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**

Студент Д. С. Кончик

Преподаватель Н. Ю. Гриценко

Минск 2023

СОДЕРЖАНИЕ

[1 Цель работы 3](#_Toc146752068)

[2 Теоретические сведения 4](#_Toc146752069)

[3 Результат выполнения 6](#_Toc146752070)

[Заключение 8](#_Toc146752071)

[Список использованных источников 9](#_Toc146752072)

[Приложение А (обязательное) Листинг кода 10](#_Toc146752073)

1. **ЦЕЛЬ РАБОТЫ**
2. Изучение основных принципов работы с Win32 API.
3. Обработка основных оконных сообщений (создание и удаление окна, сообщения управляющих элементов).
4. Разработка оконного приложения с минимальной функциональной достаточностью – приложение для перевода чисел между системами счисления (2, 10, 16).
5. **ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

В операционной системе Windows реализована объектноориентированная идеология. Базовый объект системы – окно, поведение которого определяется методом, называемым *функцией окна*. Графический образ окна на экране дисплея – прямоугольная рабочая область.

Независимо от своего типа любой объект Windows идентифицируется описателем или дескриптором (handle). *Дескриптор* – это ссылка на объект. Все взаимоотношения программного кода с объектом осуществляются только через его дескриптор.

Интерфейс прикладного программирования (API – Application Programming Interface) представляет собой совокупность 32-битных функций (Win32 API), которые предназначены для создания приложений (программ), работающих под управлением Microsoft Windows. Функции объявлены в заголовочных файлах. Главный из них − файл *windows.h*, в котором содержатся ссылки на другие заголовочные файлы.

*Окно* – это прямоугольная область экрана, в котором приложение отображает информацию и получает реакцию от пользователя. Одновременно на экране может отображаться несколько окон, в том числе, окон других приложений, однако лишь одно из них может получать реакцию от пользователя – *активное окно*. Пользователь использует клавиатуру, мышь и прочие устройства ввода для взаимодействия с приложением, которому принадлежит активное окно.

Каждое 32-битное приложение создает, по крайней мере, одно окно, называемое *главным окном*, которое обеспечивает пользователя основным интерфейсом взаимодействия с приложением. Кроме главного окна, приложение может использовать еще и другие типы окон: управляющие элементы, диалоговые окна, окна-сообщения.

*Управляющий элемент* – окно, непосредственно обеспечивающее тот или иной способ ввода информации пользователем. К управляющим элементам относятся: кнопки, поля ввода, списки, полосы прокрутки и т.п. Управляющие элементы обычно находятся в каком-либо диалоговом окне.

*Диалоговое окно* – это временное окно, содержащее управляющие элементы, обычно использующееся для получения дополнительной информации от пользователя. Диалоговые окна бывают модальные и немодальные. *Модальное* диалоговое окно требует, чтобы пользователь обязательно ввел обозначенную в окне информацию и закрыл окно прежде, чем приложение продолжит работу. *Немодальное* диалоговое окно позволяет пользователю, не закрывая диалогового окна, переключаться на другие окна этого приложения.

*Окно-сообщение* – это диалоговое окно предопределенного системой формата, предназначенное для вывода небольшого текстового сообщения с одной или несколькими кнопками.

В отличие от традиционного программирования на основе линейных алгоритмов, программы для Windows строятся по принципам событийно-управляемого программирования – стиля программирования, при котором поведение компонента системы определяется набором возможных внешних событий и ответных реакций компонента на них. Такими компонентами в Windows являются окна. С каждым окном в Windows связана определенная функция обработки событий. События для окон называются *сообщениями*. Сообщение относится к тому или иному типу, идентифицируемому 32-битным целым числом (например, *WM\_COMMAND*, *WM\_CREATE* и *WM\_DESTROY*), и сопровождается парой 32-битных параметров (*WPARAM* и *LPARAM*), интерпретация которых зависит от типа сообщения.

Каждое окно принадлежит определенному *классу окон*. Окна одного класса имеют схожий вид, обслуживаются общей процедурой обработки событий, имеют одинаковые иконки и меню. Обычно каждое приложение создает для главного окна программы свой класс. Если приложению требуются дополнительные нестандартные окна, оно регистрирует другие классы. Стандартные диалоги и управляющие элементы принадлежат к предопределенным классам окон, для них не надо регистрировать новые классы.

Управляющие элементы, как и другие окна, принадлежат тому или иному классу окон. Windows предоставляет несколько предопределенных классов управляющих элементов. Программа может создавать управляющие элементы поштучно при помощи функции *CreateWindow* или оптом, загружая их вместе с шаблоном диалога из своих ресурсов. Управляющие элементы – это всегда дочерние окна. Управляющие элементы при возникновении некоторых событий, связанных с реакцией пользователя, посылают своему родительскому окну *сообщения-оповещения*.

1. **РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ**

В результате выполнения лабораторной работы было создано приложение для перевода чисел между системами счисления (рисунок 1).

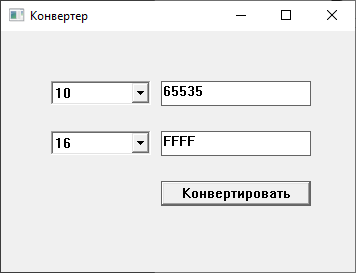


Рисунок 1 – Окно приложения

По нажатию кнопки *«Конвертировать»* происходит конвертация числа, указанного в верхнем поле для ввода, из одной системы счисления в другую.

Для выбора исходной и конечной системы счисления предусмотрены выпадающие списки (рисунок 2).

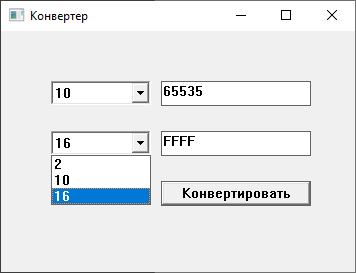


Рисунок 2– Выбор системы счисления

Приложение работает только с положительными числами размером до 16 бит. В случае ввода некорректного значения в поле, которое предназначено для исходного числа, и нажатия кнопки *«Конвертировать»* происходит появление окна-сообщения с требованиями к исходному числу и вывод во второе поле надписи *«Некорректный ввод»* (рисунки 3, 4).

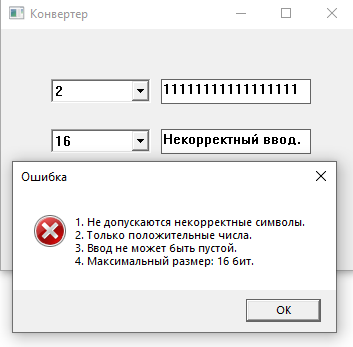


Рисунок 3– Введено 17-битное число

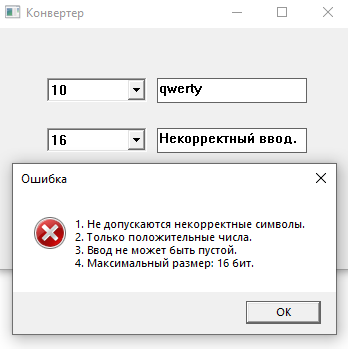
**

Рисунок 4– Введено некорректное значение

# ЗАКЛЮЧЕНИЕ

В результате лабораторной работы были изучены основные принципы работы с Win32 API: виды окон, классы окон и их регистрация, обработка сообщений разных типов. Было создано оконное приложение с минимальной функциональной достаточностью – приложение для перевода чисел между системами счисления с возможностью выбора исходной и конечной системы, а также обработкой некорректного ввода: слишком большие числа, некорректный символы в поле для ввода числа.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Справочник по программированию для API Win32 [Электронный ресурс]. – Режим доступа: <https://learn.microsoft.com/ru-ru/windows/win32/api/>.
2. Безруков В.А. Win32 API. Программирование / учебное пособие. –СПб: СПбГУ ИТМО, 2009. – 90 с.
3. Основы программирования для Win32 API [Электронный ресурс]. – Режим доступа: https://dims.karelia.ru/win32/.

# ПРИЛОЖЕНИЕ А (обязательное) Листинг кода

Листинг 1 – Файл Source.cpp

#include <windows.h>

#include <string>

#include <regex>

#include "Converters.h"

#include "Constants.h"

using namespace std;

int selected\_index\_combobox1 = 1;

int selected\_index\_combobox2 = 0;

void MainWindowAddComboBoxes(HWND hWnd);

void MainWindowAddEdits(HWND hWnd);

void MainWindowAddButtons(HWND hWnd);

LRESULT CALLBACK MainWindowProcedure(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam);

int WINAPI WinMain(HINSTANCE hInst, HINSTANCE, LPSTR, int ss) {

// Создание класса главного окна

WNDCLASS wc;

wc.style = NULL;

wc.lpfnWndProc = MainWindowProcedure;

wc.cbClsExtra = NULL;

wc.cbWndExtra = NULL;

wc.hInstance = hInst;

wc.hIcon = NULL;

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

wc.hbrBackground = (HBRUSH)(COLOR\_WINDOW);

wc.lpszMenuName = NULL;

wc.lpszClassName = L"MainWindowClass";

if (!RegisterClass(&wc))

return EXIT\_FAILURE;

// Создание главного окна

HWND hw = CreateWindow(

L"MainWindowClass", L"Конвертер",

WS\_OVERLAPPEDWINDOW,

0, 0, 370, 280,

NULL, NULL, hInst, NULL

);

if (!hw) return EXIT\_FAILURE;

ShowWindow(hw, ss);

// Цикл обработки сообщений

MSG msg;

while (GetMessageW(&msg, NULL, 0, 0)) {

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return msg.wParam;

}

void MainWindowAddComboBoxes(HWND hWnd) {

HWND hComboBox1 = CreateWindow(

L"COMBOBOX", L"",

WS\_CHILD | WS\_VISIBLE | CBS\_DROPDOWNLIST,

50, 50, 100, 200,

hWnd,

(HMENU)ID\_COMBOBOX1,

NULL,

NULL

);

for (auto base : select\_bases)

SendMessage(hComboBox1, CB\_ADDSTRING, 0, reinterpret\_cast<LPARAM>(to\_wstring(base).c\_str()));

SendMessage(hComboBox1, CB\_SETCURSEL, selected\_index\_combobox1, 0);

HWND hComboBox2 = CreateWindow(

L"COMBOBOX", L"",

WS\_CHILD | WS\_VISIBLE | CBS\_DROPDOWNLIST,

50, 100, 100, 200,

hWnd,

(HMENU)ID\_COMBOBOX2,

NULL,

NULL

);

for (auto base : select\_bases)

SendMessage(hComboBox2, CB\_ADDSTRING, 0, reinterpret\_cast<LPARAM>(to\_wstring(base).c\_str()));

SendMessage(hComboBox2, CB\_SETCURSEL, selected\_index\_combobox2, 0);

}

void MainWindowAddEdits(HWND hWnd) {

HWND hEdit1 = CreateWindow(

L"EDIT", L"",

WS\_CHILD | WS\_VISIBLE | WS\_BORDER | ES\_AUTOHSCROLL,

160, 50, 150, 25,

hWnd,

(HMENU)ID\_EDIT1,

NULL,

NULL

);

HWND hEdit2 = CreateWindow(

L"EDIT", L"",

WS\_CHILD | WS\_VISIBLE | WS\_BORDER | ES\_AUTOHSCROLL,

160, 100, 150, 25,

hWnd,

(HMENU)ID\_EDIT2,

NULL, NULL

);

}

void MainWindowAddButtons(HWND hWnd) {

HWND hButton = CreateWindow(

L"BUTTON", L"Конвертировать",

WS\_CHILD | WS\_VISIBLE | BS\_DEFPUSHBUTTON,

160, 150, 150, 25,

hWnd,

(HMENU)ID\_BUTTON\_CONVERT,

NULL, NULL

);

}

// Функция обработки сообщений

LRESULT CALLBACK MainWindowProcedure(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam) {

switch (msg)

{

case WM\_CREATE:

{

MainWindowAddComboBoxes(hWnd);

MainWindowAddEdits(hWnd);

MainWindowAddButtons(hWnd);

break;

}

// LOWORD(wParam) - Идентификатор элемента управления

// HIWORD(wParam) - Код уведомления, определяемый элементом управления

// lParam - Дескриптор окна управления

// https://learn.microsoft.com/ru-ru/windows/win32/menurc/wm-command

case WM\_COMMAND:

{

switch (LOWORD(wParam))

{

case ID\_COMBOBOX1:

if (HIWORD(wParam) == CBN\_SELCHANGE)

selected\_index\_combobox1 = SendMessage((HWND)lParam, CB\_GETCURSEL, 0, 0);

break;

case ID\_COMBOBOX2:

if (HIWORD(wParam) == CBN\_SELCHANGE)

selected\_index\_combobox2 = SendMessage((HWND)lParam, CB\_GETCURSEL, 0, 0);

break;

case ID\_BUTTON\_CONVERT:

if (HIWORD(wParam) == BN\_CLICKED)

{

WCHAR readFirst[50];

GetWindowText(GetDlgItem(hWnd, ID\_EDIT1), readFirst, sizeof(readFirst) / sizeof(readFirst[0]));

wstring w\_inputFirst(readFirst);

string inputFirst = std::string(w\_inputFirst.begin(), w\_inputFirst.end());

int selected\_base = select\_bases[selected\_index\_combobox1];

if (

(selected\_base == 2 && !regex\_match(inputFirst, binary\_pattern)) ||

(selected\_base == 10 && (!regex\_match(inputFirst, decimal\_pattern) || stoi(inputFirst) > 65535)) ||

(selected\_base == 16 && !regex\_match(inputFirst, hexadecimal\_pattern))

)

{

MessageBox(hWnd, L"1. Не допускаются некорректные символы.\n2. Только положительные числа.\n3. Ввод не может быть пустой. \n4. Максимальный размер: 16 бит.", L"Ошибка", MB\_ICONERROR | MB\_OK);

SetWindowText(GetDlgItem(hWnd, ID\_EDIT2), L"Некорректный ввод.");

}

else {

try {

string temp = baseToDecimal(inputFirst, select\_bases[selected\_index\_combobox1]);

string result = decimalToBase(temp, select\_bases[selected\_index\_combobox2]);

SetWindowText(GetDlgItem(hWnd, ID\_EDIT2), wstring(result.begin(), result.end()).c\_str());

}

catch (exception e) {

MessageBox(hWnd, L"Ошибка конвертации.", L"Ошибка", MB\_ICONERROR | MB\_OK);

SetWindowText(GetDlgItem(hWnd, ID\_EDIT2), L"Ошибка конвертации.");

}

}

}

break;

}

break;

}

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

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

}

}

Листинг 2 – Файл Constants.h

#pragma once

#include <map>

#include <string>

#include <regex>

using namespace std;

#define ID\_COMBOBOX1 1

#define ID\_COMBOBOX2 2

#define ID\_EDIT1 3

#define ID\_EDIT2 4

#define ID\_BUTTON\_CONVERT 5

int select\_bases[] = { 2, 10, 16 };

regex binary\_pattern("^[01]{1,16}$");

regex decimal\_pattern("^[0-9]{1,5}$");

regex hexadecimal\_pattern("^[0-9A-F]{1,4}$");.

Листинг 3 – Файл Converters.h

#pragma once

#include <string>

#include <cmath>

using namespace std;

// Функция для перевода из десятичной системы в другие системы счисления

string decimalToBase(const string& numStr, int base) {

int num = stoi(numStr);

if (num == 0)

return "0";

string result = "";

while (num > 0) {

int remainder = num % base;

if (remainder < 10) {

result = char('0' + remainder) + result;

}

else {

result = char('A' + remainder - 10) + result;

}

num /= base;

}

return result;

}

// Функция для перевода из других систем счисления в десятичную

string baseToDecimal(const string& numStr, int base) {

int result = 0;

int power = 0;

for (int i = numStr.length() - 1; i >= 0; i--) {

int digit;

if (numStr[i] >= '0' && numStr[i] <= '9') {

digit = numStr[i] - '0';

}

else if (numStr[i] >= 'A' && numStr[i] <= 'F') {

digit = numStr[i] - 'A' + 10;

}

else {

return "Неверный символ в символе: " + numStr[i];

}

result += digit \* pow(base, power);

power++;

}

return to\_string(result);

}