Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Отчет по лабораторной работе №5

«РАБОТА С COM ЧЕРЕЗ CLSID на примере EXCEL»

Студент: Шавейко А.А.

Пахолко А.С.

ФИТ 3 курс 6 группа

Преподаватель: Герман Ю.О.

Минск 2019

**Задание.**

Построить dll для запуска EXCEL. Использовать данную программу для тела функции dll.

|  |
| --- |
| // lab\_05.cpp : Определяет экспортированные функции для приложения DLL.  //  #include "stdafx.h"  #include<oaidl.h>  #include <windows.h> // Program Demonstrates Late Bound OLE COM Access To MS Excel Spreadsheet Using C++.  #include <tchar.h> // "Hello, World! Is Written To Cell A1 Of Sheet #1 In Visible Workbook. IDispatch  #include <cstdio> // Interface Using GetIDsOfNames() And Invoke() Used Throughout.  const CLSID CLSID\_XLApplication = { 0x00024500,0x0000,0x0000,{ 0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x46 } }; // CLSID of Excel  const IID IID\_Application = { 0x000208D5,0x0000,0x0000,{ 0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x46 } }; // IID of \_Application  extern "C++" \_\_declspec(dllexport) void Call\_excell()  {  DISPPARAMS NoArgs = { NULL,NULL,0,0 }; // This variable is used in easiest Invoke() call when the method has no parameters. When  IDispatch\* pXLApp = NULL; // using the IDispatch interface in conjunction with Invoke() method parameters must be loaded  DISPPARAMS DispParams; // into a DISPPARAMS struct. The actual parameters are loaded into VARIANTs, and one of the  VARIANT CallArgs[1]; // members of the DISPPARAMS struct is a pointer to the array of VARIANT. The other members  VARIANT vResult; // of the DISPARAMS struct tell Invoke() how many parameters are being passed, as well as other  DISPID dispid; // specifics such as the type of the call (propput, propget, etc.).  HRESULT hr;  HRESULT hr2;  CoInitialize(NULL);  hr = CoCreateInstance(CLSID\_XLApplication, NULL, CLSCTX\_LOCAL\_SERVER, IID\_Application, (void\*\*)&pXLApp);  if (SUCCEEDED(hr))  {  OLECHAR\* szVisible = (OLECHAR\*)L"Visible";  hr = pXLApp->GetIDsOfNames(IID\_NULL, &szVisible, 1, GetUserDefaultLCID(), &dispid);  if (SUCCEEDED(hr))  {  VariantInit(&CallArgs[0]);  CallArgs[0].vt = VT\_BOOL;  CallArgs[0].boolVal = TRUE;  DISPID dispidNamed = DISPID\_PROPERTYPUT;  DispParams.rgvarg = CallArgs;  DispParams.rgdispidNamedArgs = &dispidNamed;  DispParams.cArgs = 1;  DispParams.cNamedArgs = 1;  VariantInit(&vResult); // Call or Invoke \_Application::Visible(true);  hr = pXLApp->Invoke(dispid, IID\_NULL, LOCALE\_USER\_DEFAULT, DISPATCH\_PROPERTYPUT, &DispParams, &vResult, NULL, NULL);  OLECHAR\* szWorkbooks = (OLECHAR\*)L"Workbooks";  hr = pXLApp->GetIDsOfNames(IID\_NULL, &szWorkbooks, 1, GetUserDefaultLCID(), &dispid);  if (SUCCEEDED(hr))  {  IDispatch\* pXLBooks = NULL; // Get Workbooks Collection  VariantInit(&vResult); // Invoke \_Application::Workbooks(&pXLBooks) << returns IDispatch\*\* of Workbooks Collection  hr = pXLApp->Invoke(dispid, IID\_NULL, LOCALE\_USER\_DEFAULT, DISPATCH\_PROPERTYGET, &NoArgs, &vResult, NULL, NULL);  if (SUCCEEDED(hr))  {  pXLBooks = vResult.pdispVal;  IDispatch\* pXLBook = NULL; // Try to add Workbook  OLECHAR\* szAdd = (OLECHAR\*)L"Add";  hr = pXLBooks->GetIDsOfNames(IID\_NULL, &szAdd, 1, GetUserDefaultLCID(), &dispid);  if (SUCCEEDED(hr))  {  VariantInit(&vResult); // Invoke Workbooks::Add(&Workbook) << returns IDispatch\*\* of Workbook Object  hr = pXLBooks->Invoke(dispid, IID\_NULL, LOCALE\_USER\_DEFAULT, DISPATCH\_METHOD | DISPATCH\_PROPERTYGET, &NoArgs, &vResult, NULL, NULL);  if (SUCCEEDED(hr))  {  pXLBook = vResult.pdispVal;  OLECHAR\* szActiveSheet = (OLECHAR\*)L"ActiveSheet";  hr = pXLApp->GetIDsOfNames(IID\_NULL, &szActiveSheet, 1, GetUserDefaultLCID(), &dispid);  if (SUCCEEDED(hr))  {  IDispatch\* pXLSheet = NULL; // Try To Get ActiveSheet  IDispatch\* pXLSheet1 = NULL; // Try To Get ActiveSheet  VariantInit(&vResult); // Invoke \_Application::ActiveSheet(&pXLSheet); << ret IDispatch\*\* to Worksheet (Worksheet)  hr = pXLApp->Invoke(dispid, IID\_NULL, LOCALE\_USER\_DEFAULT, DISPATCH\_PROPERTYGET, &NoArgs, &vResult, NULL, NULL);  if (SUCCEEDED(hr))  {  pXLSheet = vResult.pdispVal;  OLECHAR\* szRange = (OLECHAR\*)L"Range";  hr = pXLSheet->GetIDsOfNames(IID\_NULL, &szRange, 1, GetUserDefaultLCID(), &dispid);  if (SUCCEEDED(hr))  {  IDispatch\* pXLRange = NULL;  IDispatch\* pXLRange2 = NULL;  VariantInit(&vResult);  CallArgs[0].vt = VT\_BSTR,  CallArgs[0].bstrVal = SysAllocString(L"B3");  DispParams.rgvarg = CallArgs;  DispParams.rgdispidNamedArgs = 0;  DispParams.cArgs = 1; // Try to get Range  DispParams.cNamedArgs = 0; // Invoke \_Worksheet::Range("A1") << returns IDispatch\*\* to dispinterface Range  hr = pXLSheet->Invoke(dispid, IID\_NULL, LOCALE\_USER\_DEFAULT, DISPATCH\_PROPERTYGET, &DispParams, &vResult, NULL, NULL);  hr2 = hr;  if (SUCCEEDED(hr))  {  pXLRange = vResult.pdispVal;  pXLRange2 = pXLRange;  OLECHAR\* szValue = (OLECHAR\*)L"Value";  hr = pXLRange->GetIDsOfNames(IID\_NULL, &szValue, 1, GetUserDefaultLCID(), &dispid);  if (SUCCEEDED(hr))  {  VariantClear(&CallArgs[0]);  CallArgs[0].vt = VT\_BSTR;  CallArgs[0].bstrVal = SysAllocString(L"Hellow");  DispParams.rgvarg = CallArgs;  DispParams.rgdispidNamedArgs = &dispidNamed;  DispParams.cArgs = 1;  DispParams.cNamedArgs = 1;  hr = pXLRange->Invoke(dispid, IID\_NULL, LOCALE\_USER\_DEFAULT, DISPATCH\_PROPERTYPUT, &DispParams, NULL, NULL, NULL);  }  }  if (SUCCEEDED(hr))  {  pXLRange = vResult.pdispVal;  OLECHAR\* szInterior = (OLECHAR\*)L"Interior";  hr = pXLRange->GetIDsOfNames(IID\_NULL, &szInterior, 1, GetUserDefaultLCID(), &dispid);  if (SUCCEEDED(hr))  {  IDispatch\* pXLInterior = NULL;  VariantInit(&vResult);  hr = pXLRange->Invoke(dispid, IID\_NULL, LOCALE\_USER\_DEFAULT, DISPATCH\_PROPERTYGET, &NoArgs, &vResult, NULL, NULL);  if (SUCCEEDED(hr))  {  pXLInterior = vResult.pdispVal;  OLECHAR\* szRange = (OLECHAR\*)L"ColorIndex";  hr = pXLInterior->GetIDsOfNames(IID\_NULL, &szRange, 1, GetUserDefaultLCID(), &dispid);  if (SUCCEEDED(hr))  {  VariantClear(&CallArgs[0]);  CallArgs[0].vt = VT\_I4;  CallArgs[0].intVal = 36;  DispParams.rgvarg = CallArgs;  DispParams.rgdispidNamedArgs = &dispidNamed;  DispParams.cArgs = 1;  DispParams.cNamedArgs = 1;  hr = pXLInterior->Invoke(dispid, IID\_NULL, LOCALE\_USER\_DEFAULT, DISPATCH\_PROPERTYPUT, &DispParams, NULL, NULL, NULL);  }  }  }  }  if (SUCCEEDED(hr))  {  OLECHAR\* szInterior = (OLECHAR\*)L"Font";  hr = pXLRange2->GetIDsOfNames(IID\_NULL, &szInterior, 1, GetUserDefaultLCID(), &dispid);  if (SUCCEEDED(hr))  {  IDispatch\* pXLInterior = NULL;  VariantInit(&vResult);  hr = pXLRange2->Invoke(dispid, IID\_NULL, LOCALE\_USER\_DEFAULT, DISPATCH\_PROPERTYGET, &NoArgs, &vResult, NULL, NULL);  if (SUCCEEDED(hr))  {  pXLInterior = vResult.pdispVal;  OLECHAR\* szRange = (OLECHAR\*)L"ColorIndex";  hr = pXLInterior->GetIDsOfNames(IID\_NULL, &szRange, 1, GetUserDefaultLCID(), &dispid);  if (SUCCEEDED(hr))  {  VariantClear(&CallArgs[0]);  CallArgs[0].vt = VT\_I4;  CallArgs[0].intVal = 5;  DispParams.rgvarg = CallArgs;  DispParams.rgdispidNamedArgs = &dispidNamed;  DispParams.cArgs = 1;  DispParams.cNamedArgs = 1;  hr = pXLInterior->Invoke(dispid, IID\_NULL, LOCALE\_USER\_DEFAULT, DISPATCH\_PROPERTYPUT, &DispParams, NULL, NULL, NULL);  }  }  ////  if (SUCCEEDED(hr))  {  pXLInterior = vResult.pdispVal;  OLECHAR\* szRange = (OLECHAR\*)L"Name";  hr = pXLInterior->GetIDsOfNames(IID\_NULL, &szRange, 1, GetUserDefaultLCID(), &dispid);  if (SUCCEEDED(hr))  {  VariantClear(&CallArgs[0]);  CallArgs[0].vt = VT\_BSTR;  CallArgs[0].bstrVal = SysAllocString(L"Wide Latin");  DispParams.rgvarg = CallArgs;  DispParams.rgdispidNamedArgs = &dispidNamed;  DispParams.cArgs = 1;  DispParams.cNamedArgs = 1;  hr = pXLInterior->Invoke(dispid, IID\_NULL, LOCALE\_USER\_DEFAULT, DISPATCH\_PROPERTYPUT, &DispParams, NULL, NULL, NULL);  pXLInterior->Release();  }  }  }  }  pXLSheet->Release();  }  }    }  pXLBook->Release();  }  pXLBooks->Release();  }  }  getchar();  }  VariantInit(&vResult); // Try to do \_Application::Close()  hr = pXLApp->Invoke(0x0000012e, IID\_NULL, LOCALE\_USER\_DEFAULT, DISPATCH\_METHOD, &NoArgs, &vResult, NULL, NULL);  pXLApp->Release();  }  CoUninitialize();  }  } |

|  |
| --- |
| //dllmain.cpp  // dllmain.cpp : Определяет точку входа для приложения DLL.  #include "stdafx.h"  BOOL APIENTRY DllMain( HMODULE hModule,  DWORD ul\_reason\_for\_call,  LPVOID lpReserved  )  {  switch (ul\_reason\_for\_call)  {  case DLL\_PROCESS\_ATTACH:  case DLL\_THREAD\_ATTACH:  case DLL\_THREAD\_DETACH:  case DLL\_PROCESS\_DETACH:  break;  }  return TRUE;  } |

|  |
| --- |
| //Aplication.cpp  // ConsoleApplication1.cpp : Этот файл содержит функцию "main". Здесь начинается и заканчивается выполнение программы.  //  #include "pch.h"  #include <iostream>  #include "Header.h"  int main()  {  Call\_excell();  } |