|  |  |  |
| --- | --- | --- |
| Министерство науки и высшего образования  Российской Федерации | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования | | |
| «Новосибирский государственный технический университет» | | |
| https://lh3.googleusercontent.com/0mwH3EqF2nXPGNdaGFqfMGaoAsDjLJ4wqE-gGxadp4VBkvcrJbYcowLDrPNQCDqeCIyyEGgaXfWxk6j6AeDOBmIxZDmluCv_z8tfmvIXWpMXmQiwZWHoII_yPMRjE85Wgw | | |
| Кафедра Теоретической и прикладной информатики | | |
|  | | |
| Расчетно-графическая работа | | |
| по дисциплине «Управление ресурсами в вычислительных системах» | | |
|  | | |
|  | | |
|  | | |
| https://lh6.googleusercontent.com/n_IEPqDfZab1eC2gHt4Hn2EMCwFanVWWkZW1lgPlJh3dwRm6FHBz83qqd38JhHo8koCyXYhN-Vu-fEb8v_3rxgIZ2JLxzlMgMF6axuxALKOkeRQsLFAYfUgKux3uCfEChA | Факультет: | ПМИ |
| Группа: | ПМИ-72 |
| Вариант: | 8 |
| Студенты: | Антонов Сергей |
|  |  |
|  |  |
| Преподаватель: | Стасышин Владимир Михайлович  Сивак Мария Алексеевна |
|  | | |
| Новосибирск | | |
| 2020 | | |

1. Постановка задачи

Реализовать минимальное графическое Windows-приложение, в котором при создании окна, но до момента его отображения на экране, создается дочерний поток, вызывающий функцию динамической библиотеки. Эта функция определяет высоту экрана в пикселях. Полученный результат отображается в графическом окне.

1. Анализ задачи

Структура оконной Windows-программы, создание дочернего потока и вызов функции динамической библиотеки являются типовыми, главное в данном задании определение высоты экрана в пикселях.

* 1. **Определение высоты экрана**

Высота экрана может быть определена с помощью функции

**GetSystemMetrics(SM\_CYSCREEN)** которая возвращает информацию о текущей системе.

1. Алгоритм

Создаем главное окно с помощью функции CreateWindow, устанавливаем необходимые цвета и шрифты. Затем запускаем цикл обработки сообщений, при поступлении сообщения вызывается функция WndProc, в которой определены следующие основные реакции:

1. Создание окна кнопки;
2. Создаём окно относительно главного окна для вывода информации. Создаётся поток, в котором, подключается динамическая библиотека, вызываются функции получения высоты экрана.
3. Уничтожение окна.

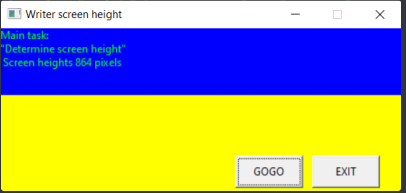
Основное приложение содержится в файле Source.cpp, а динамическая библиотека – в файле info.cpp. Для их компиляции необходимо запустить файл сценария компиляции compile.bat из командной строки VisualStudio

1. Тестирование разработанной программы

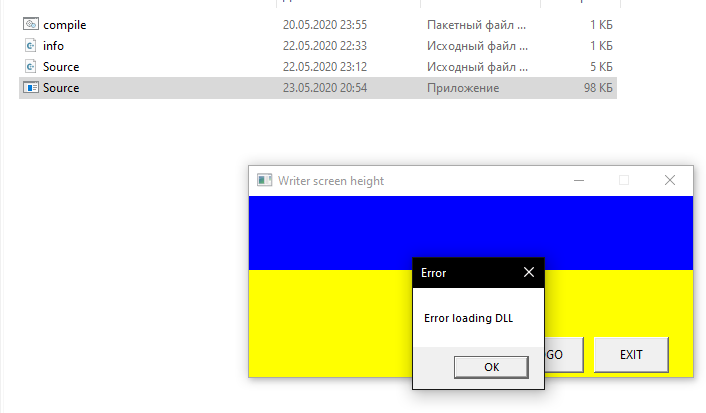
*Тестирование системы c монитором ПК*



*Тестирование системы с монитором ноутбука*



*Тестирование, когда dll файл не удалось открыть*



1. Тексты разработанных программ

Source.cpp

#define \_CRT\_SECURE\_NO\_WARNINGS

#define UNICODE

#include <windows.h>

#include <string>

#include <sstream>

#include <tchar.h>

HWND hwnd, button, button\_exit, screenheightLabel;

static HBRUSH hBrush = CreateSolidBrush(RGB(0, 0, 255));

void thread()

{

HINSTANCE hinstLib = LoadLibrary(TEXT("info.dll"));//загрузка динамической библиотеки

if (hinstLib != NULL)

{

typedef int (\*getSystemMetrics\_t)();

getSystemMetrics\_t getSystemMetrics = (getSystemMetrics\_t)GetProcAddress(hinstLib, "getSystemMetrics");

int hight = getSystemMetrics();

FreeLibrary(hinstLib);//Закрытие библиотеки

char screenhight[1024];

sprintf\_s(screenhight, "Main task:\n\"Determine screen height\"\

\n Screen heights %d pixels", hight);

WCHAR screenhightWString[1024];

MultiByteToWideChar(CP\_ACP, 0, screenhight, strlen(screenhight) + 1, (LPWSTR)screenhightWString, sizeof(screenhightWString) / sizeof(screenhightWString[0]));

// И выведем в окно

SetWindowText(screenheightLabel, LPWSTR(screenhightWString));

}

else

{

MessageBox(hwnd, L"Error loading DLL", L"Error", 0);// вывод ошибки

}

}

// Обработчик сообщений

LONG WINAPI WndProc(HWND hwnd, UINT Message, WPARAM wparam, LPARAM lparam)

{

switch (Message)

{

case WM\_CTLCOLORSTATIC:

{

DWORD CtrlID = GetDlgCtrlID((HWND)lparam);

if (CtrlID == 3001)

{

HDC hdcStatic = (HDC)wparam;

SetTextColor(hdcStatic, RGB(0, 255, 0));// установка цвета текста

SetBkColor(hdcStatic, RGB(0, 0, 255));// установка цвета обводки текста

return (INT\_PTR)hBrush;

}

}

break;

case WM\_COMMAND:

{

if (LOWORD(wparam) == 3000)

{

HANDLE hThread;//Дескриптор потока

DWORD IDThread;//Идентификатор потока

hThread = CreateThread(NULL, 0, (LPTHREAD\_START\_ROUTINE)thread, NULL, 0, &IDThread); // Запуск потока

CloseHandle(hThread);// Удаление дескриптора потока

}

if (LOWORD(wparam) == 3002)

{

PostQuitMessage(0);

}

}

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

return DefWindowProc(hwnd, Message, wparam, lparam);

}

return 0;

}

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)

{

WNDCLASS w; //Элемент структуры класса

memset(&w, 0, sizeof(WNDCLASS));

w.style = CS\_HREDRAW | CS\_VREDRAW | CS\_BYTEALIGNCLIENT;//Стиль окна

w.lpfnWndProc = WndProc; // Обработчик сообщений

w.hInstance = hInstance; //Дескриптор приложения

w.hbrBackground = CreateSolidBrush(RGB(255, 255, 0));

w.lpszClassName = L"window";//Имя окна

RegisterClass(&w); //Регистрация класса окна

NONCLIENTMETRICS ncm;

ncm.cbSize = sizeof(NONCLIENTMETRICS);

OSVERSIONINFO osvi;

ZeroMemory(&osvi, sizeof(OSVERSIONINFO));

osvi.dwOSVersionInfoSize = sizeof(OSVERSIONINFO);

SystemParametersInfo(SPI\_GETNONCLIENTMETRICS, sizeof(NONCLIENTMETRICS), &ncm, 0);

HFONT hf = CreateFontIndirect(&ncm.lfMessageFont);

HDC hDCScreen = GetDC(NULL);

int Horres = GetDeviceCaps(hDCScreen, HORZRES);

int Vertres = GetDeviceCaps(hDCScreen, VERTRES);

int Height = 220;

int width = 460;

int ButtonHeight = 36;

int Buttonwidth = 75;

hwnd = CreateWindow(L"window", L"Writer screen height", \

WS\_OVERLAPPED | WS\_CAPTION | WS\_SYSMENU | WS\_MINIMIZEBOX, \

(Horres - width) / 2, (Vertres - Height) / 2, width, Height, NULL, NULL, hInstance, NULL);

CreateWindow(L"button", L"GOGO", \

WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON | BS\_VCENTER | BS\_CENTER, width - 2 \* Buttonwidth - 50, Height - 2 \* ButtonHeight - 7, \

Buttonwidth, ButtonHeight, hwnd, (HMENU)3000, hInstance, NULL);

button\_exit = CreateWindow(L"button", L"EXIT", \

WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON | BS\_VCENTER | BS\_CENTER, width - Buttonwidth - 40, Height - 2 \* ButtonHeight - 7, \

Buttonwidth, ButtonHeight, hwnd, (HMENU)3002, hInstance, NULL);

screenheightLabel = CreateWindow(L"static", L"", WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON, \

0, 0, width, Height / 2 - ButtonHeight, hwnd, (HMENU)3001, hInstance, NULL);

for (int i = 0; i < 4; i++)

SendDlgItemMessage(hwnd, 3000 + i, WM\_SETFONT, (WPARAM)hf, TRUE);

ShowWindow(hwnd, nCmdShow);

UpdateWindow(hwnd);

MSG msg;

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

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

info.cpp

#define \_CRT\_SECURE\_NO\_WARNINGS

#define UNICODE

#include <stdio.h>

#include <windows.h>

#include <time.h>

// Определение высоты экрана в пикселях

extern "C" \_\_declspec(dllexport)

int getSystemMetrics()

{

return GetSystemMetrics(SM\_CYSCREEN);

}