессе того же компьютера и др.

У этой функции есть еще один параметр, pUnkOuter, который будет рассмотрен позднее в лекции.

Для создания объекта COM функция берет информацию о сервере из реестра. Информацию о нахождении COM-сервера можно внести в реестр различными способами:

Задание 1. Регистрация COM-компонента. Зарегистрируйте тестовый компонент на локальном компьютере.

Самым простым способом регистрации компонента является использование утилиты REGSVR32.exe. Для регистрации используется следующий синтаксис:

REGSVR32.EXE <имя_файла_компонента>

Для удаления регистрации компонента используется команда:

REGSVR32.EXE /U < имя_файла_компонента>

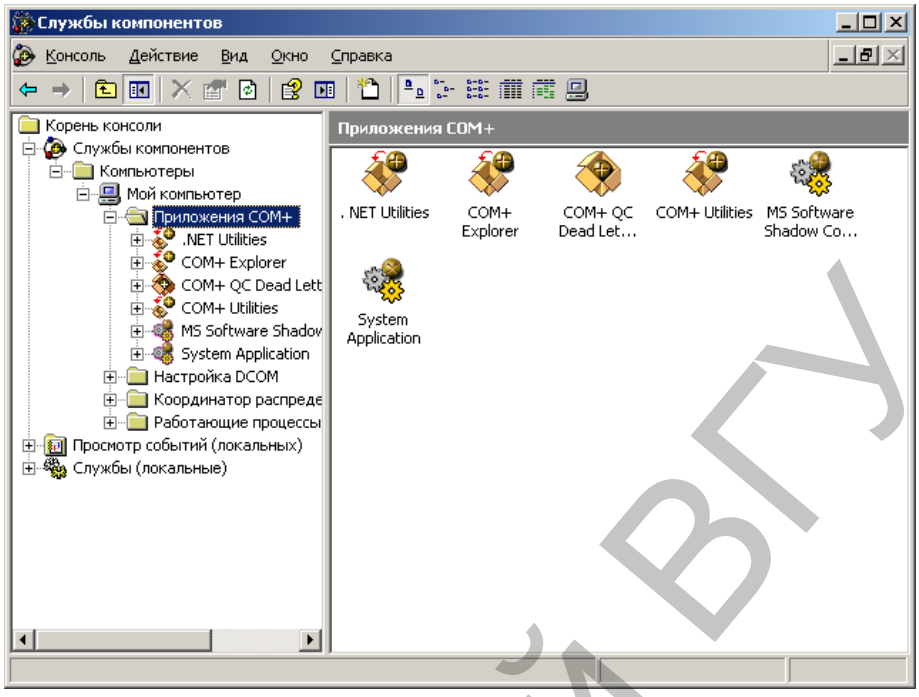
Другим вариантом регистрации является создание COM+ приложения с помощью средств администрирования Сервиса компонентов или DCOMCNFG.

Задание 2. Знакомство с Component Service. Одним из средств администрирования приложений COM/COM+ является Component Service из консоли администрирования. Для доступа к Component Service:

Пуск | Программы | Администрирование | Службы компонентов

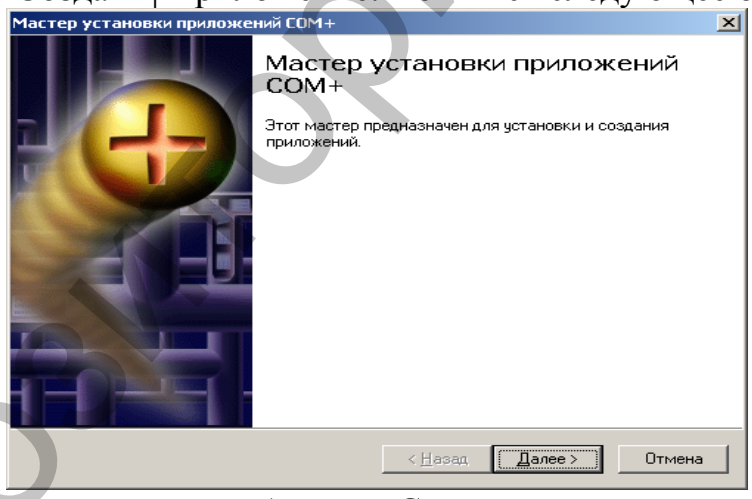
или

Пуск | Настройка | Панель управления | Администрирование | Службы компонентов.

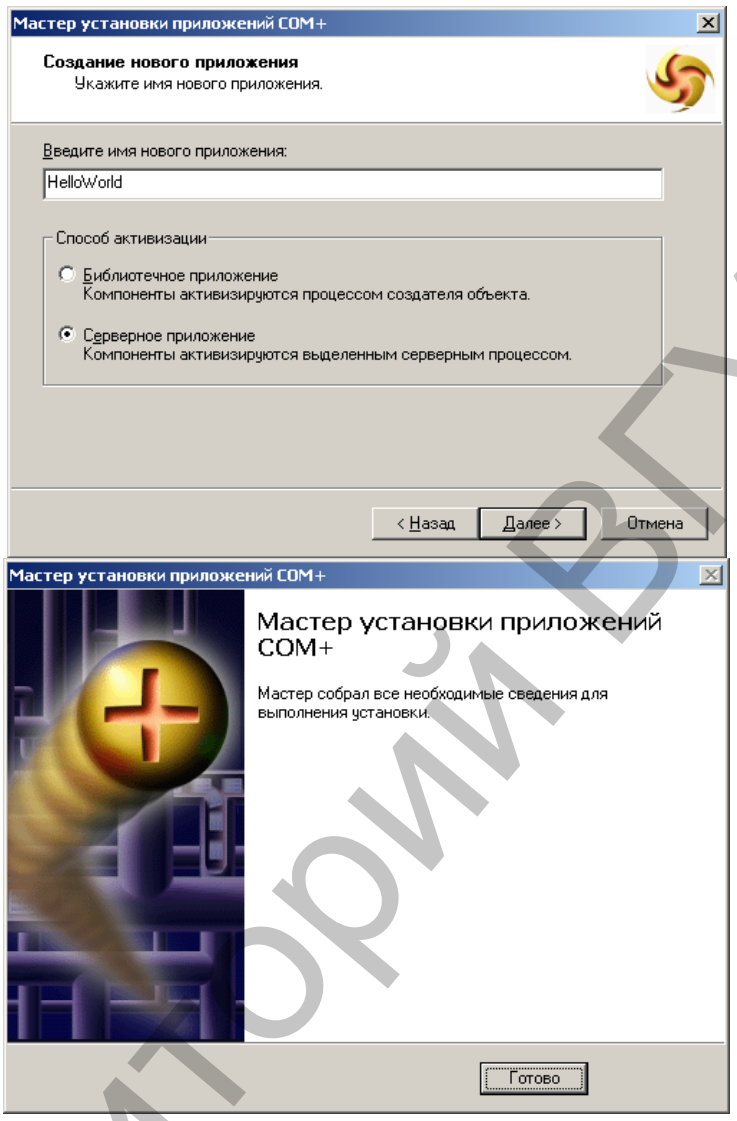


Рассмотрим пример создания COM+ приложения:

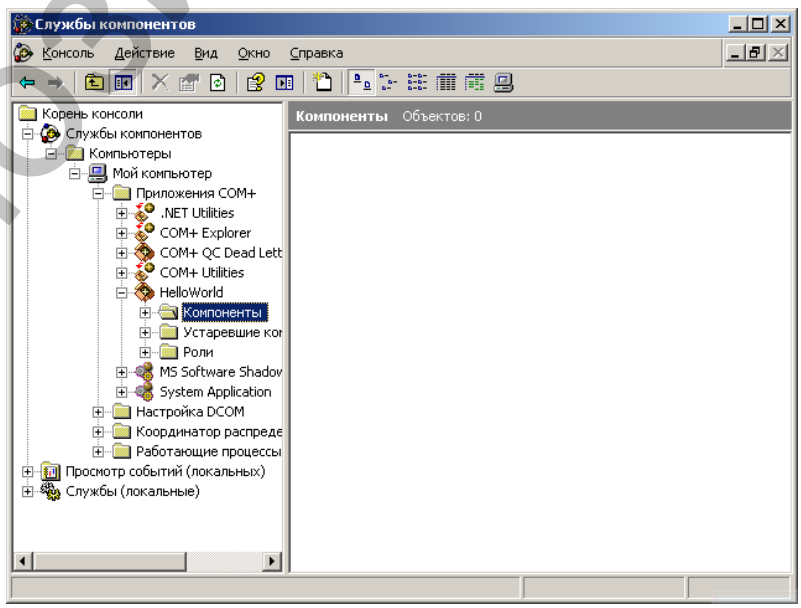
1) Создаем приложение COM+. Для этого выберите пункт «Приложения COM+» и, щелкнув на нем правой кнопкой мыши, выберите в контекстном меню Создать | Приложение. Появится следующее окно:



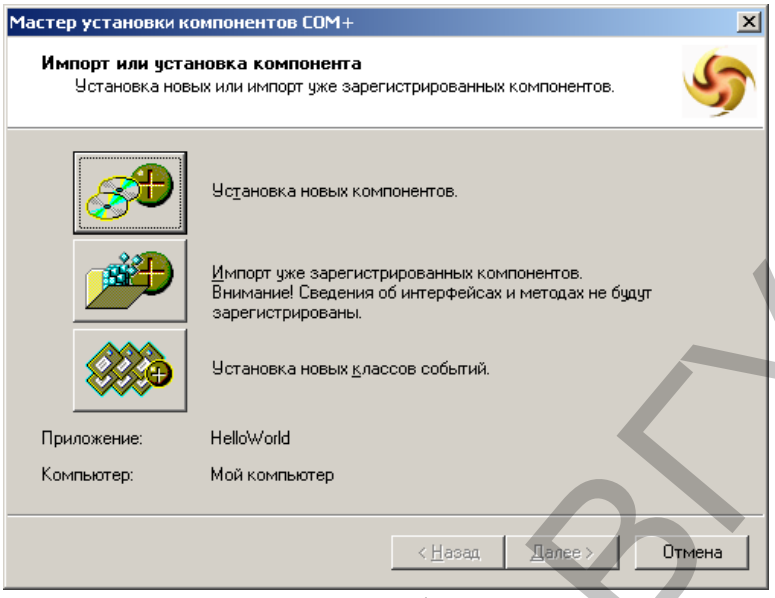
2) В мастере установки выберите «Создать новое приложение» и на следующем экране задайте имя создаваемого приложения, например, «HelloWorld» и согласитесь с настройками по умолчанию.



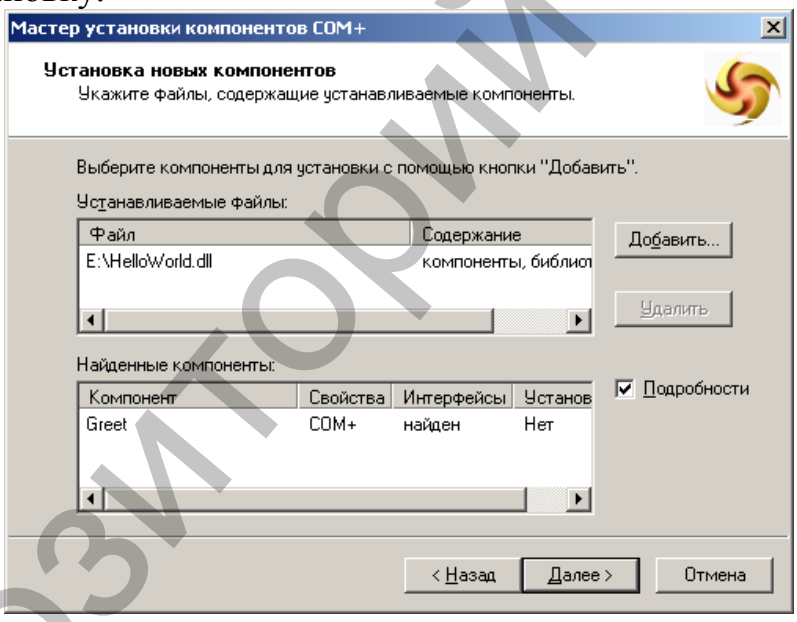
3) Следующий шаг – установка компонентов для созданного приложения:



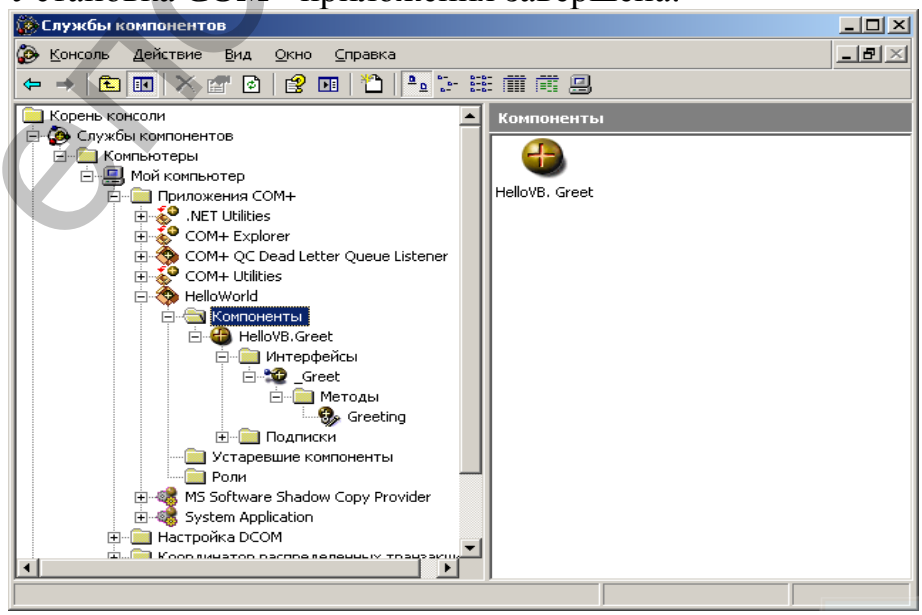
Для этого выберите подменю «Компоненты» меню приложения «HelloWorld» и в контекстном меню щелкните по пунктам Создать | Компонент. В окне мастера выберите «Установка новых компонентов».



Затем в диалоговом окне выберите файл примера «HelloWorld.dll» и подтвердите установку.



Установка COM+ приложения завершена:



Для удаления приложения COM+ выберите удаляемое приложение и в контекстном меню выберите команду «Удалить».

Другими вариантами создания COM-объекта является загрузка его состояния из различных источников или использование моникера. Моникер это такой COM-объект, который умеет создавать другие COM-объекты.

Физически COM-объекты размещаются в DLL или EXE файлах. COM-объекты из таких файлов загружаются автоматически при необходимости платформой COM. Процесс, в котором создан объект COM называется сервером, а процесс, который вызывает методы этого сервера – клиентом. Причем, в крайнем случае, это может быть один и тот же процесс.

Чаще всего COM-объекты помещаются в библиотеки – файлы с расширением DLL или OCX. EXE-файл для COM-объекта используется в том случае, если создается приложение поддерживающее технологию Automation или не требующее наличия COM+ или MTS.

В MIDL библиотека будет описываться, например, так:

```
[
   uuid(86479D03-B2D5-11D3-AE78-004095E1F072),
   version(1.0),
   helpstring("Библиотека управления 1.0")
]
library ControlLib
{
   [
      uuid(35F3208B-F9AC-11d3-A6AD-0050BAC0EF0F),
      helpstring("Lib Control")
   ]
   coclass Control
   {
      [default] interface iControl;
   };
}
```

Задание 3. Создание элементарного COM-объекта. Для создания COM-объекта проще всего воспользоваться библиотекой ATL и визардами, предоставляемыми Microsoft Visual Studio или Borland C++ Builder. (Внимание, библиотека ATL не входит в Express Edition Visual Studio).

Рассмотрим создание элементарного COM-объекта без использования библиотек – компонента для выполнения базовых математических операции.

Шаг 1. Определяем интерфейс компонента. Как Вам известно, любой интерфейс наследуется от IUnknown:

```
class IMath : public IUnknown
{
public:
```

```
        virtual HRESULT __stdcall Add( long, long, long* )=0;
        virtual HRESULT __stdcall Subtract( long, long, long*
) = 0;
        virtual HRESULT __stdcall Multiply( long, long, long*
) = 0;
        virtual HRESULT __stdcall Divide( long, long, long* )
= 0;
    };
```

Обычно подобная запись сокращается при помощи макросов, определенных в OBJBASE.H:

```
    #define STDMETHODCALLTYPE       __stdcall
    #define  STDMETHOD(method)           virtual  HRESULT
STDMETHODCALLTYPE method
    #define     STDMETHOD_(type,method)    virtual     type
STDMETHODCALLTYPE method
    #define PURE                    = 0
    #define  STDMETHODIMP                          HRESULT
STDMETHODCALLTYPE
    #define STDMETHODIMP_(type)    type STDMETHODCALLTYPE
```

После замены получим:

```
    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;
    };
```

Шаг 2. Генерируем GUID для проекта. Для этого используется, например, утилита uuidgen из состава Visual Studio:

```
    Uuidgen -n2 > GUIDS.txt
```

или программа GUIDGEN с графическим интерфейсом.

Дополним описание интерфейса сгенерированными GUID и запишем все в заголовочный файл imath.h:

```
    /* ------------------------------
    /             imath.h
    /------------------------------*/

    // {A888F560-58E4-11d0-A68A-0000837E3100}
    DEFINE_GUID( CLSID_Math,
                 0xa888f560,  0x58e4,  0x11d0,  0xa6,  0x8a,
0x0, 0x0, 0x83, 0x7e, 0x31, 0x0);
    // {A888F561-58E4-11d0-A68A-0000837E3100}
    DEFINE_GUID( IID_IMath,
```

```
                0xa888f561,  0x58e4,  0x11d0,  0xa6,  0x8a,
0x0, 0x0, 0x83, 0x7e, 0x31, 0x0);

    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;
    };
```

Шаг 3. Создание COM-сервера.

Так как данная тема не была рассмотрена до конца в лекции, приведем необходимые файлы целиком. Пояснения к их содержанию будут даны на лекции:

Файл math.h:

```
    /*----------------------------------------
    / math.h
    /----------------------------------------*/

    #include "imath.h"

    extern long g_lObjs;
    extern long g_lLocks;


    class Math : public IMath
    {
    protected:
       // Reference counter
       long         m_lRef;

    public:
       Math();
       ~Math();

    public:
       // интерфейс IUnknown
       STDMETHOD(QueryInterface( REFIID, void** ));
       STDMETHOD_(ULONG, AddRef());
       STDMETHOD_(ULONG, Release());

       // интерфейс 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));
};
```

Файл math.cpp:

```
/*---------------------------------
/            Math.cpp
/---------------------------------*/

#include <windows.h>
#include "math.h"

//
// реализация класса Math
//
Math::Math()
{
   m_lRef = 0;
   InterlockedIncrement( &g_lObjs );
}

Math::~Math()
{
```

```
    InterlockedDecrement( &g_lObjs );
}

STDMETHODIMP Math::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) Math::AddRef()
{
   return InterlockedIncrement( &m_lRef );
}

STDMETHODIMP_(ULONG) Math::Release()
{
   if ( InterlockedDecrement( &m_lRef ) == 0 )
   {
      delete this;
      return 0;
   }

   return m_lRef;
}

STDMETHODIMP Math::Add( long lOp1, long lOp2, long*
pResult )
{
   *pResult = lOp1 + lOp2;
   return S_OK;
}

STDMETHODIMP Math::Subtract( long lOp1, long lOp2, long*
pResult )
{
   *pResult = lOp1 - lOp2;
```

```
        return S_OK;
    }

    STDMETHODIMP Math::Multiply( long lOp1, long lOp2, long*
pResult )
    {
        *pResult = lOp1 * lOp2;
        return S_OK;
    }

    STDMETHODIMP_(long) Math::Divide( long lOp1, long lOp2,
long* pResult )
    {
        *pResult = lOp1 / lOp2;
        return S_OK;
    }

    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 )
{
   Math*      pMath;
   HRESULT    hr;
   *ppvObj = 0;
   pMath = new Math;
   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;
}
```

Файл server.cpp:

```
/*-------------------------------------------------
/ server.cpp : Определение функции DLL.
/-------------------------------------------------*/
#include <windows.h>
#include <initguid.h>
```

```
#include "math.h"

long    g_lObjs = 0;
long    g_lLocks = 0;

STDAPI DllGetClassObject( REFCLSID rclsid, REFIID riid,
void** ppv )
{
   HRESULT             hr;
   MathClassFactory    *pCF;
   pCF = 0;
   if ( rclsid != CLSID_Math )
      return( E_FAIL );
   pCF = new MathClassFactory;
   if ( pCF == 0 )
      return( E_OUTOFMEMORY );
   hr = pCF->QueryInterface( riid, ppv );
   if ( FAILED( hr ) )
   {
      delete pCF;
      pCF = 0;
   }
   return hr;
}

STDAPI DllCanUnloadNow(void)
{
    if ( g_lObjs || g_lLocks )
       return( S_FALSE );
    else
       return( S_OK );
}
```

Файл server.def:

```
;
; Server.def : определение модуля для DLL.
;

LIBRARY      "SERVER"
DESCRIPTION  'SERVER Windows Dynamic Link Library'
EXPORTS
    DllGetClassObject   PRIVATE
    DllCanUnloadNow     PRIVATE
```

Функция DllGetClassObject вызывается для создания экземпляра COM объекта. DllCanUnloadNow периодически вызывается Windows для освобождения памяти от неиспользуемых DLL.

Кроме того, динамическая библиотека может включать в себя еще две функции:

DllRegisterServer используется regsvr32.exe для регистрации COM-объектов в реестре,

DllUnregisterServer вызывается для удаления всех ключей реестра, относящихся к компоненту.

Шаг 4. Создание клиента.

Клиентская программа должна инициализировать библиотеку COM, создать экземпляр COM-объекта нужно класса, получить указатель на нужный интерфейс и вызвать необходимый метод:

```
/*--------------------------------------
/               Client.cpp
/-------------------------------------*/

#include <windows.h>
#include <tchar.h>
#include <iostream.h>
#include <initguid.h>
#include "imath.h"

int main( int argc, char *argv[] )
{
   SetConsoleOutputCP(1251);
   cout << "Инициализация библиотеки COM" << endl;
   if(FAILED( CoInitialize( NULL )))
   {
      cout << "Ошибка. Нельзя инициализировать COM" <<
endl;
      return -1;
   }

   char* szProgID = "Math.Component.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 ))
        {
           cout.setf( ios::hex, ios::basefield );
           cout << "Нельзя получить CLSID  для ProgID. HR = "
<< hr << endl;
           return -1;
        }

        IClassFactory* pCF;
        hr = CoGetClassObject( clsid,
                               CLSCTX_INPROC,
                               NULL,
                               IID_IClassFactory,
                               (void**) &pCF );
        if ( FAILED( hr ))
        {
           cout.setf( ios::hex, ios::basefield );
           cout << "Не удалось создать объект фабрики. HR = "
<< hr << endl;
           return -1;
        }

        IUnknown* pUnk;
        hr   =   pCF->CreateInstance(   NULL,   IID_IUnknown,
(void**) &pUnk );
        pCF->Release();
        if ( FAILED( hr ))
        {
           cout.setf( ios::hex, ios::basefield );
           cout << "Не удалось создать экземпляр сервера. HR
= " << hr << endl;
           return -1;
        }
        cout << "Экземпляр сервера создан" << endl;
        IMath* pMath = NULL;
        hr = pUnk->QueryInterface( IID_IMath, (LPVOID*)&pMath
);
        pUnk->Release();
        if ( FAILED( hr ))
        {
```

```
            cout << "QueryInterface() для IMath не выполнен"
<< endl;
            return -1;
        }
        long result;
        pMath->Multiply( 100, 8, &result );
        cout << "100 * 8 = " << result << endl;

        pMath->Subtract( 1000, 333, &result );
        cout << "1000 - 333 = " << result << endl;

        cout << "Освобождаем интерфейс" << endl;
        pMath->Release();
        cout << "Освобождаем библиотеку COM" << endl;
        CoUninitialize();
        return 0;
    }
```

Шаг 5. Регистрация компонента в реестре.

Создайте текстовый файл и сохраните его с расширением *.reg. Внесите в него следующий текст, предварительно отредактировав его, указав точный путь к вашему COM компоненту.