Министерство науки и высшего образования Российской Федерации

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Индивидуальная работа

по дисциплине

«Управление ресурсами в вычислительных системах»

Факультет ПМИ

Группа ПМ-63

Студенты Майер В.А.

Преподаватели Стасышин В.М.

Хайленко Е.А.

Вариант 8

Уровень сложности 4

Новосибирск 2019

1. Постановка задачи

Индивидуальная работа включает 4 уровня сложности. Студент самостоятельно принимает решение, в рамках какого уровня он будет выполнять индивидуальную работу.

В рамках первого уровня необходимо реализовать консольное Windows-приложение, в котором создаётся один дочерний поток, реализующий требования варианта задания и выводящий результат на консоль.

В рамках второго уровня необходимо реализовать минимальное графическое Windows-приложение, в котором при создании окна, но до момента его отображения на экране, создаётся дочерний поток, который реализует задачу в соответствии с выбранным вариантом, после чего полученный результат отображается в графическом окне.

При выполнении задания на третьем уровне необходимо реализовать все требований второго уровня с той разницей, что функция, выполняющая задание варианта задания, должна быть реализована в виде динамической библиотеки.

Четвёртый уровень предполагает полную реализацию требований третьего уровня, но дополнительно в функции из динамической библиотеки используется ассемблерная вставка, содержание которой зависит от номера варианта задания, который выполняется. Полученная информация из ассемблерной вставки выводится вместе с результатом решения задачи в соответствии с вариантом задания.

|  |  |
| --- | --- |
| Задание для выполнения  I, II, III уровня сложности | Дополнительное задание IV уровня сложности |
| Определить высоту экрана(в пикселях) | Определить наличие поддержки технологии SSE3 |

1. Описание метода решения задачи
2. Запуск главного приложения
3. Инициализация окна
4. При нажатии на кнопку создавать дочерний поток
5. Дочерний поток загружает динамическую библиотеку
6. Проверяет ее наличие и функции в ней
7. Вызывает функции из динамической библиотеки
8. Вызов функции определения высоты экрана
9. Вызов функции определения поддержки SSE3

* Вызывается ассемблерная вставка cpuid
* Проверка поддержки cpuid
* Проверка соответствующего бита информирующего о поддержке SSE3

1. Изменяет соответствующие элементы окна (очистка их содержимого и последующая их запись)
2. Закрывает библиотеку
3. При закрытии закрывает приложение(крестик или кнопка закрыть)
4. Описание программного средства

При помощи макроса CreateWindowW создается главное окно приложения.

DestroyWindow закрывается окно приложения

SetWindowText сложит для передачи сообщения окну, в нашем случае используются для изменения данных в Edite

Для открытия динамической библиотеке используется LoadLibrart

Для получения функции используется GetProcAddress

Для вызова функции вызывается функция, которая была результатом GetProcAddress

Побочный поток создается классом thread, и функцией detach переходит в независимый режим

GetSystemMatrix с параметром SM\_CYSCREEN возвращает высоту экрана в пиеселях

\_\_asm – место ассемблерной вставки

cupid - получает информацию о процессоре, результат помещается в регистры eax, ebx, ecx, edx

информацию о SSE3 мы смотрим в нулевом бите регистра ecx

Вывод результата происходит на экран в 2 дочерних окна.

Результатом работы программы является окно, и после нажатия на кнопку запуск информация о работе появляется на экран, в случае ошибок, они появляются в сплывающем окне.

Основное окно является полупрозрачным, и используется иконка из файла icon.bmp

***Запуск приложения***: запуск compile.bat, будет создано 2 файла SGT.dll и Window.exe, который сразу будет запущен.

Так же есть compile\_запасной.bat, который нужно запустить в самой командной строке VC, на тот случай если не удастся найти стандартный путь. И будет уже скомпилированные мной файлы.

Для выполнения функции надо нажать на кнопку Запуск(нижнюю левую кнопку)



Для запуска приложения требуется ОС не ниже Windows 7 (Программа проверялась на Windows 7 и Windows 10, на Linux работать не будет, т.к. используется библиотеки Windows.h).

1. **Исходный текст**

Файл compile.bat

cls

set pathVC="NULLPATH"

if exist "C:\Program Files (x86)\Microsoft Visual Studio 10.0\Common7\Tools\VsDevCmd.bat" set pathVC="C:\Program Files (x86)\Microsoft Visual Studio 10.0\Common7\Tools\VsDevCmd.bat"

if exist "C:\Program Files (x86)\Microsoft Visual Studio 11.0\Common7\Tools\VsDevCmd.bat" set pathVC="C:\Program Files (x86)\Microsoft Visual Studio 11.0\Common7\Tools\VsDevCmd.bat"

if exist "C:\Program Files (x86)\Microsoft Visual Studio 14.0\Common7\Tools\VsDevCmd.bat" set pathVC="C:\Program Files (x86)\Microsoft Visual Studio 14.0\Common7\Tools\VsDevCmd.bat"

if exist "C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\Common7\Tools\VsDevCmd.bat" set pathVC="C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\Common7\Tools\VsDevCmd.bat"

if /I %pathVC%=="NULLPATH" EXIT

call %pathVC%

if exist \*.exe del \*.exe

if exist \*.dll del \*.dll

cl /EHsc /utf-8 /Zc:wchar\_t Window.cpp stdafx.cpp kernel32.lib user32.lib gdi32.lib advapi32.lib

cl /LD dllmain.cpp kernel32.lib user32.lib gdi32.lib advapi32.lib /link /out:SGT.dll

del \*.obj \*.lib \*.exp

call Window.exe

/\*dllmain.cpp\*/

// dllmain.cpp : Определяет точку входа для приложения DLL.

#include "stdafx.h"

#include <intrin.h>

extern "C"

{

\_declspec(dllexport) int CYScreen(void)

{

return GetSystemMetrics(SM\_CYSCREEN); //Получение высоты экрана

}

\_declspec(dllexport) bool CheckSSE3(void)

{

bool result;

\_\_asm

{

//Проверка поддержки cpuid

mov eax, 0

cpuid

cmp eax, 1

jl fals //Если не поддерживе

mov eax, 1 //Проверка поддержки sse3

cpuid

//Рещультат -> 1 байт 3 переменной (1 - eax, 2-ebx, 3-ecx, 4-edx)

and ecx, 1 //накладываем маску проверки

cmp ecx, 0 //Если этот байт ноль, значит не поддерживается

je fals //Если этот бит

tru: //Вызод поддерживается

mov result, 1

jmp end //Переход в конец

fals: //Вызод не поддерживается или cpuid не поддерживается

mov result, 0

jmp end //Переход в конец

end:

}

return result;

#pragma region Код на с++

//int CPUInfo[4] = { -1 };

//

//cpuid(CPUInfo, 0);

//Проверка поддержки cpuid

//if (CPUInfo[0] >= 0x00000001)

//{

// cpuid(CPUInfo, 0x00000001); //Полчение информации о процессоре

// return (CPUInfo[2] & ((int)1 << 0)) != 0; //Проверка SSE3

//}

//return false;

#pragma endregion

}

}

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;

}

/\***Window.cpp**\*/

#include "stdafx.h"

#include "Window.h"

typedef int(\*CYScreen)(void); //Тип функции определение высоты экрана

typedef bool(\*CheckSSE3)(void); //Тип функции проверка SSE3

BOOL(WINAPI \*pSetLayer)(HWND, COLORREF, BYTE, DWORD);

HINSTANCE hInst; //Текущий экземпляр

HWND Edit1, Edit2; //Textboxs

ATOM MyRegisterClass(HINSTANCE hInstance); //Создание класса окна

BOOL InitInstance(HINSTANCE); //Создание окна

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);//Функция обработки сообщений

HICON HICONFromCBitmap(); //Функция загрузки иконки

int APIENTRY wWinMain(\_In\_ HINSTANCE hInstance, //Точка входа

\_In\_opt\_ HINSTANCE hPrevInstance,

\_In\_ LPWSTR lpCmdLine,

\_In\_ int nCmdShow)

{

MyRegisterClass(hInstance); // Создание класса

if (!InitInstance (hInstance)) // Выполнить инициализацию приложения:

{ // (Создает окно)

return FALSE; // В случае ошибки окна завершается

}

HACCEL hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_WINDOW));

MSG msg;

// Цикл основного сообщения:

while (GetMessage(&msg, nullptr, 0, 0))

{

if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

return (int) msg.wParam;

}

ATOM MyRegisterClass(HINSTANCE hInstance)

{

WNDCLASSEXW wcex = {0}; //Заполняем нулями

wcex.cbSize = sizeof(WNDCLASSEX);

auto ic = HICONFromCBitmap(); //Загружаем иконку

wcex.hIcon = ic ? ic : LoadIcon(nullptr, IDI\_WARNING);

wcex.style = CS\_VREDRAW; //Стиль окна

wcex.lpfnWndProc = WndProc; //Функция обработки

wcex.hInstance = hInstance;

wcex.hCursor = LoadCursor(nullptr, IDC\_HELP);

wcex.hbrBackground = (HBRUSH)CreateSolidBrush( RGB(230, 230, 255));

wcex.lpszClassName = L"ClassName";

return RegisterClassExW(&wcex);

}

BOOL InitInstance(HINSTANCE hInstance)

{

hInst = hInstance; // Сохранить маркер экземпляра в глобальной переменной

HWND hWnd = CreateWindowW(L"ClassName", L"var 8 by Valerka", WS\_OVERLAPPED | WS\_SYSMENU,

CW\_USEDEFAULT, 0,400,300, nullptr, nullptr, hInstance, nullptr);

//Ошибка создания окна

if (!hWnd)

{

return FALSE;

}

ShowWindow(hWnd, SW\_SHOWNOACTIVATE);

UpdateWindow(hWnd);

return TRUE;

}

void threadfunction(HWND hWnd)

{

auto hinstLib = LoadLibrary(TEXT("SGT.dll")); //Загружает библиотеку

if (hinstLib) //Проверка на наличие библиотеки

{

CYScreen cyscreen = (CYScreen)GetProcAddress(hinstLib, "CYScreen"); //Обращение кк функции из библиотеки

if (cyscreen) //Проверка на наличие такой функции

{

WCHAR t[255];

int x = cyscreen(); //Выполнение функции

wsprintf(t, L"Основное задание:\r\n\r\n\r\n\"Определить высоту экрана(в пикселях)\"\r\n\r\nВысота экрана: %d", x);

SetWindowText(Edit1, t);

}

else

MessageBox(hWnd, L"Не удалось найти функцию определения размера экрана", 0, MB\_OK);

CheckSSE3 sse3 = (CheckSSE3)GetProcAddress(hinstLib, "CheckSSE3");

if (sse3)

{

WCHAR t[255];

bool x = sse3(); //Выполнение функции

wsprintf(t, L"Дополнительное задание:\r\n\r\n\"Определить наличие поддержки технологии SSE3\"\r\n\r\nПоддержка SSE3: %s", x ? L"true" : L"false");

SetWindowText(Edit2, t);

}

else

MessageBox(hWnd, L"Не удалось найти функцию проверка SSE3", 0, MB\_OK);

}

else

MessageBox(hWnd, L"Ошибка открытия DLL", 0, MB\_OK);

FreeLibrary(hinstLib);

}

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)

{

switch (message)

{

case WM\_COMMAND:

switch (wParam)

{

case 1: //Кнопка выполнить

{

thread t(threadfunction, hWnd); //Создает поток

t.detach(); //Отцепляет поток

} //Если не отцепить поток, то программа будет ждать

break; //его заверения и окно будет это время висеть

case 2: //Закрытие по кнопке

DestroyWindow(hWnd); //Вызываем закрытие окна

break;

default:

break;

}

break;

case WM\_CREATE:

{ //Создание кнопок и тектовых полей

HWND hwndButton = CreateWindow(L"BUTTON",L"Запуск", WS\_TABSTOP | WS\_VISIBLE | WS\_CHILD | BS\_DEFPUSHBUTTON, 10,225,100,25,hWnd,(HMENU)1,hInst, NULL);

HWND hwndButton1 = CreateWindow(L"BUTTON", L"Закрыть", WS\_TABSTOP | WS\_VISIBLE | WS\_CHILD | BS\_DEFPUSHBUTTON,270, 225, 100, 25, hWnd, (HMENU)2, hInst, NULL);

Edit1 = CreateWindow(L"EDIT", L"", ES\_LEFT | ES\_MULTILINE | ES\_READONLY | WS\_VISIBLE | WS\_CHILD, 10, 10, 175, 200, hWnd, (HMENU)3, hInst, NULL);

Edit2 = CreateWindow(L"EDIT", L"", ES\_LEFT | ES\_MULTILINE | ES\_READONLY | WS\_VISIBLE | WS\_CHILD, 200, 10, 175, 200, hWnd, (HMENU)4, hInst, NULL);

HINSTANCE hMod = LoadLibrary(L"user32.dll"); //Полупрозрачность окна

if (hMod) //если удалось подгрузить библиотеку

{

pSetLayer = (int(\_\_stdcall\*)(HWND, COLORREF, BYTE, DWORD))GetProcAddress(hMod, "SetLayeredWindowAttributes");

if (pSetLayer) //Если удалось найти функцию

if (SetWindowLong(hWnd, GWL\_EXSTYLE, 0x80000))

pSetLayer(hWnd, 0, 255 - 10 \* 255 / 100, 0x02);

}

}

break;

case WM\_CTLCOLORSTATIC:

{

//Стиль для текстовых полей

HFONT hFont = CreateFont(18, 0, 0, 0, FW\_MEDIUM, TRUE, FALSE, FALSE, DEFAULT\_CHARSET, OUT\_OUTLINE\_PRECIS,

CLIP\_DEFAULT\_PRECIS, CLEARTYPE\_QUALITY, VARIABLE\_PITCH, TEXT("Comic sans MS"));

SetBkMode((HDC)wParam, TRANSPARENT); //Убираем зарисовку за буквами

SelectObject((HDC)wParam, hFont); //Подкрепляем стиль

return (INT\_PTR)GetStockObject(NULL\_BRUSH); //Цвет фона текстовых полей прозрачный

}

case WM\_CTLCOLORBTN:

{

//Стиль для строк

HFONT hFont = CreateFont(20, 0, 15, 0, FW\_NORMAL, FALSE, FALSE, FALSE, ANSI\_CHARSET, OUT\_DEFAULT\_PRECIS,

CLIP\_DEFAULT\_PRECIS, ANTIALIASED\_QUALITY, FF\_MODERN, TEXT("Mistral"));

SetBkMode((HDC)wParam, TRANSPARENT); //Убираем зарисовку за буквами

SelectObject((HDC)wParam, hFont);

return (INT\_PTR)GetStockObject(NULL\_BRUSH); //Цвет фона прозрачный(стандартный для кнопок)

}

case WM\_DESTROY:

PostQuitMessage(0); //Завершение работы

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

return 0;

}

//Загрузка иконки из файла icon.h (находящегося в этой же папке)

HICON HICONFromCBitmap()

{

HBITMAP hBmp = (HBITMAP)LoadImage(NULL,

L"icon.bmp",

IMAGE\_BITMAP, 0, 0,

LR\_LOADFROMFILE);

BITMAP bmp;

GetObject(hBmp, sizeof(BITMAP), (LPVOID)&bmp);

ICONINFO ii = { 0 };

ii.fIcon = TRUE;

ii.hbmColor = hBmp;

ii.hbmMask = hBmp;

HICON hIcon = CreateIconIndirect(&ii);

return hIcon;

}

/\*stdafx.h\*/

// stdafx.h: включаемый файл для стандартных системных включаемых файлов

// или включаемых файлов для конкретного проекта, которые часто используются, но

// нечасто изменяются

//

#pragma once

#include "targetver.h"

#define WIN32\_LEAN\_AND\_MEAN // Исключите редко используемые компоненты из заголовков Windows

// Файлы заголовков Windows

#include <windows.h>

// Файлы заголовков среды выполнения C

#include <stdlib.h>

#include <malloc.h>

#include <memory.h>

#include <tchar.h>

#include <Thread>

#include <iostream>

using namespace std;

// установите здесь ссылки на дополнительные заголовки, требующиеся для программы

/\*Остальные файлы, которые были созданы автоматически\*/

#targetver.h

// Включение SDKDDKVer.h обеспечивает определение самой последней доступной платформы Windows.

// Если требуется выполнить сборку приложения для предыдущей версии Windows, включите WinSDKVer.h и

// задайте для макроса \_WIN32\_WINNT значение поддерживаемой платформы перед включением SDKDDKVer.h.

#include <SDKDDKVer.h>

#Window.h

#include "resource.h"

// Resource.h

//{{NO\_DEPENDENCIES}}

// Включаемый файл, созданный в Microsoft Visual C++.

// Используется Window.rc

#define IDS\_APP\_TITLE 103

#define IDR\_MAINFRAME 128

#define IDD\_WINDOW\_DIALOG 102

#define IDD\_ABOUTBOX 103

#define IDM\_ABOUT 104

#define IDM\_EXIT 105

#define IDI\_WINDOW 107

#define IDI\_SMALL 108

#define IDC\_WINDOW 109

#define IDC\_MYICON 2

#ifndef IDC\_STATIC

#define IDC\_STATIC -1

#endif

// Следующие стандартные значения для новых объектов

//

#ifdef APSTUDIO\_INVOKED

#ifndef APSTUDIO\_READONLY\_SYMBOLS

#define \_APS\_NO\_MFC 130

#define \_APS\_NEXT\_RESOURCE\_VALUE 129

#define \_APS\_NEXT\_COMMAND\_VALUE 32771

#define \_APS\_NEXT\_CONTROL\_VALUE 1000

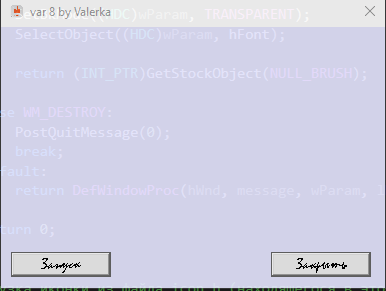
#define \_APS\_NEXT\_SYMED\_VALUE 110

#endif

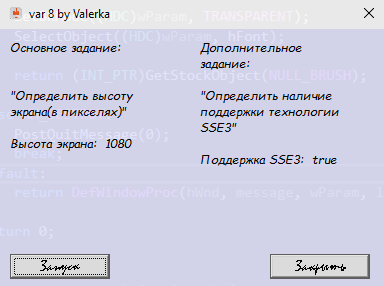
#endif

1. Тестирование программы

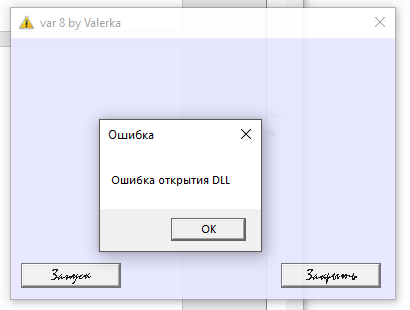
***Запуск приложения***



***Выполнение приложения***

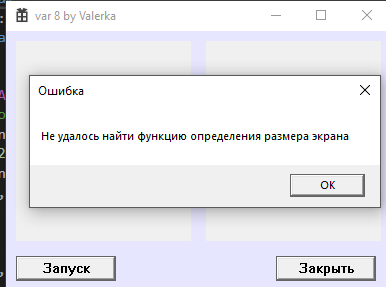


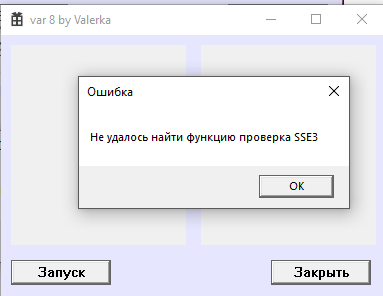
***Выполнение без dll***



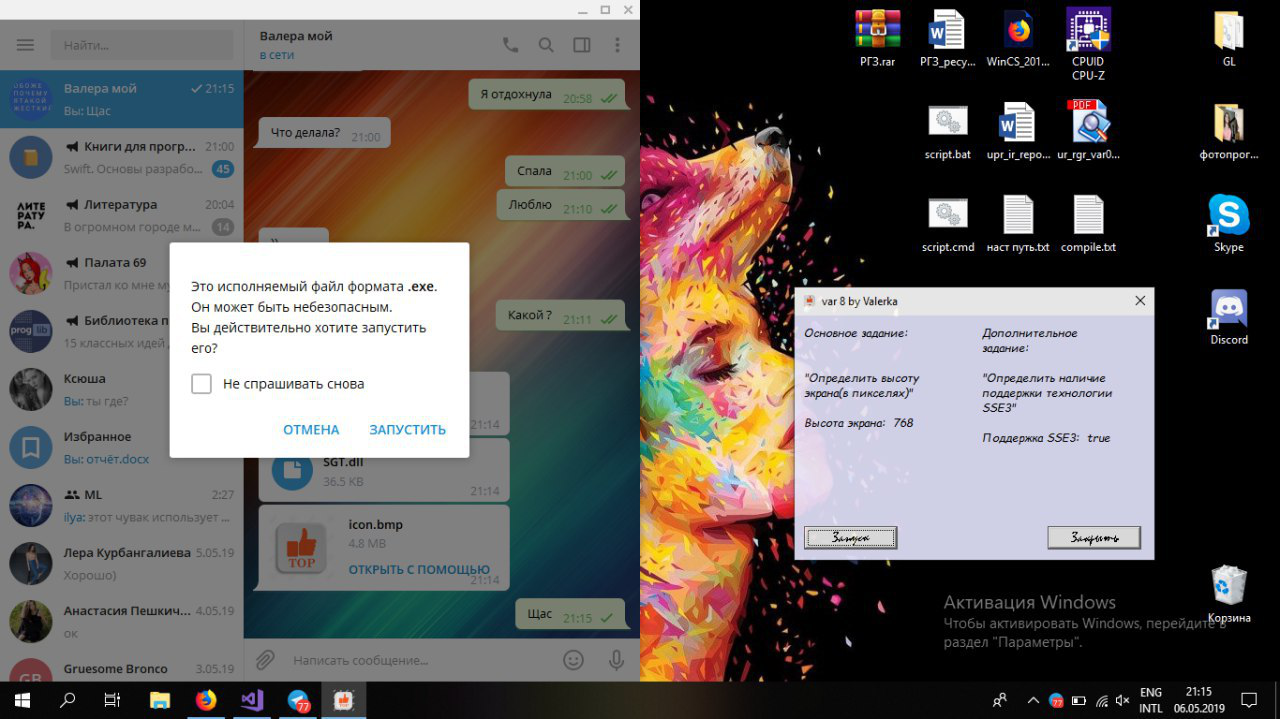
 означает что не удалось найти иконку приложения

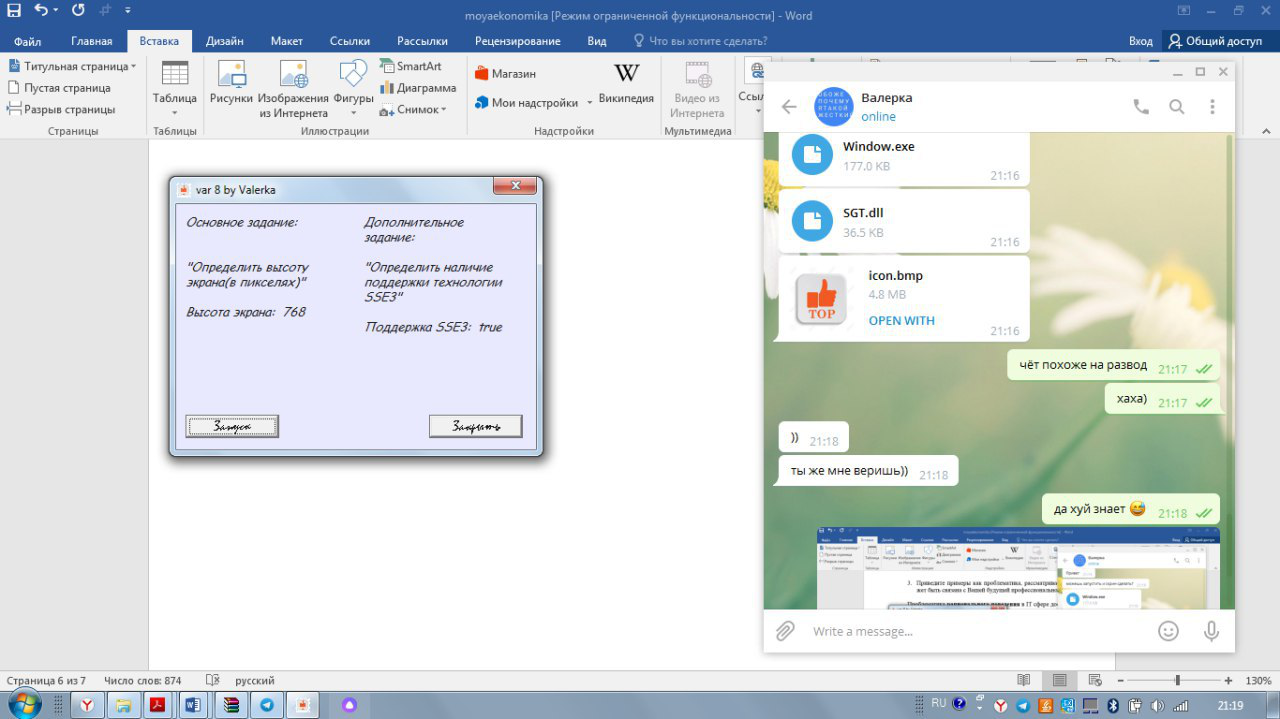
***Выполнение с другой dll***





***Тесты на других компьютерах***





Стиль приложения зависит от стиля операционной системы.