Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**Электронный отчёт**

По предмету “Операционные системы и системы программирования”

Студент: Савенко А.В.

ФИТ 3 курс 6 группа

Преподаватель: Герман Ю.О.

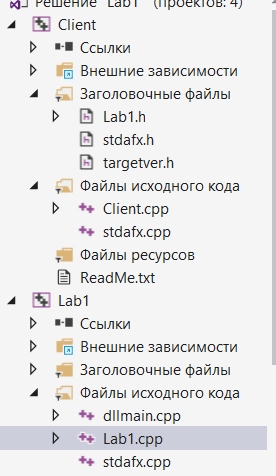
Минск 2019

**Лабораторная работа № 1. Создание и использование DLL.**

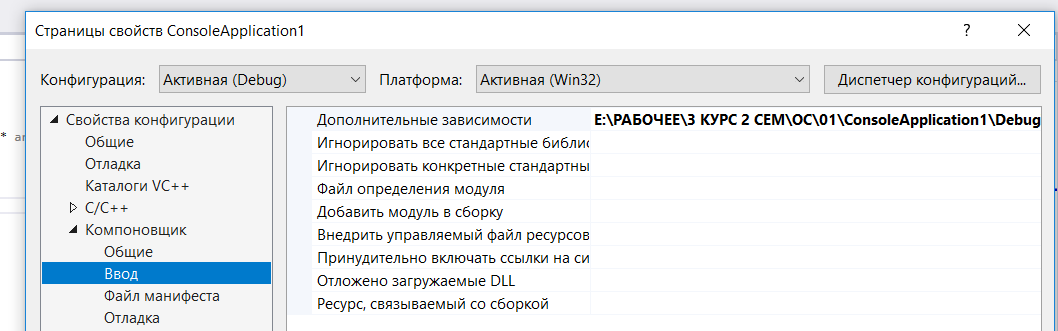
**DLL** (англ. Dynamic-link library - динамически загружаемая библиотека) - реализованные компанией Microsoft общие библиотеки в ОС Windows и OS / 2. Как правило, библиотеки имеют расширение файла \*. dll, \*. ocx (для библиотек, содержащих элементы управления ActiveX) или \*. drv (драйверы старых версий ОС). DLL может содержать код, данные и ресурсы в любой комбинации.

Под Windows динамические библиотеки хранятся в файлах с расширением \*. dll. Кроме подпрограмм, там могут также храниться другие ресурсы, например, иконки или битмапы. В коде приложения, использующего функции из динамической библиотеки, загрузки и выгрузки библиотеки должны быть прописаны непосредственно. Компилятору не требуется код функций, содержащихся в динамических библиотеках. При запуске программы, она, однако, будет искать соответствующую dll. Если такая dll не найдена на компьютере, то программа выдаст сообщение об отсутствии динамической библиотеки.

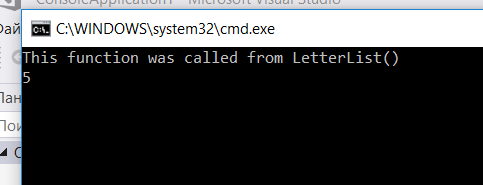
## DLL вызов функции c++



## Подключение dll файла в Visual Studio



Получаем такой результат:



У вас должно быть вот так:

\*.dll и \*.lib в папке Debug, а \*.h в проекте.



**ЗАДАНИЕ**.

1. Написать DLL- функцию для создания и записи строки в файл.
2. Написать DLL – функцию для запуска процесса на выполнение.
3. Написать DLL-функцию для проверки наличия файла в папке.

|  |
| --- |
| Client.cpp  #include "stdafx.h"  #include "Lab1.h"  #include <iostream>  #include <stdlib.h>  #include "string.h"  #include <string>  #include "stdafx.h"  using namespace std;  int \_tmain(int argc, \_TCHAR\* argv[])  {  WriteFile();  StartProcess();  CheckFile();  return 0;  } |

|  |
| --- |
| // DllMy.cpp: определяет экспортированные функции для приложения DLL.  #include "stdafx.h"  // Lab1.cpp: определяет экспортированные функции для приложения DLL.  //  #include <iostream>  #include "string.h"  #include <string>  #include <fstream>  #include <stdio.h>  #include <windows.h>  #include <ctime>  using namespace std;  extern "C++" \_\_declspec(dllexport) void WriteFile()  {  setlocale(NULL, "RUS");  HANDLE hFile;  DWORD dwBytesWritten;  char buf[50];  cout << "Ввведите строку для записи в файл: " << endl;  fgets(buf, sizeof(buf), stdin);  ofstream out;  out.open("E:\\РАБОЧЕЕ\\3 КУРС 2 СЕМ\\ОС\\Lab1\\task1.txt"); // окрываем файл для записи  if (out.is\_open())  {  out << buf << std::endl;  }  }  extern "C++" \_\_declspec(dllexport) void StartProcess()  {  STARTUPINFO cif;  ZeroMemory(&cif, sizeof(STARTUPINFO));  PROCESS\_INFORMATION pi;  TCHAR buf[] = TEXT("mspaint.exe");  if (!(CreateProcess(NULL, buf, NULL, NULL, FALSE, 0,  NULL, NULL, &cif, &pi)))  std::wcout << "Error " << '\n';  }  extern "C++" \_\_declspec(dllexport) void CheckFile()  {  WIN32\_FIND\_DATA FindFileData;  HANDLE hFind = INVALID\_HANDLE\_VALUE;  TCHAR directorySearch[] = TEXT("E:\\РАБОЧЕЕ\\3 КУРС 2 СЕМ\\ОС\\Lab1\\\*");  TCHAR filesearch[] = TEXT("E:\\РАБОЧЕЕ\\3 КУРС 2 СЕМ\\ОС\\Lab1\\task1\*.txt");  // Find the first file in the directory.  hFind = FindFirstFile(filesearch, &FindFileData);  if (hFind == INVALID\_HANDLE\_VALUE)  {  printf("Invalid file handle. Error is %u.\n", GetLastError());  }  else  {  std::wcout << "Found: " << FindFileData.cFileName << '\n';  // List all the other files in the directory.  while (FindNextFile(hFind, &FindFileData) != 0)  {  std::wcout << "Found: " << FindFileData.cFileName << '\n';  }  FindClose(hFind);  }  } |

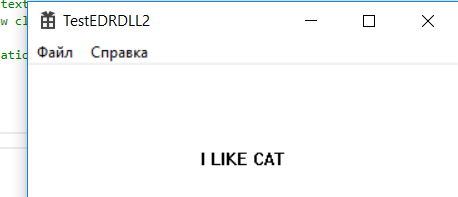
**Продолжение.**

Теперь создадим dll и подключим ее к оконному приложению. Тело (cpp) dll образует следующий код EdrLib2.cpp

|  |
| --- |
| extern "C++" \_\_declspec(dllexport) BOOL EdrCenterText(HDC hdc, PRECT prc, LPCWSTR pStr)  {  int iLength;  SIZE size;  iLength = wcslen(pStr);  GetTextExtentPoint32(hdc, pStr, iLength, &size);  return TextOut(hdc, (prc->right - prc->left - size.cx) / 2,  (prc->bottom - prc->top - size.cy) / 2,  pStr, iLength);  } |

Теперь строим клиентское оконное приложение TestEDRDLL2. Cpp:

|  |
| --- |
| // TestEDRDLL2.cpp: определяет точку входа для приложения.  //  #include "stdafx.h"  #include "TestEdrDLL2.h"  //#include "Lab1.h"  #define MAX\_LOADSTRING 100  // Global Variables:  HINSTANCE hInst; // current instance  TCHAR szTitle[MAX\_LOADSTRING]; // The title bar text  TCHAR szWindowClass[MAX\_LOADSTRING]; // the main window class name  // Forward declarations of functions included in this code module:  ATOM MyRegisterClass(HINSTANCE hInstance);  BOOL InitInstance(HINSTANCE, int);  LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);  INT\_PTR CALLBACK About(HWND, UINT, WPARAM, LPARAM);  ;  BOOL EdrCenterText(HDC hdc, PRECT prc, LPCWSTR pStr);  int APIENTRY \_tWinMain(HINSTANCE hInstance,  HINSTANCE hPrevInstance,  LPTSTR lpCmdLine,  int nCmdShow)  {  UNREFERENCED\_PARAMETER(hPrevInstance);  UNREFERENCED\_PARAMETER(lpCmdLine);  // TODO: Place code here.  MSG msg;  HACCEL hAccelTable;  // Initialize global strings  LoadString(hInstance, IDS\_APP\_TITLE, szTitle, MAX\_LOADSTRING);  LoadString(hInstance, IDC\_TESTEDRDLL2, szWindowClass, MAX\_LOADSTRING);  MyRegisterClass(hInstance);  // Perform application initialization:  if (!InitInstance(hInstance, nCmdShow))  {  return FALSE;  }  hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_TESTEDRDLL2));  // Main message loop:  while (GetMessage(&msg, NULL, 0, 0))  {  if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))  {  TranslateMessage(&msg);  DispatchMessage(&msg);  }  }  return (int)msg.wParam;  }  //  // FUNCTION: MyRegisterClass()  //  // PURPOSE: Registers the window class.  //  // COMMENTS:  //  // This function and its usage are only necessary if you want this code  // to be compatible with Win32 systems prior to the 'RegisterClassEx'  // function that was added to Windows 95. It is important to call this function  // so that the application will get 'well formed' small icons associated  // with it.  //  ATOM MyRegisterClass(HINSTANCE hInstance)  {  WNDCLASSEX wcex;  wcex.cbSize = sizeof(WNDCLASSEX);  wcex.style = CS\_HREDRAW | CS\_VREDRAW;  wcex.lpfnWndProc = WndProc;  wcex.cbClsExtra = 0;  wcex.cbWndExtra = 0;  wcex.hInstance = hInstance;  wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_TESTEDRDLL2));  wcex.hCursor = LoadCursor(NULL, IDC\_ARROW);  wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW + 1);  wcex.lpszMenuName = MAKEINTRESOURCE(IDC\_TESTEDRDLL2);  wcex.lpszClassName = szWindowClass;  wcex.hIconSm = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI\_SMALL));  return RegisterClassEx(&wcex);  }  //  // FUNCTION: InitInstance(HINSTANCE, int)  //  // PURPOSE: Saves instance handle and creates main window  //  // COMMENTS:  //  // In this function, we save the instance handle in a global variable and  // create and display the main program window.  //  BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)  {  HWND hWnd;  hInst = hInstance; // Store instance handle in our global variable  hWnd = CreateWindow(szWindowClass, szTitle, WS\_OVERLAPPEDWINDOW,  CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0, NULL, NULL, hInstance, NULL);  if (!hWnd)  {  return FALSE;  }  ShowWindow(hWnd, nCmdShow);  UpdateWindow(hWnd);  return TRUE;  }  //  // FUNCTION: WndProc(HWND, UINT, WPARAM, LPARAM)  //  // PURPOSE: Processes messages for the main window.  //  // WM\_COMMAND - process the application menu  // WM\_PAINT - Paint the main window  // WM\_DESTROY - post a quit message and return  //  //  LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)  {  int wmId, wmEvent;  PAINTSTRUCT ps;  HDC hdc;  switch (message)  {  case WM\_COMMAND:  wmId = LOWORD(wParam);  wmEvent = HIWORD(wParam);  // Parse the menu selections:  switch (wmId)  {  case IDM\_ABOUT:  DialogBox(hInst, MAKEINTRESOURCE(IDD\_ABOUTBOX), hWnd, About);  break;  case IDM\_EXIT:  DestroyWindow(hWnd);  break;  default:  return DefWindowProc(hWnd, message, wParam, lParam);  }  break;  case WM\_PAINT:  hdc = BeginPaint(hWnd, &ps);  RECT rec;  GetClientRect(hWnd, &rec);  EdrCenterText(hdc, &rec, TEXT("I LIKE CAT"));  EndPaint(hWnd, &ps);  break;  case WM\_DESTROY:  PostQuitMessage(0);  break;  default:  return DefWindowProc(hWnd, message, wParam, lParam);  }  return 0;  }  // Message handler for about box.  INT\_PTR CALLBACK About(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)  {  UNREFERENCED\_PARAMETER(lParam);  switch (message)  {  case WM\_INITDIALOG:  return (INT\_PTR)TRUE;  case WM\_COMMAND:  if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)  {  EndDialog(hDlg, LOWORD(wParam));  return (INT\_PTR)TRUE;  }  break;  }  return (INT\_PTR)FALSE;  } |

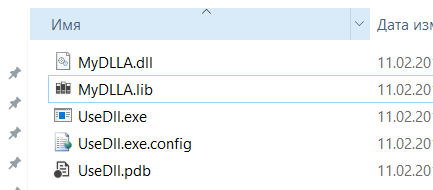


**ИСПОЛЬЗОВАНИЕ DLL из С++.**

Создаем следующий библиотечный компонент

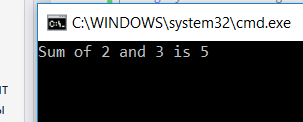
|  |
| --- |
| #include "stdafx.h"  #include <stdio.h>  #include <time.h>  #include <ctime>  #include <iostream>  #include <iomanip>  #include <chrono>  #pragma warning(disable : 4996)  extern "C" \_\_declspec(dllexport) int \_\_stdcall AddNumbers(int x, int y)  {  return x + y;  } |

Подключим его в проект (Console Application) с#:



|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Runtime.InteropServices;  namespace UseDll  {  class Program  {  [DllImport("MyDLLA.dll", EntryPoint = "AddNumbers")]  public static extern int AddNumbers();  [DllImport("user32.dll")]  public static extern int MessageBox(int hWnd, String pText,  String pCaption, int utype);  [STAThread]  static void Main(string[] args)  {  String pText = "Hello, programmer";  String pCaption = "See dll in action";  MessageBox(0, pText, pCaption, 0);  Console.WriteLine("Sum ", AddNumbers());  Console.ReadLine();  }  }  } |

Работа:



**ЗАДАНИЕ**.

Написать dll, которая возвращает текущее время, и вставить ее в c#-проект.

|  |
| --- |
| #include "stdafx.h"  #include <stdio.h>  #include <time.h>  #include <ctime>  #include <iostream>  #include <iomanip>  #include <chrono>  #pragma warning(disable : 4996)  using namespace std;  extern "C" \_\_declspec(dllexport) int \_\_stdcall AddNumbers()  {  time\_t now = time(0);  tm\* localtm = localtime(&now);  cout << "The local date and time is: " << asctime(localtm) << endl;  return 1;  } |

