Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей Кафедра информатики

Дисциплина: Операционные среды и системное программирование

**ОТЧЕТ**

к лабораторной работе № 1

на тему

ОСНОВЫ ПРОГРАММИРОВАНИЯ В WIN 32 API

ОКОННОЕ ПРИЛОЖЕНИЕ WIN 32 С МИНИМАЛЬНОЙ ДОСТАТОЧНОЙ ФУНКЦИОНАЛЬНОСТЬЮ.

ОБРАБОТКА ОСНОВНЫХ ОКОННЫХ СООБЩЕНИЙ

Выполнил:

студент гр. 153503

Татаринов В.В.

Проверил:

Гриценко Н.Ю.

Минск 2023

**СОДЕРЖАНИЕ**

[1 Цель работы 3](#_Toc146836467)

[2 Теоретические сведения 4](#_Toc146836468)

[3 Полученные результаты 6](#_Toc146836469)

[Выводы 7](#_Toc146836470)

[Список использованных источников 8](#_Toc146836471)

[Приложение А (обязательное) листинг кода 9](#_Toc146836472)

# **1 ЦЕЛЬ РАБОТЫ**

Изучить основы программирования в Win 32 API. Создать приложение Win 32 API, реализующее минимальный достаточный функционал, позволяющий обрабатывать основные оконные сообщения. Сделать простое оконное приложение, позволяющее вводить значения, обрабатывать их и выводить результат на экран.

**2 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Win API представляет собой основной набор интерфейсов прикладного программирования (API) Microsoft, доступных в операционных системах Microsoft Windows. Название Windows API в совокупности относится к нескольким различным реализациям платформы, которые часто называются собственными, одним из которых является Win32 API. Почти все программы Windows взаимодействуют с Windows API.

Win32 API был изначально разработан для 32-битных версий Windows, но в некоторых случаях его можно использовать и в более новых версиях Windows. Он представляет собой набор функций и библиотек для разработки приложений под операционные системы Windows. Этот API ориентирован на 32-битные системы (хотя существует и 64-битная версия, известная как Win64 API), и он предоставляет доступ к основным службам и функциям Windows, таким как создание окон, управление ресурсами, работа с файлами и многими другими.

Для создания такого приложения могут быть использованы различные языки программирования, включая C, C++, C#, и другие. Множество сред разработки (IDE), таких как Visual Studio, предоставляют инструменты для более удобной работы с Win32 API.

Win32 API предоставляет низкоуровневый доступ к функциональности Windows. Это позволяет создавать более гибкие и настраиваемые приложения, но также требует более тщательной работы с деталями.

В ходе выполнения данной лабораторной работы использовались следующие инструменты и концепции:

– WindowProcedure обрабатывает различные сообщения от операционной системы и взаимодействует с приложением в соответствии с этими сообщениями. Например, когда окно должно быть перерисовано, WindowProcedure может обработать сообщение WM\_PAINT и нарисовать содержимое окна. Когда пользователь закрывает окно, обработчик может обработать сообщение WM\_DESTROY и выполнить соответствующие действия, такие как закрытие приложения.

– CreateWindowW в Win32 API (Windows API) используется для создания окна с использованием широких символов (UNICODE). Она позволяет создавать различные типы окон, такие как главные окна, дочерние окна, кнопки, текстовые поля и многие другие элементы пользовательского интерфейса. Все вызовы данной функции были вынесены в функцию AddMenus.

– Message Loop (цикл обработки сообщений) в Win32 API представляет собой основной цикл выполнения приложения, который отвечает за обработку и диспетчеризацию всех сообщений, поступающих к окнам в приложении. Message Loop обеспечивает взаимодействие пользователя с приложением, обработку событий и управление окнами. Перед запуском Message Loop происходит инициализация приложения, включая создание окон и ресурсов, регистрацию классов окон, установку начальных параметров и т. д. Основной цикл приложения, который выполняется в функции WinMain или аналогичной точке входа, обычно обрабатывается циклом while. В оконной процедуре WindowProcedure, зарегистрированной для каждого окна, обрабатываются и реагируют на различные сообщения.

– Button представляет собой стандартный элемент управления (control) для создания кнопок в пользовательском интерфейсе Windows-приложений. Кнопки используются для взаимодействия пользователя с приложением, например, для запуска определенных действий или операций. Кнопки, как и другие элементы, создаются внутри родительского окна, таким образом, родительское окно является контейнером для кнопки. Каждой кнопке можно назначить уникальный идентификатор, который используется для идентификации кнопки и обработки событий, связанных с ней.

# **3 ПОЛУЧЕННЫЕ РЕЗУЛЬТАТЫ**

В результате выполнения лабораторной работы было создано простое оконное приложение, позволяющее вводить значения, обрабатывать их и выводить результат на экран. В качестве входных данных вводится число и месяц. При нажатии на кнопку, пользователю отображается дата на основе введенных им данных.

Интерфейс программы представляет собой поле для ввода, выпадающий список и кнопку (рисунок 1).



Рисунок *1* – Интерфейс приложения

Программа предусматривает обработку ситуаций, когда вводится некорректное число (рисунок 2).

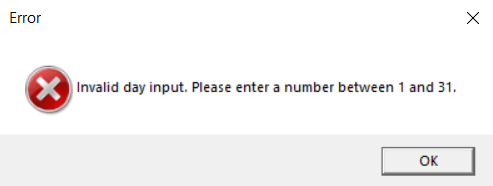


Рисунок *2* – Некорректный ввод

# **ВЫВОДЫ**

В результате выполнения лабораторной работы были изучены основы программирования в Win 32 API. Было создано простое оконное приложение, реализующее минимальный достаточный функционал, который позволяет обрабатывать основные оконные сообщения. Приложение позволяет вводить число, выбирать месяц и выводить полученную дату на экран. Так же была предусмотрена обработка некорректных входных данных.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Основы программирования для Win32 API [Электронный ресурс]. – Режим доступа:  [https://dims.karelia.ru/win32/](https://learn.microsoft.com/ru-ru/windows/win32/api%20–%20Дата%20доступа%2023.09.2023).
2. Модуль 1. Ваша первая программа для Windows [Электронный ресурс]. – Режим доступа:  [https://learn.microsoft.com/en-us/windows/win32/learnwin32/your-first-windows-program](https://learn.microsoft.com/ru-ru/windows/win32/api%20–%20Дата%20доступа%2023.09.2023).

# **ПРИЛОЖЕНИЕ А (обязательное) Листинг кода**

Листинг 1 **–** Файл Lab\_1.cpp:

#include <windows.h>

#include <stdlib.h>

#include <regex>

#define GENERATE\_BUTTON 1

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

void AddMenus(HWND hWnd);

void AddControls(HWND hWnd);

void FillMonths();

bool Validate(std::wregex pattern, wchar\_t string[], int size, const wchar\_t err\_msg[], HWND hWnd, HWND hCustom);

HMENU hMenu;

HWND hDay, hMonth, hDate, hMonthList;

int WINAPI WinMain(HINSTANCE hInst, HINSTANCE hPrevInst, LPSTR arts, int ncmdshow) {

WNDCLASSW wc = { 0 };

wc.hbrBackground = (HBRUSH)COLOR\_WINDOW;

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

wc.hInstance = hInst;

wc.lpszClassName = L"Class";

wc.lpfnWndProc = WindowProcedure;

if (!RegisterClassW(&wc))

return -1;

CreateWindowW(L"Class", L"Lab\_1", WS\_OVERLAPPEDWINDOW | WS\_VISIBLE, 100, 100, 500, 500, NULL, NULL, NULL, NULL);

MSG msg = { 0 };

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

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return 0;

}

LRESULT CALLBACK WindowProcedure(HWND hWnd, UINT msg, WPARAM wp, LPARAM lp) {

switch (msg)

{

case WM\_COMMAND:

switch (wp)

{

case GENERATE\_BUTTON:

wchar\_t day[100], month[12], date[30];

std::wregex day\_pattern(L"^(0?[1-9]|[12][0-9]|3[01])$");

if (!Validate(day\_pattern, day, 100, L"Invalid day input. Please enter a number between 1 and 31.", hWnd, hDay))

break;

int dayAsInt = \_wtoi(day);

int selectedMonthIndex = SendMessageW(hMonthList, CB\_GETCURSEL, 0, 0);

SendMessageW(hMonthList, CB\_GETLBTEXT, selectedMonthIndex, (LPARAM)month);

if (wcslen(day) > 0 && wcslen(month) > 0) {

wcscpy\_s(date, L"Today is ");

wcscat\_s(date, day);

wcscat\_s(date, L" of ");

wcscat\_s(date, month);

}

else {

wcscpy\_s(date, L"Empty field(s) exist");

}

SetWindowText(hDate, date);

break;

}

break;

case WM\_CREATE:

AddMenus(hWnd);

AddControls(hWnd);

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

return DefWindowProcW(hWnd, msg, wp, lp);

}

}

void AddMenus(HWND hWnd) {

hMenu = CreateMenu();

SetMenu(hWnd, hMenu);

}

void AddControls(HWND hWnd) {

CreateWindowW(L"Static", L"Day (1 - 31):", WS\_VISIBLE | WS\_CHILD, 20, 40, 78, 38, hWnd, NULL, NULL, NULL);

hDay = CreateWindowW(L"Edit", L"", WS\_VISIBLE | WS\_CHILD | WS\_BORDER, 110, 40, 58, 38, hWnd, NULL, NULL, NULL);

CreateWindowW(L"Static", L"Month:", WS\_VISIBLE | WS\_CHILD, 20, 90, 58, 38, hWnd, NULL, NULL, NULL);

hMonthList = CreateWindowW(L"ComboBox", NULL, WS\_VISIBLE | WS\_CHILD | CBS\_DROPDOWNLIST, 90, 90, 110, 250, hWnd, NULL, NULL, NULL);

FillMonths();

CreateWindowW(L"Button", L"Generate Date", WS\_VISIBLE | WS\_CHILD, 20, 140, 98, 38, hWnd, (HMENU)GENERATE\_BUTTON, NULL, NULL);

hDate = CreateWindowW(L"Edit", L"", WS\_VISIBLE | WS\_CHILD | WS\_BORDER, 20, 190, 180, 20, hWnd, NULL, NULL, NULL);

}

void FillMonths() {

const wchar\_t\* months[] = { L"January", L"February", L"March", L"April", L"May", L"June", L"July", L"August", L"September", L"October", L"November", L"December" };

for (int i = 0; i < 12; i++) {

SendMessageW(hMonthList, CB\_ADDSTRING, 0, (LPARAM)months[i]);

}

SendMessageW(hMonthList, CB\_SETCURSEL, 0, 0);

}

bool Validate(std::wregex pattern, wchar\_t string[], int size, const wchar\_t err\_msg[], HWND hWnd, HWND hCustom) {

GetWindowText(hCustom, string, size);

if (!std::regex\_match(string, pattern)) {

MessageBox(hWnd, err\_msg, L"Error", MB\_ICONERROR);

SetWindowText(hCustom, L"");

return false;

}

return true;

}