|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | | | |
| Институт информационных технологий (ИТ) | |
| Кафедра инструментального и прикладного программного обеспечения (ИиППО) | |

|  |  |
| --- | --- |
| **ОТЧЁТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №4** | |
| **по дисциплине** | |
| «Системное программное обеспечение»  по теме: «Программирование внешних устройств» | |
|  | |
| Выполнил студент группы ИКБО-16-18 | Шамян Г.Г. |
| Принял преподаватель | Волков М.Ю. |

|  |  |  |
| --- | --- | --- |
| Работа выполнена | «\_\_»\_\_\_\_\_\_\_2021 г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_2021 г. | *(подпись руководителя)* |

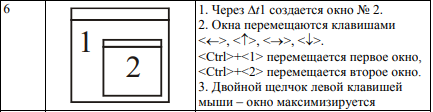
Москва 2021

**Цель работы**

Изучить основы программирования аппаратных устройств: клавиатуры, мыши, таймера.

**Задание**

На базе архитектуры WIN 32 Application создать многооконное приложение, где первое окно отображает информацию по клавиатуре, второе окно отображает информацию по мыши и третье окно отображает информацию по таймеру.



*Рис. 1 – Индивидуальный вариант.*

**Ход работы**

Итоговая программа представлена в листинге 1. Результат работы представлен на рисунках 2-4.

Листинг 1

#include <windows.h>

#include <stdlib.h>

#include <string.h>

#include <tchar.h>

#define ID\_TIMER\_1 1

static TCHAR szMainWindowClass[] = \_T("MainWindowClass");

static TCHAR szChildWindowClass[] = \_T("ChildWindowClass");

static TCHAR szMainWindowTitle[] = \_T("MainWindow");

static TCHAR szChildTitle[] = \_T("Child Window");

HINSTANCE hInst;

UINT uFirstDeltaTime = 1000;

BOOL bControlFlag = false;

HWND hWndMain;

HWND hWndChild;

RECT rect;

LONG x = 0;

LONG y = 0;

LONG sizeMain = 600;

LONG sizeChild = 200;

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

LRESULT CALLBACK WndProcChild(HWND, UINT, WPARAM, LPARAM);

int CALLBACK WinMain(

\_In\_ HINSTANCE hInstance,

\_In\_opt\_ HINSTANCE hPrevInstance,

\_In\_ LPSTR lpCmdLine,

\_In\_ int nCmdShow

)

{

WNDCLASSEX wcex;

wcex.cbSize = sizeof(WNDCLASSEX);

wcex.style = CS\_DBLCLKS | CS\_HREDRAW | CS\_VREDRAW;

wcex.lpfnWndProc = WndProc;

wcex.cbClsExtra = 0;

wcex.cbWndExtra = 0;

wcex.hInstance = hInstance;

wcex.hIcon = LoadIcon(hInstance, IDI\_APPLICATION);

wcex.hCursor = LoadCursor(NULL, IDC\_ARROW);

wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW + 1);

wcex.lpszMenuName = NULL;

wcex.lpszClassName = szMainWindowClass;

wcex.hIconSm = LoadIcon(wcex.hInstance, IDI\_APPLICATION);

if (!RegisterClassEx(&wcex))

{

MessageBox(NULL,

\_T("Call to RegisterClassEx failed!"),

\_T("Windows Desktop Guided Tour"),

NULL);

return 1;

}

hInst = hInstance;

hWndMain = CreateWindow(

szMainWindowClass,

szMainWindowTitle,

WS\_OVERLAPPEDWINDOW,

0, 0,

sizeMain, sizeMain,

NULL,

NULL,

hInstance,

NULL

);

if (!hWndMain)

{

MessageBox(NULL,

\_T("Call to CreateWindow failed!"),

\_T("Windows Desktop Guided Tour"),

NULL);

return 1;

}

ShowWindow(hWndMain,

nCmdShow);

UpdateWindow(hWndMain);

SetTimer(hWndMain, ID\_TIMER\_1, uFirstDeltaTime, NULL);

MSG msg;

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

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return (int)msg.wParam;

}

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

{

switch (message)

{

case WM\_TIMER:

switch (wParam) {

case ID\_TIMER\_1: {

KillTimer(hWnd, ID\_TIMER\_1);

WNDCLASSEX wcexFirstChild;

wcexFirstChild.cbSize = sizeof(WNDCLASSEX);

wcexFirstChild.style = CS\_DBLCLKS | CS\_HREDRAW | CS\_VREDRAW;

wcexFirstChild.lpfnWndProc = WndProcChild;

wcexFirstChild.cbClsExtra = 0;

wcexFirstChild.cbWndExtra = 0;

wcexFirstChild.hInstance = hInst;

wcexFirstChild.hIcon = LoadIcon(hInst, IDI\_APPLICATION);

wcexFirstChild.hCursor = LoadCursor(NULL, IDC\_ARROW);

wcexFirstChild.hbrBackground = (HBRUSH)(COLOR\_WINDOW + 1);

wcexFirstChild.lpszMenuName = NULL;

wcexFirstChild.lpszClassName = szChildWindowClass;

wcexFirstChild.hIconSm = LoadIcon(wcexFirstChild.hInstance, IDI\_APPLICATION);

if (!RegisterClassEx(&wcexFirstChild))

{

MessageBox(NULL,

\_T("Call to RegisterClassEx failed! (wcexFirstChild)"),

\_T("Windows Desktop Guided Tour"),

NULL);

return 1;

}

hWndChild = CreateWindow(

szChildWindowClass,

szChildTitle,

WS\_OVERLAPPEDWINDOW | WS\_VISIBLE,

200, 200,

sizeChild, sizeChild,

hWnd,

NULL,

hInst,

NULL

);

if (!hWndChild)

{

MessageBox(NULL,

\_T("Call to CreateWindow failed! (hWndFirstChild)"),

\_T("Windows Desktop Guided Tour"),

NULL);

return 1;

}

CreateWindow(TEXT("STATIC"), TEXT("1"),

WS\_VISIBLE | WS\_CHILD,

0, 0, 20, 20,

hWndChild, (HMENU)NULL, NULL, NULL);

break;

}

default:

break;

}

break;

case WM\_KEYDOWN:

switch (wParam) {

case VK\_CONTROL:

bControlFlag = true;

break;

case VK\_LEFT: // Обрабатывает клавишу LEFT ARROW (Стрелка влево).

GetWindowRect(hWnd, &rect);

x = rect.left - 20;

y = rect.top;

SetWindowPos(hWnd, HWND\_TOP, x, y, sizeMain, sizeMain, NULL);

break;

case VK\_RIGHT: // Обрабатывает клавишу RIGHT ARROW (Стрелка вправо).

GetWindowRect(hWnd, &rect);

x = rect.left + 20;

y = rect.top;

SetWindowPos(hWnd, HWND\_TOP, x, y, sizeMain, sizeMain, NULL);

break;

case VK\_UP: // Обрабатывает клавишу UP ARROW (Стрелка вверх).

GetWindowRect(hWnd, &rect);

x = rect.left;

y = rect.top - 20;

SetWindowPos(hWnd, HWND\_TOP, x, y, sizeMain, sizeMain, NULL);

break;

case VK\_DOWN: // Обрабатывает клавишу DOWN ARROW (Стрелка вниз).

GetWindowRect(hWnd, &rect);

x = rect.left;

y = rect.top + 20;

SetWindowPos(hWnd, HWND\_TOP, x, y, sizeMain, sizeMain, NULL);

break;

case 0x32: //окно 2

if (bControlFlag) {

HWND hWindow = hWndChild; // Дескриптор нужного окна, которому хотим передать фокус

DWORD dwThreadID = GetWindowThreadProcessId(hWindow, NULL); // Получаем идентификатор потока окна

DWORD dwThisThreadID = GetWindowThreadProcessId(hWndChild, NULL);

AttachThreadInput(dwThisThreadID, dwThreadID, TRUE); // Присоединяемся к потоку обработки ввода

SetFocus(hWindow); // Посылаем сообщение для установки фокуса

AttachThreadInput(dwThisThreadID, dwThreadID, FALSE); // Отсоединяемся

}

break;

default:

break;

}

break;

case WM\_KEYUP:

if (wParam == VK\_CONTROL) {

bControlFlag = false;

MessageBox(NULL,

\_T("CTRL down"),

\_T("MSG"),

NULL);

}

break;

case WM\_LBUTTONDBLCLK:

GetWindowRect(hWnd, &rect);

x = rect.left;

y = rect.top + 20;

sizeMain = 1000;

SetWindowPos(hWnd, HWND\_TOP, x, y, sizeMain, sizeMain, NULL);

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

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

break;

}

return 0;

}

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

{

switch (message)

{

case WM\_KEYDOWN:

switch (wParam) {

case VK\_CONTROL:

bControlFlag = true;

break;

case VK\_LEFT: // Обрабатывает клавишу LEFT ARROW (Стрелка влево).

GetWindowRect(hWnd, &rect);

x = rect.left - 20;

y = rect.top;

SetWindowPos(hWnd, HWND\_TOP, x, y, sizeChild, sizeChild, NULL);

break;

case VK\_RIGHT: // Обрабатывает клавишу RIGHT ARROW (Стрелка вправо).

GetWindowRect(hWnd, &rect);

x = rect.left + 20;

y = rect.top;

SetWindowPos(hWnd, HWND\_TOP, x, y, sizeChild, sizeChild, NULL);

break;

case VK\_UP: // Обрабатывает клавишу UP ARROW (Стрелка вверх).

GetWindowRect(hWnd, &rect);

x = rect.left;

y = rect.top - 20;

SetWindowPos(hWnd, HWND\_TOP, x, y, sizeChild, sizeChild, NULL);

break;

case VK\_DOWN: // Обрабатывает клавишу DOWN ARROW (Стрелка вниз).

GetWindowRect(hWnd, &rect);

x = rect.left;

y = rect.top + 20;

SetWindowPos(hWnd, HWND\_TOP, x, y, sizeChild, sizeChild, NULL);

break;

case 0x31: //окно 1

if (bControlFlag) {

HWND hWindow = hWndMain; // Дескриптор нужного окна, которому хотим передать фокус

DWORD dwThreadID = GetWindowThreadProcessId(hWindow, NULL); // Получаем идентификатор потока окна

DWORD dwThisThreadID = GetWindowThreadProcessId(hWndMain, NULL);

AttachThreadInput(dwThisThreadID, dwThreadID, TRUE); // Присоединяемся к потоку обработки ввода

SetFocus(hWindow); // Посылаем сообщение для установки фокуса

AttachThreadInput(dwThisThreadID, dwThreadID, FALSE); // Отсоединяемся

}

break;

default:

break;

}

break;

case WM\_KEYUP:

if (wParam == VK\_CONTROL) {

bControlFlag = false;

MessageBox(NULL,

\_T("CTRL down"),

\_T("MSG"),

NULL);

}

break;

case WM\_LBUTTONDBLCLK:

GetWindowRect(hWnd, &rect);

x = rect.left;

y = rect.top + 20;

sizeChild = 1000;

SetWindowPos(hWnd, HWND\_TOP, x, y, sizeChild, sizeChild, NULL);

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

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

break;

}

return 0;

}

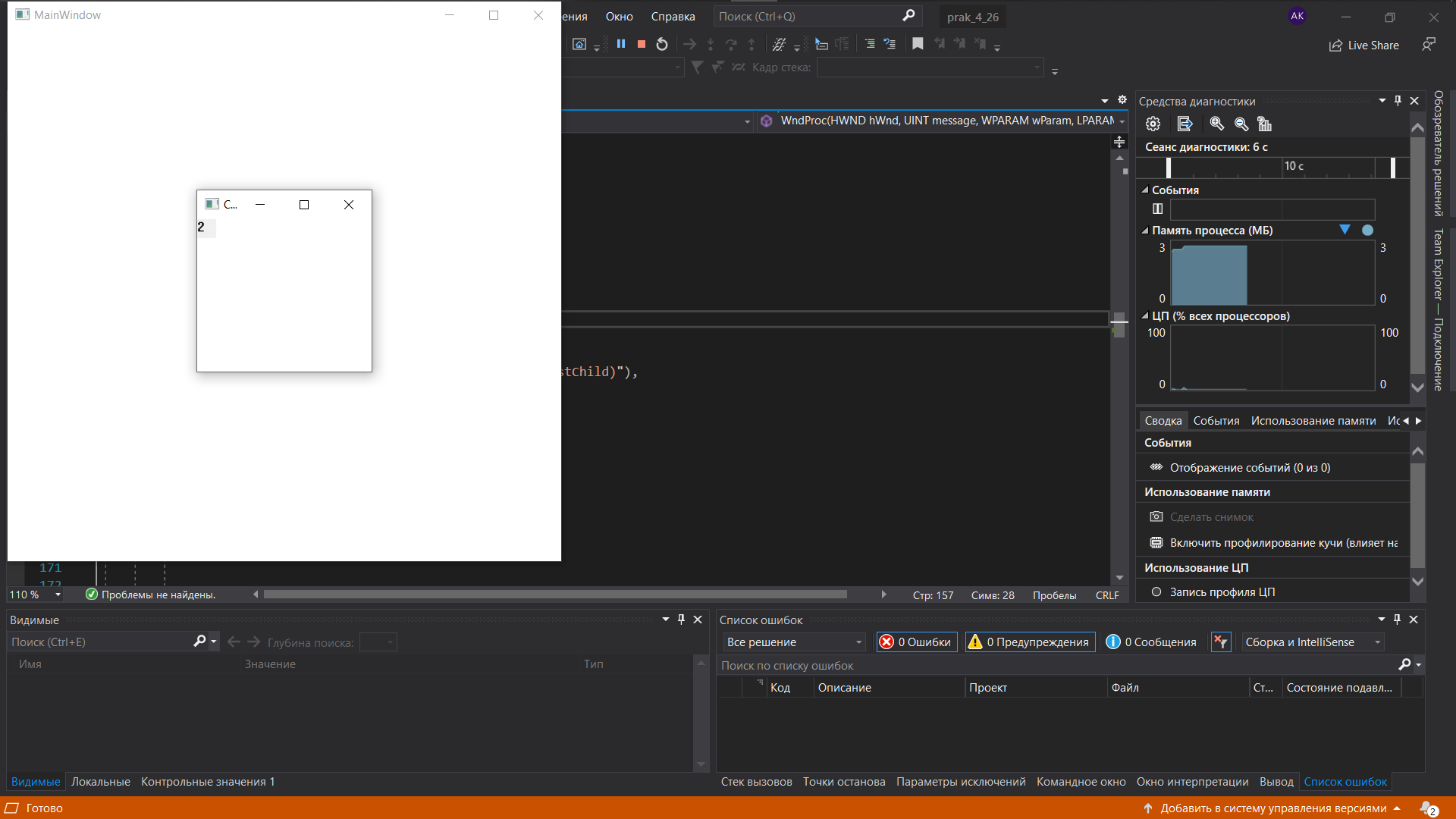


Рис. 2 – Созданное окно программы.

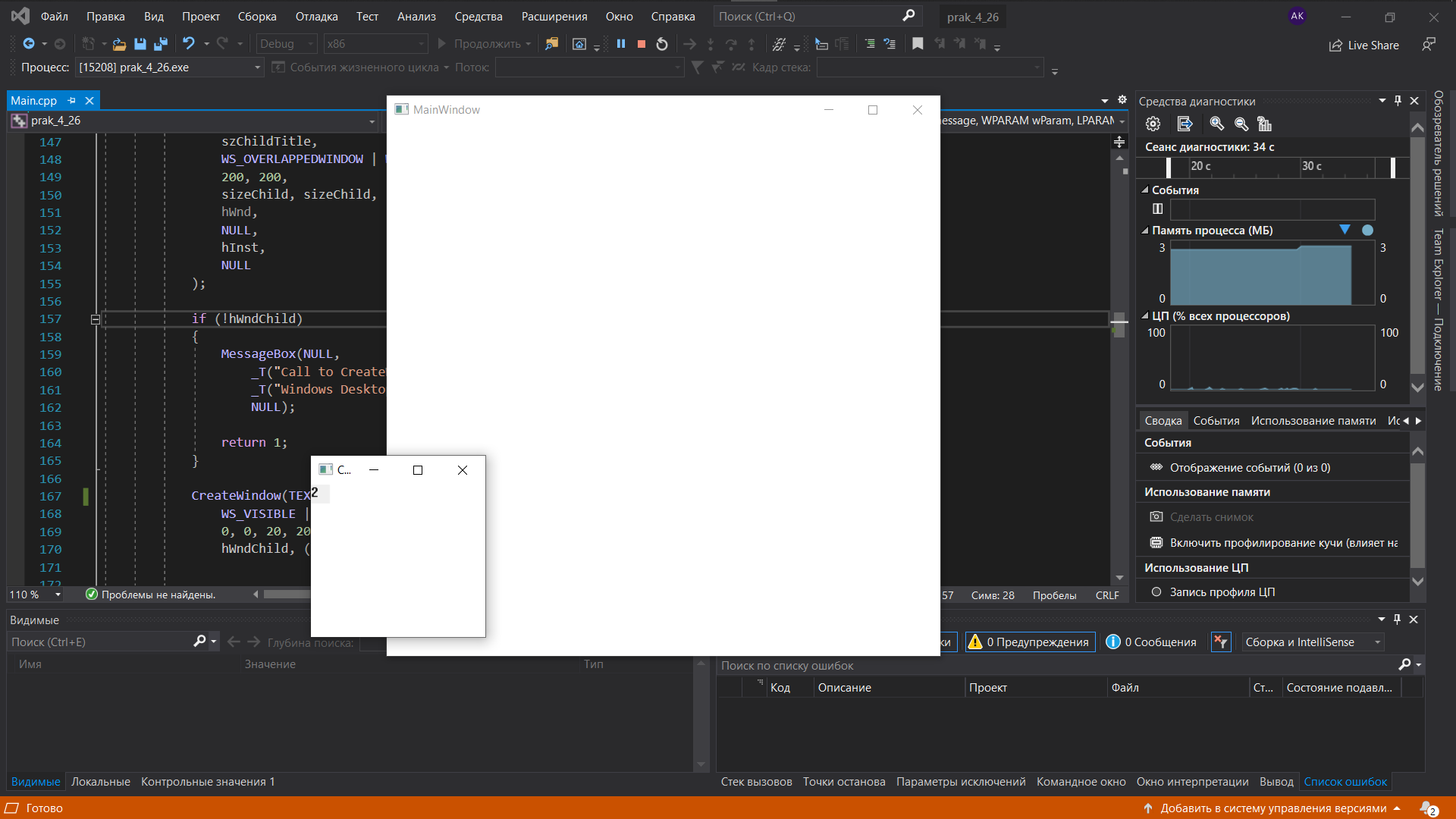


Рис.3 – Перемещение окон.

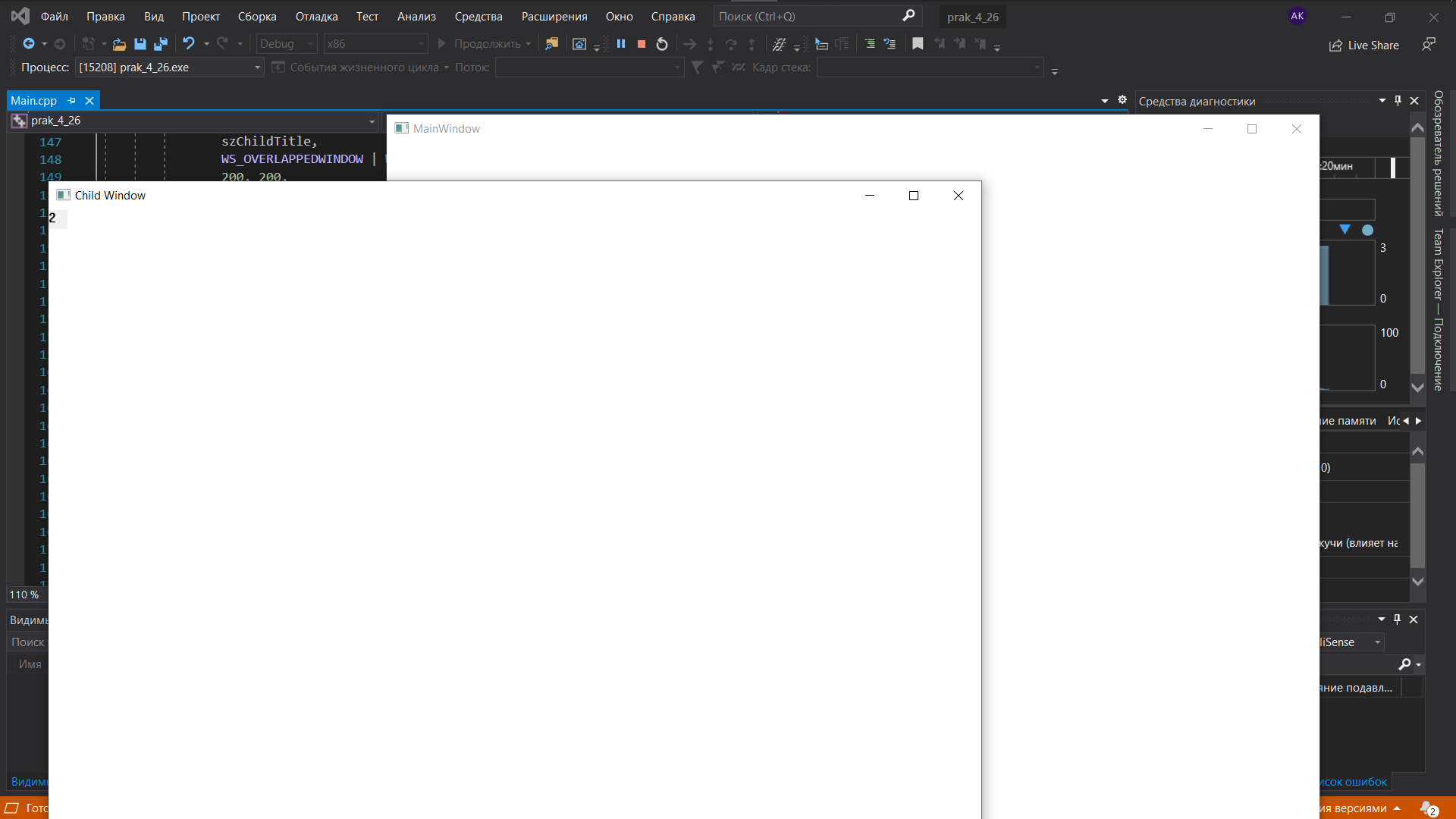


Рис.4 – Увеличение окон по двойному щелчку.

**Вывод**

В ходе практической работы были изучены основы программирования аппаратных устройств: клавиатуры, мыши, таймера.

**Список использованных источников**

1. Пошаговое руководство. Создание классического оконного приложение [Электронный ресурс]. URL: https://docs.microsoft.com/ru-ru/cpp/windows/walkthrough-creating-windows-desktop-applications-cpp?view=msvc-160
2. Администрирование в Windows 10. [Электронный ресурс] : [https://docs.microsoft.com/ru-ru/windows/client-management/administrative-tools-in-windows-10 /](https://docs.microsoft.com/ru-ru/windows/client-management/administrative-tools-in-windows-10%20/) Ричард Ворд 2020 — 100 с.
3. «Тайм-менеджмент для системных администраторов». / Лимончелли Т.А. М.: Символ-плюс 2007 — 247 с. ISBN:978-5-93286-090-8