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 А. С. Сорокин

Преподаватель Н. Ю. Гриценко

Минск 2023

СОДЕРЖАНИЕ

[1 Цель работы 3](#_Toc146752068)

[2 Теоретические сведения 4](#_Toc146752069)

[3 Результат выполнения 6](#_Toc146752070)

[Заключение 7](#_Toc146752071)

[Список использованных источников 8](#_Toc146752072)

[Приложение А (обязательное) Листинг кода 9](#_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 – Редактор подсчета символов.
5. **ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Win32 API (Application Programming Interface) - это набор функций, структур и констант, предоставляемых операционной системой Windows для разработки приложений под Windows. Программирование в Win32 API позволяет создавать нативные приложения для Windows, имеющие полный контроль над ресурсами и функциональностью операционной системы.

Для выполнения данной лабораторной работы, были использованы следующие теоретические сведения и концепции:

*Окно* (Window): Окно является основной сущностью в Win32 API и представляет графическую область на экране. Окна могут иметь различные стили (например, перекрывающиеся, фиксированные размеры и т. д.) и могут взаимодействовать с пользователем (обрабатывать сообщения, получать ввод и т. д.).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.

*Процедура обработки окна* (Window Procedure): Каждое окно имеет свою процедуру обработки (Window Procedure), которая обрабатывает сообщения, поступающие от операционной системы. Процедура обработки окна определяет поведение окна, например, обрабатывает сообщения о событиях (нажатия клавиш, движения мыши и т. д.) и выполняет соответствующие действия.

*Управляющие элементы* (Controls): Управляющие элементы, такие как кнопки, полосы прокрутки, текстовые поля и другие элементы интерфейса, могут быть добавлены в окно. Они предоставляют дополнительную функциональность и взаимодействие с пользователем.

*Ресурсы* (Resources): Ресурсы представляют собой внешние файлы, содержащие данные, такие как иконки, курсоры, изображения, строки и другие ресурсы, используемые в приложении. Ресурсы могут быть включены в исполняемый файл или загружены динамически.

*Обработка меню*: для создания меню и обработки команд меню использовалась функция CreateMenu и связанные с ней функции для добавления пунктов меню и обработки выбора команд.

*Класс окна* (Window Class): Класс окна определяет атрибуты и поведение окна, такие как иконка, фон, курсор, процедура обработки окна и другие свойства. При создании окна необходимо зарегистрировать класс окна с помощью функции RegisterClass.

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 ходе выполнения лабораторной работы было разработано приложения для подсчета символов во введенной строке (рисунок 1).

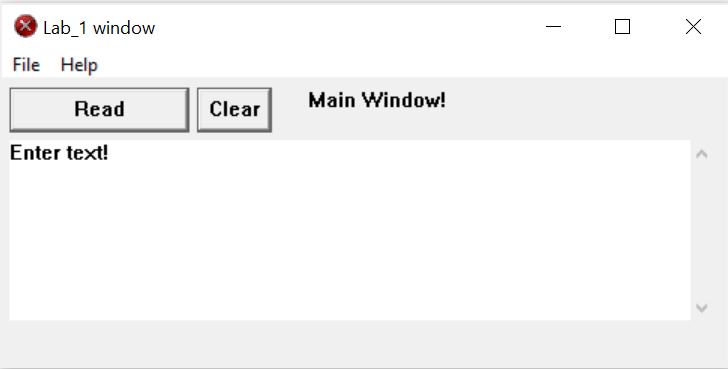


Рисунок 1 – Окно приложения

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

Пользователь может нажать на кнопку “Read”, чтобы подсчитать количество символов во введенной строке (рисунок 2).

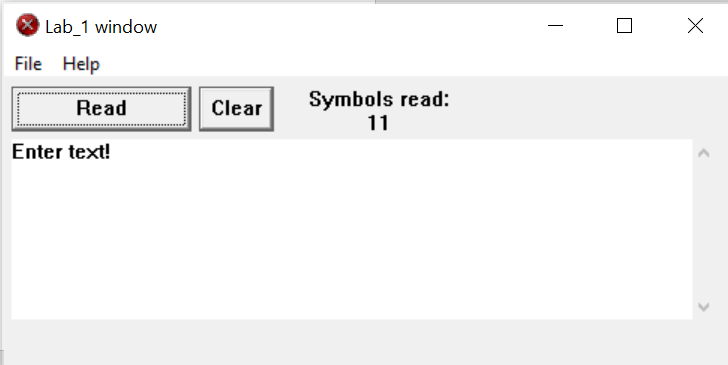


Рисунок 2– Подсчет символов

# ЗАКЛЮЧЕНИЕ

В результате выполнения работы было создано оконное приложение, способное считать введенные пользователем. Была создана область для отображения текста с возможностью прокрутки, а также добавлены кнопки управления для выполнения операций над текстовым полем. В ходе обработки основных оконных сообщений были реализованы необходимые функции для работы с текстовым полем. Таким образом, лабораторная работа позволила ознакомиться с основами программирования в Win32 API и создать оконное приложение с базовой функциональностью, отражающей основные принципы работы с окнами и обработкой сообщений.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Справочник по программированию для API Win32 [Электронный ресурс]. – Режим доступа: <https://learn.microsoft.com/ru-ru/windows/win32/api/>.
2. Основы программирования для Win32 API [Электронный ресурс]. – Режим доступа: https://dims.karelia.ru/win32/.

# ПРИЛОЖЕНИЕ А (обязательное) Листинг кода

Листинг 1 – Файл main.cpp

#ifndef UNICODE

#define UNICODE

#endif

#include <windows.h>

#include "SoftwareDefinitions.h"

#include <string>

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

WNDCLASS SoftwareMainClass = NewWindowClass((HBRUSH)COLOR\_WINDOW, LoadCursor(NULL, IDC\_HAND), hInst, LoadIcon(NULL, IDI\_ERROR),

L"MainWndClass", SoftwareMainProcedure);

if (!RegisterClassW(&SoftwareMainClass)) { return -1; }

CreateWindow(L"MainWndClass", L"Lab\_1 window", WS\_OVERLAPPEDWINDOW | WS\_VISIBLE, 100, 100, 500, 250, NULL, NULL, NULL, NULL);

MSG msg = { };

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

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return 0;

}

WNDCLASS NewWindowClass(HBRUSH BGColor, HCURSOR Cursor, HINSTANCE hInst, HICON Icon, LPCWSTR Name, WNDPROC Procedure) {

WNDCLASS NWC = { 0 };

NWC.hIcon = Icon;

NWC.hCursor = Cursor;

NWC.hInstance = hInst;

NWC.lpszClassName = Name;

NWC.hbrBackground = BGColor;

NWC.lpfnWndProc = Procedure;

return NWC;

}

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

switch (msg)

{

case WM\_COMMAND:

switch (wp) {

case OnMenuClicked1:

MessageBoxA(hWnd, "menu 1 was klicked", "Menu worked", MB\_OK);

break;

case OnClear:

SetWindowTextA(hEditControl, "");

break;

case OnRead:

readChars = GetWindowTextA(hEditControl, Buffer, TextBufferSize);

//SetWindowTextA(hStaticControl, Buffer);

SetWindowTextA(hStaticControl, ("Symbols read: " + std::to\_string(readChars)).c\_str());

break;

case OnExitSoftware:

PostQuitMessage(0);

break;

}

break;

case WM\_CREATE:

MainWndAddMenus(hWnd);

MainWndAddWidgets(hWnd);

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

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

}

}

void MainWndAddMenus(HWND hwnd) {

HMENU RootMenu = CreateMenu();

HMENU SubMenu = CreateMenu();

//HMENU SubActionMenu = CreateMenu();

//AppendMenu(SubActionMenu, MF\_STRING, OnMenuClicked3, L"Menu 3");

//AppendMenu(SubMenu, MF\_POPUP, (UINT\_PTR)SubActionMenu, L"Action");

AppendMenu(SubMenu, MF\_STRING, OnClear, L"Clear");

AppendMenu(SubMenu, MF\_SEPARATOR, NULL, NULL);

AppendMenu(SubMenu, MF\_STRING, OnExitSoftware, L"Exit");

AppendMenu(RootMenu, MF\_POPUP, (UINT\_PTR)SubMenu, L"File");

AppendMenu(RootMenu, MF\_STRING, (UINT\_PTR)SubMenu, L"Help");

SetMenu(hwnd, RootMenu);

}

void MainWndAddWidgets(HWND hWnd) {

hStaticControl = CreateWindowA("static", "Main Window!", WS\_VISIBLE | WS\_CHILD | ES\_CENTER, 200, 5, 100, 30, hWnd, NULL, NULL, NULL);

hEditControl = CreateWindowA("edit", "Enter text!", WS\_VISIBLE | WS\_CHILD | ES\_MULTILINE | WS\_VSCROLL, 5, 40, 470, 120, hWnd, NULL, NULL, NULL);

CreateWindowA("button", "Clear", WS\_VISIBLE | WS\_CHILD | ES\_CENTER, 130, 5, 50, 30, hWnd, (HMENU)OnClear , NULL, NULL);

CreateWindowA("button", "Read", WS\_VISIBLE | WS\_CHILD | ES\_CENTER, 5, 5, 120, 30, hWnd, (HMENU)OnRead, NULL, NULL);

}

Листинг 2 – Файл SoftwareDefinitions.h

#pragma once

#define OnMenuClicked1 1

#define OnMenuClicked2 2

#define OnMenuClicked3 3

#define OnExitSoftware 4

#define OnClear 5

#define OnRead 6

#define TextBufferSize 256

char Buffer[TextBufferSize];

HWND hEditControl;

HWND hStaticControl;

int readChars;

LRESULT CALLBACK SoftwareMainProcedure(HWND hWnd, UINT msg, WPARAM wp, LPARAM lp);

WNDCLASS NewWindowClass(HBRUSH BGColor, HCURSOR Cursor, HINSTANCE hInst, HICON Icon, LPCWSTR Name, WNDPROC Procedure);

void MainWndAddMenus(HWND hwnd);

void MainWndAddWidgets(HWND hwnd);