**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

###### ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

###### КЕМЕРОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

**ИНСТИТУТ ЦИФРЫ**

**ОТЧЁТ**

**О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

«Спецификация COM»

Студентки 2 курса, ФИТ-211 группы

**Колесник Полины Олеговны**

Направление 02.03.02 – «Фундаментальная информатика и информационные технологии»

Руководитель:

Доцент Иванов К. С.

Работа защищена

« »

“ ” 2023 г.

Кемерово 2023 г.

**ОТЧЁТ О ПРОДЕЛАННОЙ РАБОТЕ**

**Файл client.cpp**

#include <objbase.h>

#include <stdio.h>

#include <iostream>

#include "Interfaces.h"

using namespace std;

// typedef HRESULT \_\_stdcall (\*GetClassObjectType) (const CLSID& clsid, const IID& iid, void\*\* ppv);

int main() {

    printf("Client::Main::Start\n");

    printf("Client::Main::GetClassObject CServer&IClassFactory\_\n" );

    try {

        // GetClassObjectType GetClassObject;

        // HINSTANCE h;

        // h = LoadLibrary("./build/manager/main.dll");

        // if (!h) {

        //  throw "No manager";

        // }

        // GetClassObject = (GetClassObjectType)GetProcAddress(h,"GetClassObject");

        // if (!GetClassObject)

        // {

        //  throw "No manager method";

        // }

        CoInitialize(NULL);

        // {91A42CAA-5777-4E80-934E-07DE64502FD6}

        const CLSID CLSID\_Server = {0x91A42CAA,0x5777,0x4E80,{0x93,0x4E,0x07,0xDE,0x64,0x50,0x2F,0xD6}};

        const CLSID CLSID\_Server\_2 = {0x91A42CBA,0x2577,0x4E80,{0x93,0x4E,0x07,0xDE,0x64,0x50,0x2F,0xD7}};

        IClassFactory\* pCF = NULL;

        HRESULT resFactory = CoGetClassObject(CLSID\_Server\_2,CLSCTX\_INPROC\_SERVER,NULL,IID\_IClassFactory,(void\*\*)&pCF);

        if (!(SUCCEEDED(resFactory))) {

            throw "Client::Main::No factory";

        }

        IGet\_Array\* pGA = NULL;

        HRESULT resInstance = pCF->CreateInstance(NULL,IID\_IGet\_Array,(void\*\*)&pGA);

        if (!(SUCCEEDED(resInstance))) {

            throw "Client::Main::No instance";

        }

        printf("Client::Main::Success IGet\_Array: \n");

        pGA->InputMas1();

        pGA->InputMas2();

        printf("Client::Main::QueryInterface IGet\_Array->ISample\_Processing\n");

        ISample\_Processing\* pSP = NULL;

        HRESULT resQuery = pGA->QueryInterface(IID\_ISample\_Processing,(void\*\*)&pSP);

        if (!(SUCCEEDED(resQuery))) {

          throw "Client::Main::No query";

        }

        ISumma\* is = NULL;

        resQuery = pGA->QueryInterface(IID\_ISumma,(void\*\*)&is);

        if (!(SUCCEEDED(resQuery))) {

            throw "Client::Main::No query\_2";

        }

        printf("Client::Main::Success ISample\_Processing:\n");

        pSP->Sample\_Average();

        pSP->Sample\_Variance();

        pSP->Corrected\_Sample\_Variance();

        is->summ();

        pSP->Release();

        pGA->Release();

        is->Release();

        pCF->Release();

    }

    catch (const char\* str) {

        printf("Main::Error: \n");

        printf(str);

        printf("\n");

    }

    catch (...) {

        printf("Main::Error::Unknown\n");

    }

    CoUninitialize();

    printf("Main::Finish\n");

    return 0;

}

**Файл Interfaces.h**

#ifndef Interfaces\_H\_INCLUDED

#define Interfaces\_H\_INCLUDED

#include <windows.h>

const IID IID\_IUnknown = {0x00000001,0x0000,0x0000,{0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x46}};

const IID IID\_IClassFactory = {0x00000002,0x0000,0x0000,{0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x46}};

const IID IID\_IGet\_Array = {0x00000101,0x0000,0x0000,{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}};

const IID IID\_ISample\_Processing = {0x00000102,0x0000,0x0000,{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}};

const IID IID\_ISumma = {0x00000103,0x0000,0x0000,{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}};

class ISample\_Processing: public IUnknown {

    public:

     virtual HRESULT \_\_stdcall Sample\_Average()=0;  //выборочное среднее

     virtual HRESULT \_\_stdcall Sample\_Variance()=0; //выборочная дисперсия

     virtual HRESULT \_\_stdcall Corrected\_Sample\_Variance()=0; //исправленная выборочная дисперсия

};

//интерфейс с методами для считывания из файла

class IGet\_Array: public IUnknown {

    public:

     virtual HRESULT \_\_stdcall InputMas2() = 0;

     virtual HRESULT \_\_stdcall InputMas1() = 0;

};

class ISumma: public IUnknown {

    public:

     virtual void \_\_stdcall summ() = 0;

};

#endif // Interfaces\_H\_INCLUDED

**Файл Components.h**

#ifndef COMPONENTS\_H\_INCLUDED

#define COMPONENTS\_H\_INCLUDED

#include <windows.h>

void println(const char\* str);

HRESULT \_\_stdcall GetClassObject(const CLSID& clsid, const IID& iid, void\*\* ppv);

#endif // COMPONENTS\_H\_INCLUDED

**Файл Interfaces.h**

#ifndef Interfaces\_H\_INCLUDED

#define Interfaces\_H\_INCLUDED

#include <windows.h>

const IID IID\_IUnknown = {0x00000001,0x0000,0x0000,{0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x46}};

const IID IID\_IClassFactory = {0x00000002,0x0000,0x0000,{0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x46}};

const IID IID\_IGet\_Array = {0x00000101,0x0000,0x0000,{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}};

const IID IID\_ISample\_Processing = {0x00000102,0x0000,0x0000,{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}};

class ISample\_Processing: public IUnknown {

    public:

     virtual void \_\_stdcall Sample\_Average()=0;  //выборочное среднее

     virtual void \_\_stdcall Sample\_Variance()=0; //выборочная дисперсия

     virtual void \_\_stdcall Corrected\_Sample\_Variance()=0; //исправленная выборочная дисперсия

};

//интерфейс с методами для считывания из файла

class IGet\_Array: public IUnknown {

    public:

     virtual void \_\_stdcall InputMas2() = 0;

     virtual void \_\_stdcall InputMas1() = 0;

};

#endif // Interfaces\_H\_INCLUDED

**Файл main.cpp**

#include <stdio.h>

#include "Components.h"

extern "C" \_\_declspec(dllexport) HRESULT \_\_stdcall DllGetClassObject(const CLSID& clsid, const IID& iid, void\*\* ppv) {

    println("Container::DllGetClassObject");

    return GetClassObject(clsid,iid,ppv);

}

BOOL APIENTRY DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved) {

    switch (fdwReason) {

        case DLL\_PROCESS\_ATTACH:

            // attach to process

            // return FALSE to fail DLL load

            println("Container\_Simple::DLL\_PROCESS\_ATTACH");

            break;

        case DLL\_PROCESS\_DETACH:

            // detach from process

            println("Container\_Simple::DLL\_PROCESS\_DETACH");

            break;

        case DLL\_THREAD\_ATTACH:

            // attach to thread

            break;

        case DLL\_THREAD\_DETACH:

            // detach from thread

            break;

    }

    return TRUE; // succesful

}

**Файл server.cpp**

#include <iostream>

#include <fstream>

#include <math.h>

#include "Components.h"

#include "server.h"

using namespace std;

Server::Server() {

  fRefCount = 0;

  volume = 0;

  Sample\_average = 0;

  println("Server::Constructor");

}

Server::~Server() {

  println( "Server::Destructor");

}

HRESULT \_\_stdcall Server::QueryInterface(const IID& iid, void\*\* ppv) {

    println("Server::QueryInterface");

    if (iid==IID\_IUnknown) {

      \*ppv = (IUnknown\*)(ISample\_Processing\*)this;

    }

    else if (iid == IID\_ISample\_Processing) {

      \*ppv = static\_cast<ISample\_Processing\*>(this);

    }

    else if (iid == IID\_IGet\_Array) {

      \*ppv = (IGet\_Array\*)this;

    }

    else {

      \*ppv = NULL;

      return E\_NOINTERFACE;

    }

    this->AddRef();

    return S\_OK;

}

ULONG \_\_stdcall Server::AddRef() {

   println("Server::AddRef");

   fRefCount++;

   cout << "Server::Current references: " << fRefCount << endl;

   return fRefCount;

}

ULONG \_\_stdcall Server::Release() {

   println("Server::Relese");

   fRefCount--;

   cout << "Server::Current references: " << fRefCount << endl;

   if (fRefCount==0) {delete this;}

   return fRefCount;

}

//----------------------------------------------------------

//методы интерфайса IGet\_Array

void \_\_stdcall Server::InputMas2() {

  println("Server::InputMas2");

  fstream f;

  f.open("GetX.txt");

  if(f.is\_open()) {

    for (int i = 0; i < 5; i++) {

      f >> x[i];

    }

  }

  else {

    println("No file opened");

  }

  f.close();

}

void \_\_stdcall Server::InputMas1() {

  println("Server::InputMas1");

  fstream f;

  f.open("GetN.txt");

  if(f.is\_open()) {

    for (int i = 0; i < 5; i++) {

      f >> n[i];

    }

  }

  else {

    println("No file opened");

  }

  f.close();

}

//-------------------------------------------------------

//методы интерфайса ISample\_Processing

void \_\_stdcall Server::Sample\_Average() {

  println("Server::Sample\_Average");

  float sum = 0;

  for(int i = 0; i < 5; i++) {

    volume += n[i];

    sum += x[i] \* n[i];

  }

  Sample\_average = round(sum / volume \* 100) / 100;

  cout<< "Server::Sample\_Average = " << Sample\_average << endl << endl;

}

void \_\_stdcall Server::Sample\_Variance() {

  println("Server::Sample\_Variance");

  float sum = 0;

  for(int i=0; i< 5; i++) {

    sum += pow((x[i]- Sample\_average), 2) \* n[i];

  }

  cout << "Server::Sample\_Variance = " <<  round(sum / volume \* 100) / 100 << endl;

}

void \_\_stdcall Server::Corrected\_Sample\_Variance() {

  println("Server::Corrected\_Sample\_Variance");

  float sum = 0;

  for(int i=0; i< 5; i++) {

    sum += pow((x[i]- Sample\_average), 2) \* n[i];

  }

  cout << "Server::Corrected\_Sample\_Variance = " << round(sum / (volume-1) \* 100) / 100 << endl << endl;

}

//--------------------------------------------------------------

ServerFactory::ServerFactory() {

  println("ServerFactory::Constructor");

  fRefCount = 0;

}

ServerFactory::~ServerFactory() {

  println("ServerFactory::Destructor");

}

HRESULT \_\_stdcall ServerFactory::QueryInterface(const IID& iid, void\*\* ppv) {

  println("ServerFactory::QueryInterface");

  if (iid==IID\_IUnknown) {

    \*ppv = (IUnknown\*)(IClassFactory\*)(this);

  }

  else if (iid == IID\_IClassFactory) {

    \*ppv = (void\*\*)(IClassFactory\*)(this);

  }

  else {

    \*ppv = NULL;

    return E\_NOINTERFACE;

  }

  this->AddRef();

  return S\_OK;

}

ULONG \_\_stdcall ServerFactory::AddRef() {

  println("ServerFactory::AddRef");

  fRefCount++;

  cout << "ServerFactory::Current references: " << fRefCount << endl;

  return fRefCount;

}

ULONG \_\_stdcall ServerFactory::Release() {

  println("ServerFactory::Relese");

  fRefCount--;

  cout << "ServerFactory::Current references: " << fRefCount << endl;

  if (fRefCount==0) { delete this;}

  return fRefCount;

}

HRESULT  \_\_stdcall ServerFactory::CreateInstance(IUnknown\* pUnknownOuter, const IID& iid, void\*\* ppv) {

  println("ServerFactory::CreateInstance");

  if (pUnknownOuter!=NULL) {

    return E\_NOTIMPL;

  }

  Server\* p = new Server();

  return p->QueryInterface(iid,ppv);

}

HRESULT \_\_stdcall ServerFactory::LockServer(BOOL bLock) {

  println("ServerFactory::LockServer");

  return S\_OK;

}

void println(const char\* str) {

  printf(str);

  printf("\n");

}

const CLSID CLSID\_Server = {0x91A42CAA,0x5777,0x4E80,{0x93,0x4E,0x07,0xDE,0x64,0x50,0x2F,0xD6}};

HRESULT \_\_stdcall GetClassObject(const CLSID& clsid, const IID& iid, void\*\* ppv) {

  println("Component::GetClassObject");

  if (clsid==CLSID\_Server) {

    ServerFactory\* fa  = new ServerFactory();

    return fa->QueryInterface(iid,ppv);

  }

  else {

    \*ppv = NULL;

    return E\_NOTIMPL;

  }

}

**Файл server.h**

#ifndef SERVER\_H\_INCLUDED

#define SERVER\_H\_INCLUDED

#include "interfaces.h"

//компонент

class Server: public ISample\_Processing, IGet\_Array {

    private: //обработка выборки

      int fRefCount;

     int x[5];

     int n[5];

     float volume;

     float Sample\_average;

    public:

     Server();

     ~Server();

     virtual HRESULT \_\_stdcall QueryInterface(const IID& iid, void\*\* ppv);

     virtual ULONG \_\_stdcall AddRef();

     virtual ULONG \_\_stdcall Release();

     virtual void \_\_stdcall Sample\_Average();

     virtual void \_\_stdcall Sample\_Variance();

     virtual void \_\_stdcall Corrected\_Sample\_Variance();

     virtual void \_\_stdcall InputMas1(); //n

     virtual void \_\_stdcall InputMas2(); //x

};

class ServerFactory: public IClassFactory {

    int fRefCount;

    public:

     ServerFactory();

     ~ServerFactory();

     virtual HRESULT \_\_stdcall QueryInterface(const IID& iid, void\*\* ppv);

     virtual ULONG \_\_stdcall AddRef();

     virtual ULONG \_\_stdcall Release();

     virtual HRESULT \_\_stdcall CreateInstance(IUnknown\* pUnknownOuter, const IID& iid, void\*\* ppv);

     virtual HRESULT \_\_stdcall LockServer(BOOL bLock);

};

#endif // SERVER\_H\_INCLUDED

**Файл Components.h**

#ifndef COMPONENTS\_H\_INCLUDED

#define COMPONENTS\_H\_INCLUDED

#include <windows.h>

void println(const char\* str);

HRESULT \_\_stdcall GetClassObject(const CLSID& clsid, const IID& iid, void\*\* ppv);

#endif // COMPONENTS\_H\_INCLUDED

**Файл Interfaces.h**

#ifndef Interfaces\_H\_INCLUDED

#define Interfaces\_H\_INCLUDED

#include <windows.h>

const IID IID\_IUnknown = {0x00000001,0x0000,0x0000,{0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x46}};

const IID IID\_IClassFactory = {0x00000002,0x0000,0x0000,{0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x46}};

const IID IID\_IGet\_Array = {0x00000101,0x0000,0x0000,{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}};

const IID IID\_ISample\_Processing = {0x00000102,0x0000,0x0000,{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}};

const IID IID\_ISumma = {0x00000103,0x0000,0x0000,{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}};

class ISample\_Processing: public IUnknown {

    public:

    virtual void \_\_stdcall Sample\_Average() = 0;  //выборочное среднее

     virtual void \_\_stdcall Sample\_Variance() = 0; //выборочная дисперсия

     virtual void \_\_stdcall Corrected\_Sample\_Variance() = 0; //исправленная выборочная дисперсия };

};

class IGet\_Array: public IUnknown {

    public:

     virtual void \_\_stdcall InputMas2() = 0;

     virtual void \_\_stdcall InputMas1() = 0;

};

class ISumma: public IUnknown {

    public:

     virtual void \_\_stdcall summ() = 0;

};

#endif // Interfaces\_H\_INCLUDED

**Файл main.cpp**

#include <stdio.h>

#include "Components.h"

extern "C" \_\_declspec(dllexport) HRESULT \_\_stdcall DllGetClassObject(const CLSID& clsid, const IID& iid, void\*\* ppv) {

    println("Container::DllGetClassObject");

    return GetClassObject(clsid,iid,ppv);

}

BOOL APIENTRY DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved) {

    switch (fdwReason) {

        case DLL\_PROCESS\_ATTACH:

            // attach to process

            // return FALSE to fail DLL load

            println("Container\_Super::DLL\_PROCESS\_ATTACH");

            break;

        case DLL\_PROCESS\_DETACH:

            // detach from process

            println("Container\_Super::DLL\_PROCESS\_DETACH");

            break;

        case DLL\_THREAD\_ATTACH:

            // attach to thread

            break;

        case DLL\_THREAD\_DETACH:

            // detach from thread

            break;

    }

    return TRUE; // succesful

}

**Файл server.cpp**

#include "server.h"

#include <objbase.h>

#include <iostream>

#include <fstream>

#include <math.h>

using namespace std;

Server\_2::Server\_2() {

   a = 5;

   b = 8;

   println("Server\_2::Server\_2");

   fRefCount = 0;

   println("Server\_2::Constructor");

   CoInitialize(NULL);

//   HINSTANCE h;

//   h=LoadLibrary("./build/manager/main.dll");

//   if (!h)

//   {

//      throw "No manager";

//   }

//   CreateInstanceType CreateInstance = (CreateInstanceType) GetProcAddress(h,"CreateInstance");

//   if (!CreateInstance)

//   {

//      throw "No manager method";

//   }

  //Getting manager method (End)

   const CLSID CLSID\_Server = {0x91A42CAA,0x5777,0x4E80,{0x93,0x4E,0x07,0xDE,0x64,0x50,0x2F,0xD6}};

   HRESULT resFactory = CoGetClassObject(CLSID\_Server,CLSCTX\_INPROC\_SERVER,NULL,IID\_IClassFactory,(void\*\*)&pCF);

   if (!(SUCCEEDED(resFactory))) {

      throw "Server\_2::No factory";

   }

   HRESULT resInstance = pCF->CreateInstance(NULL,IID\_IGet\_Array,(void\*\*)&ig\_Simple);

   if (!(SUCCEEDED(resInstance))) {

      throw "Server\_2::No instance";

   }

   HRESULT resY = ig\_Simple->QueryInterface(IID\_ISample\_Processing,(void\*\*)&is\_Simple);

   if (!(SUCCEEDED(resY))) {

      throw "Server\_2::No query";

   }

  //Injecting component (End)

}

Server\_2::~Server\_2() {

   println( "Server\_2::Destructor");

   ig\_Simple->Release();

   is\_Simple->Release();

   pCF->Release();

   CoUninitialize();

}

HRESULT \_\_stdcall Server\_2::QueryInterface(const IID& iid, void\*\* ppv) {

   println("Server\_2::QueryInterface");

   if (iid==IID\_IUnknown) {

    \*ppv = (IUnknown\*)(ISample\_Processing\*)this;

   }

   else if (iid == IID\_ISample\_Processing) {

    \*ppv = static\_cast<ISample\_Processing\*>(this);

   }

   else if (iid == IID\_IGet\_Array) {

    \*ppv = (IGet\_Array\*)this;

   }

   else if (iid == IID\_ISumma) {

      \*ppv = (ISumma\*)this;

   }

   else {

     \*ppv = NULL;

     return E\_NOINTERFACE;

   }

   this->AddRef();

   return S\_OK;

}

ULONG \_\_stdcall Server\_2::AddRef() {

   println("Server\_2::AddRef");

   fRefCount++;

   cout << "Server\_2::Current references: " << fRefCount << endl;

   return fRefCount;

}

ULONG \_\_stdcall Server\_2::Release() {

   println("Server\_2::Release");

   fRefCount--;

   cout << "Server\_2::Current references: " << fRefCount << endl;

   if (fRefCount==0) {delete this;}

   return fRefCount;

}

//методы компонента

void \_\_stdcall Server\_2::InputMas1() {

  println("Server\_2::InputMas1:Full delegating to the server component ");

  ig\_Simple->InputMas1();

}

void \_\_stdcall Server\_2::InputMas2() {

  println("Server\_2::InputMas2:Full delegating to the server component");

  ig\_Simple->InputMas2();

}

void \_\_stdcall Server\_2::Sample\_Average() {

  println("Server\_2::Sample\_Average:Full delegating to the server component");

  is\_Simple->Sample\_Average();

}

void \_\_stdcall Server\_2::Sample\_Variance() {

  println("Server\_2::Sample\_Variance:Full delegating to the server component");

  is\_Simple->Sample\_Variance();

}

void \_\_stdcall Server\_2::Corrected\_Sample\_Variance() {

  println("Server\_2::Corrected\_Sample\_Variance:Full delegating to the server component");

  is\_Simple->Corrected\_Sample\_Variance();

}

void \_\_stdcall Server\_2::summ(){

   cout<< "Server\_2::summ = " << a + b << endl;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ServerFactory\_2::ServerFactory\_2() {

   println("ServerFactory\_2::Constructor");

   fRefCount = 0;

}

ServerFactory\_2::~ServerFactory\_2() {

  println("ServerFactory\_2::Destructor");

}

HRESULT \_\_stdcall ServerFactory\_2::QueryInterface(const IID& iid, void\*\* ppv) {

   println("ServerFactory\_2::QueryInterface");

   if (iid==IID\_IUnknown) {

    \*ppv = (IUnknown\*)(IClassFactory\*)this;

   }

   else if (iid == IID\_IClassFactory) {

    \*ppv = (IClassFactory\*)this;

   }

   else {

     \*ppv = NULL;

     return E\_NOINTERFACE;

   }

   this->AddRef();

   return S\_OK;

}

ULONG \_\_stdcall ServerFactory\_2::AddRef() {

   println("ServerFactory\_2::AddRef");

   fRefCount++;

   cout << "ServerFactory\_2::Current references: " << fRefCount << endl;

   return fRefCount;

}

ULONG \_\_stdcall ServerFactory\_2::Release()

{

   println("ServerFactory\_2::Release");

   fRefCount--;

   cout << "ServerFactory\_2::Current references: " << fRefCount << endl;

   if (fRefCount==0) {delete this;}

   return fRefCount;

}

HRESULT  \_\_stdcall ServerFactory\_2::CreateInstance(IUnknown\* pUnknownOuter, const IID& iid, void\*\* ppv) {

   println("ServerFactory\_2::CreateInstance");

   if (pUnknownOuter!=NULL) {

      return E\_NOTIMPL;

   }

   Server\_2\* p = new Server\_2();

   HRESULT res = p->QueryInterface(iid,ppv);

   return res;

}

HRESULT \_\_stdcall ServerFactory\_2::LockServer(BOOL bLock) {

  println("ServerFactory\_2::LockServer");

  return S\_OK;

}

void println(const char\* str) {

    printf(str);

    printf("\n");

}

const CLSID CLSID\_Server\_2= {0x91A42CBA,0x2577,0x4E80,{0x93,0x4E,0x07,0xDE,0x64,0x50,0x2F,0xD7}};

HRESULT \_\_stdcall GetClassObject(const CLSID& clsid, const IID& iid, void\*\* ppv) {

   println("Component Server\_2::GetClassObject");

   if (clsid==CLSID\_Server\_2) {

      try {

         ServerFactory\_2\* fa  = new ServerFactory\_2();

         return fa->QueryInterface(iid,ppv);

      }

      catch(...) {

         \*ppv = NULL;

         return E\_UNEXPECTED;

      }

   }

   else {

     \*ppv = NULL;

     return E\_NOTIMPL;

   }

}

**Файл server.h**

#ifndef SERVER\_H\_INCLUDED

#define SERVER\_H\_INCLUDED

#include "interfaces.h"

#include "Components.h"

#include <stdio.h>

#include <objbase.h>

//компонент

class Server\_2: public ISample\_Processing, IGet\_Array, ISumma  {

    private:

     int a;

     int b;

     int fRefCount;

     IGet\_Array\* ig\_Simple = NULL;

     ISample\_Processing\* is\_Simple = NULL;

     IClassFactory\* pCF = NULL;

    public:

     Server\_2();

     virtual ~Server\_2();

     virtual void \_\_stdcall InputMas2();

     virtual void \_\_stdcall InputMas1();

     virtual void \_\_stdcall Sample\_Average();  //выборочное среднее

     virtual void \_\_stdcall Sample\_Variance(); //выборочная дисперсия

     virtual void \_\_stdcall Corrected\_Sample\_Variance(); //исправленная выборочная дисперсия

     virtual void \_\_stdcall summ();

     virtual HRESULT \_\_stdcall QueryInterface(const IID& iid, void\*\* ppv);

     virtual ULONG \_\_stdcall AddRef();

     virtual ULONG \_\_stdcall Release();

};

class ServerFactory\_2: public IClassFactory {

     int fRefCount;

    public:

     ServerFactory\_2();

     ~ServerFactory\_2();

     virtual HRESULT \_\_stdcall QueryInterface(const IID& iid, void\*\* ppv);

     virtual ULONG \_\_stdcall AddRef();

     virtual ULONG \_\_stdcall Release();

     virtual HRESULT \_\_stdcall CreateInstance(IUnknown\* pUnknownOuter, const IID& iid, void\*\* ppv);

     virtual HRESULT \_\_stdcall LockServer(BOOL bLock);

};

#endif // SERVER\_H\_INCLUDED

**Файл clsidSimple32.reg**

Windows Registry Editor Version 5.00

[HKEY\_LOCAL\_MACHINE\SOFTWARE\Classes\CLSID\{91A42CAA-5777-4E80-934E-07DE64502FD6}]

[HKEY\_LOCAL\_MACHINE\SOFTWARE\Classes\WOW6432Node\CLSID\{91A42CAA-5777-4E80-934E-07DE64502FD6}\InprocServer32]

@="C:\\Users\\2003k\\Desktop\\2.4\\4\_term\\COM\\Сomponent\\build\\server\\simple.dll"

**Файл clsidSimple64.reg**

Windows Registry Editor Version 5.00

[HKEY\_LOCAL\_MACHINE\SOFTWARE\Classes\WOW6432Node\CLSID\{91A42CAA-5777-4E80-934E-07DE64502FD6}]

[HKEY\_LOCAL\_MACHINE\SOFTWARE\Classes\WOW6432Node\CLSID\{91A42CAA-5777-4E80-934E-07DE64502FD6}\InprocServer32]

@="C:\\Users\\2003k\\Desktop\\2.4\\4\_term\\COM\\Сomponent\\build\\server\\simple.dll"

**Файл clsidSuper32.reg**

Windows Registry Editor Version 5.00

[HKEY\_LOCAL\_MACHINE\SOFTWARE\Classes\CLSID\{91A42CBA-2577-4E80-934E-07DE64502FD7}]

[HKEY\_LOCAL\_MACHINE\SOFTWARE\Classes\CLSID\{91A42CBA-2577-4E80-934E-07DE64502FD7}\InprocServer32]

@="C:\\Users\\2003k\\Desktop\\2.4\\4\_term\\COM\\Сomponent\\build\\server\\super.dll"

**Файл clsidSuper64.reg**

Windows Registry Editor Version 5.00

[HKEY\_LOCAL\_MACHINE\SOFTWARE\Classes\WOW6432Node\CLSID\{91A42CBA-2577-4E80-934E-07DE64502FD7}]

[HKEY\_LOCAL\_MACHINE\SOFTWARE\Classes\WOW6432Node\CLSID\{91A42CBA-2577-4E80-934E-07DE64502FD7}\InprocServer32]

@="C:\\Users\\2003k\\Desktop\\2.4\\4\_term\\COM\\Сomponent\\build\\server\\super.dll"

 

