Министерство науки и высшего образования РФ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Курский государственный университет»

Кафедра программного обеспечения и администрирования информационных систем

Направление подготовки математическое обеспечение и администрирование информационных систем

Форма обучения очная

**Отчет**

**по лабораторной работе №8**

«Оконные приложения Windows»

дисциплина «Операционные системы и оболочки»

Выполнил:

студент группы 313.1 Козявин М.С.

Проверил:

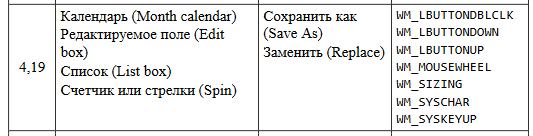
к.т.н., доцент кафедры ПОиАИС Кривонос А.В.

Курск, 2023

**Цели работы:**

Получение практических навыков создания оконных приложений Windows на языке C/C++ с применением Win32 API.

**Вариант 4**

****

**Задание:**

1. Изучить элементы управления общего пользования, указанные в варианте задания. Особое внимание уделить тому, какие стили они поддерживают, какие сообщения принимают и какие уведомления отправляют родительскому окну. Включить в отчет назначение и описание изученных элементов управления.
2. Изучить диалоговые окна общего пользования, указанные в варианте задания. Включить в отчет назначение и описание изученных диалоговых окон.
3. Изучить оконное сообщение WM\_TIMER и сообщения, указанные в варианте задания. Включить в отчет описание изученных оконных сообщений.
4. Разработать в Visual C++ оконное приложение Win32, которое:
   1. должно создавать главное окно, содержащее меню и элементы управления, указанные в варианте задания;
   2. должно обрабатывать комбинации быстрых клавиш;
   3. должно создавать одно или несколько диалоговых окон с использованием шаблона;
   4. должно создавать диалоговое окно сообщений;
   5. должно создавать диалоговые окна, указанные в варианте задания;
   6. должно обрабатывать оконное сообщение WM\_TIMER и оконные сообщения, указанные в варианте задания.
5. Протестировать работу приложения, разработанного в п. 4, на компьютере под управлением Windows XP (или выше). Результаты тестирования отразить в отчете.
6. Включить в отчет исходный программный код и выводы о проделанной работе.

**Элементы управления:**

**Month calendar** – элемент выбора даты в виде календаря на месяц.

MCN\_SELECT – код уведомления о изменении пользователем выбора даты.

**Edit box** – элемент ввода текста пользователем.

ES\_AUTOHSCROLL – стиль автоматической горизонтальной прокрутки поля ввода.

ES\_LEFT – стиль выравнивания по левому краю.

**List box** – элемент списка из которого пользователь выбирает один или несколько элементов.

LBS\_NOTIFY – стиль, позволяющий списку отправлять уведомления

LBN\_SELCHANGE – код уведомления о изменении пользователем выбранного элемента.

LB\_GETCURSEL – сообщение, позволяющее получить выбранный элемент списка.

LB\_ADDSTRING – сообщение, добавить элемент в список.

LB\_RESETCONTENT – сообщение, позволяющее очистить список.

LB\_SETCURSEL – сообщение, позволяющее выбрать нужный элемент в списке.

LB\_FINDSTRING – сообщение, позволяющее найти в списке элемент начинающийся на определённые символы.

**Spin** – элемент управления представляющий собой две стрелки для инкремента или декремента значения.

UDN\_DELTAPOS – уведомление, сообщающее, что позиция была изменена.

**Диалоговые окна общего пользования:**

**Save as** – диалоговое окно, позволяющее указать путь сохранения файла.

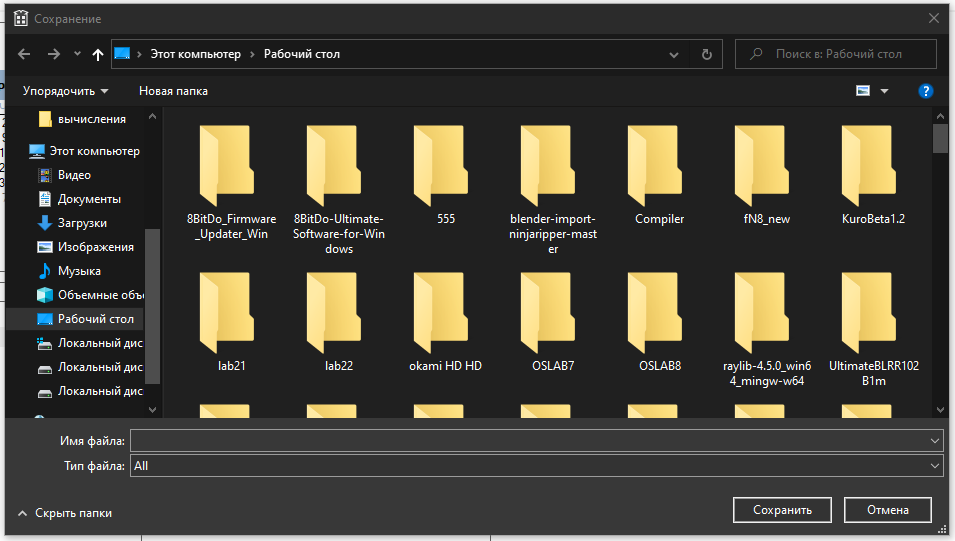
****

Рисунок 1 – Диалоговое окно Save as

**Replace** – диалоговое окно, найти информацию в приложении и заменить её.

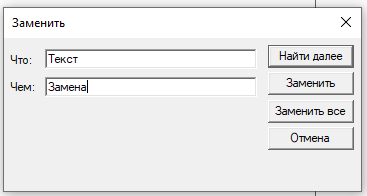
****

Рисунок 2 – Диалоговое окно Replace

Используются стили FR\_HIDEUPDOWN, FR\_HIDEMATCHCASE, FR\_HIDEWHOLEWORD, FR\_NOMATCHCASE, FR\_NOUPDOWN, FR\_NOWHOLEWORD убирающие функции следующего совпадения, поиска полного слова, совпадения по регистру букв.

**Сообщения:**

WM\_TIMER – срабатывание таймера. Используется в приложении для вывода диалогового окна напоминания на экран.

WM\_LBUTTONDBLCLK – двойной клик левой кнопкой мыши. Используется в приложении для удаления элементов из списка.

WM\_LBUTTONDOWN – нажатие левой кнопки мыши. Используется в приложении для выделения прямоугольной области.

WM\_LBUTTONUP – отпускание левой кнопки мыши. Используется в приложении для выделения прямоугольной области.

WM\_MOUSEWHEEL – вращение колёсика мыши. Используется в приложении для навигации по списку.

WM\_SIZING – изменение размера окна. Используется для масштабирования списка по вертикали.

WM\_SYSCHAR – нажатие кнопки клавиатуры при зажатом ALT. Выдаётся код символа.

WM\_SYSKEYUP – отпускание кнопки клавиатуры при зажатом ALT. Выдаётся

код клавиши.

**Тестирование приложения:**

Приложение является списком дел с возможностью вывода напоминания через введённое количество секунд.

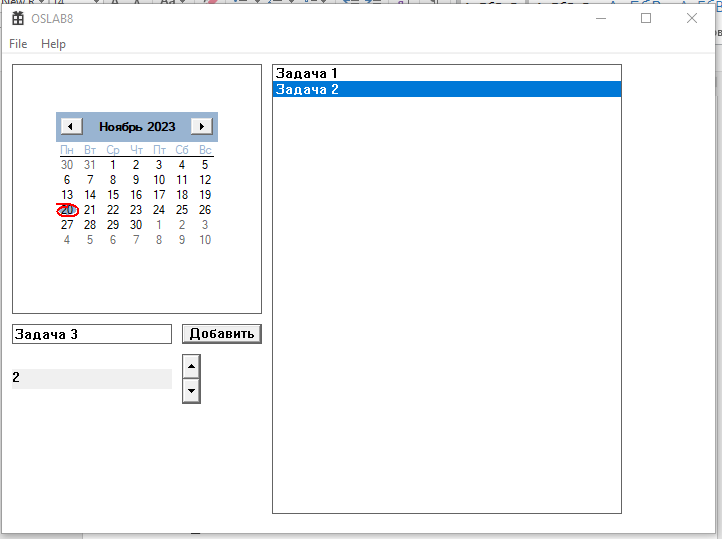


Рисунок 3 – Тест 1

Вывод напоминания происходит посредствам диалогового окна сообщения по истечении таймера.

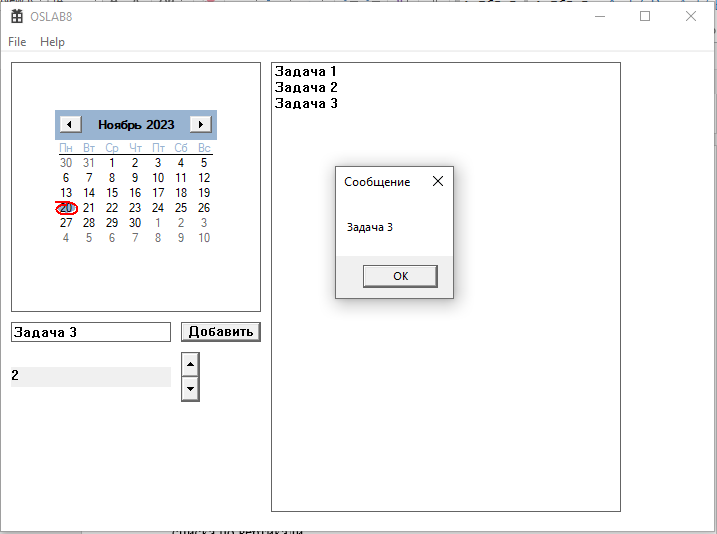


Рисунок 4 – Тест 2

В приложении реализовано меню, позволяющее перейти в диалоговые окна сохранения, замены и вывода информации о приложении. Диалоговое окно

вывода информации о приложении создано с помощью шаблона.

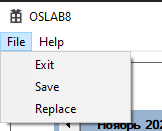


Рисунок 5 – Тест 3

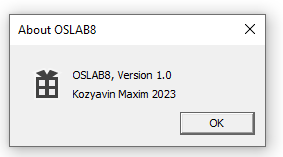


Рисунок 6 – Тест 4

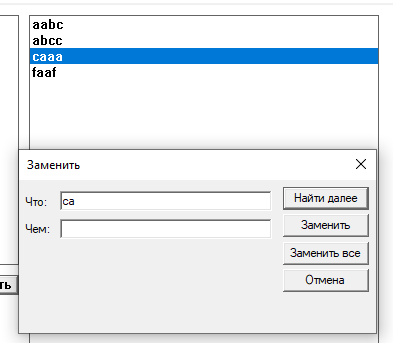


Рисунок 7 – Тест 5

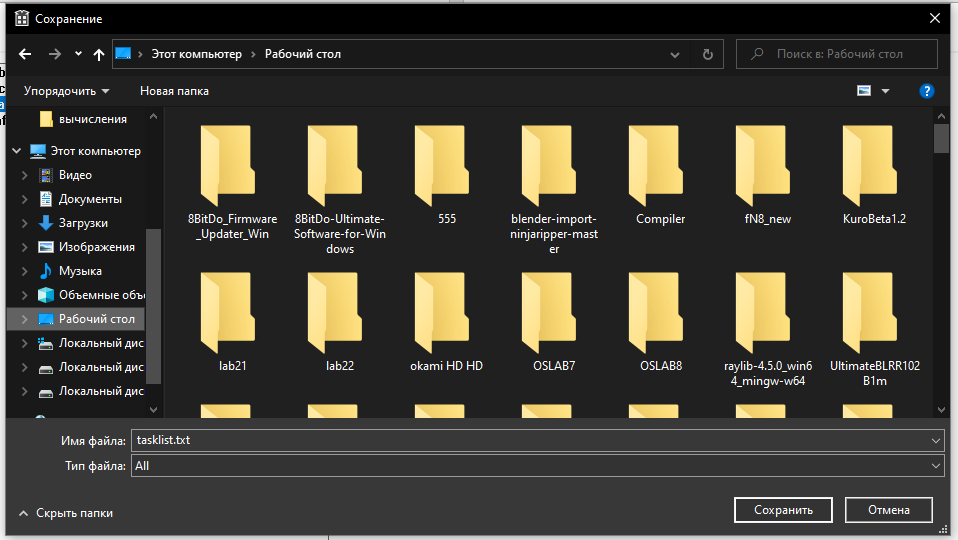


Рисунок 8 – Тест 6

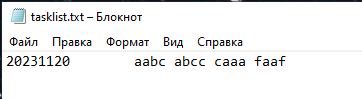


Рисунок 9 – Тест 7

Диалоговые окна так же возможно открыть посредствам быстрых клавиш CTRL + S, ALT + S, CTRL + F, ALT + F. Комбинации клавиш с использованием ALT реализованы посредствам сообщений WM\_SYSCHAR и WM\_SYSKEYUP.



Рисунок 10 – Тест 8

**Исходный код приложения:**

#pragma warning(disable : 4996)

#include "framework.h"

#include "OSLAB8.h"

#include <CommCtrl.h>

#include <vector>

#include <map>

#include <string>

#include <iostream>

#include <windowsx.h>

#include <commdlg.h>

#include <iostream>

#include <fstream>

#define MAX\_LOADSTRING 100

using namespace std;

map<wstring, vector<wstring>> todo\_data = map<wstring, vector<wstring>>();

vector<wstring> timer\_msg = vector<wstring>();

wstring curDate;

int curItem = 0;

RECT rcSelection;

BOOL bSelecting;

int timerTime = 0;

int tId = 0;

HINSTANCE hInst; // current instance

WCHAR szTitle[MAX\_LOADSTRING]; // The title bar text

WCHAR szWindowClass[MAX\_LOADSTRING]; // the main window class name

UINT uFindReplaceMsg; // message identifier for FINDMSGSTRING

WCHAR szFindWhat[100];

WCHAR szReplaceWith[100];

FINDREPLACE fr = {sizeof(FINDREPLACE)};

WCHAR szFile[260];

OPENFILENAME ofn = {sizeof(OPENFILENAME)};

HANDLE hf;

ATOM MyRegisterClass(HINSTANCE hInstance);

BOOL InitInstance(HINSTANCE, int);

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

void CALLBACK timerProc(HWND, UINT, UINT, DWORD);

INT\_PTR CALLBACK About(HWND, UINT, WPARAM, LPARAM);

INT\_PTR CALLBACK Save(HWND, UINT, WPARAM, LPARAM);

int print\_log(const char\* format, ...)

{

static char s\_printf\_buf[1024];

va\_list args;

va\_start(args, format);

\_vsnprintf(s\_printf\_buf, sizeof(s\_printf\_buf), format, args);

va\_end(args);

OutputDebugStringA(s\_printf\_buf);

return 0;

}

void DrawSelectionRect(HWND hWnd, const RECT\* pRect)

{

HDC hDC;

HPEN hOldPen;

HBRUSH hOldBrush;

hDC = GetDC(hWnd);

hOldPen = (HPEN)SelectObject(hDC, CreatePen(PS\_SOLID, 1, RGB(128, 128, 128)));

hOldBrush = (HBRUSH)SelectObject(hDC, GetStockObject(NULL\_BRUSH));

SetROP2(hDC, R2\_XORPEN);

Rectangle(hDC, pRect->left, pRect->top, pRect->right, pRect->bottom);

SelectObject(hDC, hOldBrush);

DeleteObject(SelectObject(hDC, hOldPen));

ReleaseDC(hWnd, hDC);

}

int APIENTRY wWinMain(\_In\_ HINSTANCE hInstance, \_In\_opt\_ HINSTANCE hPrevInstance, \_In\_ LPWSTR lpCmdLine, \_In\_ int nCmdShow)

{

UNREFERENCED\_PARAMETER(hPrevInstance);

UNREFERENCED\_PARAMETER(lpCmdLine);

LoadStringW(hInstance, IDS\_APP\_TITLE, szTitle, MAX\_LOADSTRING);

LoadStringW(hInstance, IDC\_OSLAB8, szWindowClass, MAX\_LOADSTRING);

MyRegisterClass(hInstance);

if (!InitInstance(hInstance, nCmdShow))

{

return FALSE;

}

HACCEL hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_OSLAB8));

MSG msg;

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

{

if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

return (int)msg.wParam;

}

ATOM MyRegisterClass(HINSTANCE hInstance)

{

WNDCLASSEXW wcex;

wcex.cbSize = sizeof(WNDCLASSEX);

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

wcex.lpfnWndProc = WndProc;

wcex.cbClsExtra = 0;

wcex.cbWndExtra = 0;

wcex.hInstance = hInstance;

wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_OSLAB8));

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

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

wcex.lpszMenuName = MAKEINTRESOURCEW(IDC\_OSLAB8);

wcex.lpszClassName = szWindowClass;

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

uFindReplaceMsg = RegisterWindowMessage(FINDMSGSTRING);

return RegisterClassExW(&wcex);

}

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)

{

hInst = hInstance;

HWND hWnd = CreateWindowW(szWindowClass, szTitle, WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0, nullptr, nullptr, hInstance, nullptr);

if (!hWnd)

{

return FALSE;

}

ShowWindow(hWnd, nCmdShow);

UpdateWindow(hWnd);

return TRUE;

}

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

{

if (message == uFindReplaceMsg)

{

LPFINDREPLACE lpfr = (LPFINDREPLACE)lParam;

if (lpfr->Flags & FR\_FINDNEXT)

{

HWND hwndList = GetDlgItem(hWnd, IDC\_LIST);

int id = SendMessage(hwndList, LB\_FINDSTRING, 0, (LPARAM)lpfr->lpstrFindWhat);

if (id != LB\_ERR)

{

curItem = id;

SendMessage(hwndList, LB\_SETCURSEL, id, NULL);

}

}

else if (lpfr->Flags & FR\_REPLACE)

{

HWND hwndList = GetDlgItem(hWnd, IDC\_LIST);

int id = SendMessage(hwndList, LB\_FINDSTRING, 0, (LPARAM)lpfr->lpstrFindWhat);

if (id != LB\_ERR)

{

curItem = id;

wstring wstr = wstring(lpfr->lpstrReplaceWith);

if (wstr.size() < 0)

{

ListBox\_DeleteString(hwndList, curItem);

todo\_data[curDate].erase(todo\_data[curDate].begin() + curItem);

curItem -= 1;

if (curItem < 0) curItem = 0;

}

else

{

ListBox\_DeleteString(hwndList, curItem);

ListBox\_InsertString(hwndList, id, lpfr->lpstrReplaceWith);

todo\_data[curDate][id] = wstr;

}

}

}

else if (lpfr->Flags & FR\_REPLACEALL)

{

HWND hwndList = GetDlgItem(hWnd, IDC\_LIST);

int id = SendMessage(hwndList, LB\_FINDSTRING, 0, (LPARAM)lpfr->lpstrFindWhat);

while (id != LB\_ERR)

{

curItem = id;

wstring wstr = wstring(lpfr->lpstrReplaceWith);

if (wstr.size() < 0)

{

ListBox\_DeleteString(hwndList, curItem);

todo\_data[curDate].erase(todo\_data[curDate].begin() + curItem);

curItem -= 1;

if (curItem < 0) curItem = 0;

}

else

{

ListBox\_DeleteString(hwndList, curItem);

ListBox\_InsertString(hwndList, id, lpfr->lpstrReplaceWith);

todo\_data[curDate][id] = wstr;

}

id = SendMessage(hwndList, LB\_FINDSTRING, 0, (LPARAM)lpfr->lpstrFindWhat);

}

}

}

switch (message)

{

case WM\_CREATE:

{

RECT rect;

GetWindowRect(hWnd, &rect);

CreateWindowEx(0, MONTHCAL\_CLASS, TEXT("Календарь"), WS\_BORDER | WS\_CHILD | WS\_VISIBLE | MCS\_NOTODAY,

10, 10, 250, 250, hWnd, (HMENU)IDC\_CAL, hInst, NULL);

CreateWindowEx(0, TEXT("LISTBOX"), TEXT("Список"), WS\_BORDER | WS\_CHILD | WS\_VISIBLE | LBS\_NOTIFY,

270, 10, 350, rect.bottom - rect.top - 80, hWnd, (HMENU)IDC\_LIST, hInst, NULL);

CreateWindowEx(0, TEXT("EDIT"), TEXT("Текст"), WS\_BORDER | ES\_AUTOHSCROLL | ES\_LEFT | WS\_CHILD | WS\_VISIBLE,

10, 270, 160, 20, hWnd, (HMENU)IDC\_EDIT, hInst, NULL);

CreateWindowEx(0, TEXT("STATIC"), TEXT("0"), WS\_CHILD | WS\_VISIBLE,

10, 315, 160, 20, hWnd, (HMENU)IDC\_TEXT, hInst, NULL);

CreateWindowEx(0, UPDOWN\_CLASS, TEXT("Вверх-вниз"), WS\_BORDER | WS\_CHILD | WS\_VISIBLE,

180, 300, 50, 50, hWnd, (HMENU)IDC\_UPDOWN, hInst, NULL);

CreateWindowEx(0, TEXT("BUTTON"), TEXT("Добавить"), WS\_BORDER | WS\_CHILD | WS\_VISIBLE,

180, 270, 80, 20, hWnd, (HMENU)IDC\_ADD, hInst, NULL);

HWND cal = GetDlgItem(hWnd, IDC\_CAL);

SYSTEMTIME time;

MonthCal\_GetToday(cal, &time);

curDate = to\_wstring(time.wYear) + to\_wstring(time.wMonth) + to\_wstring(time.wDay);

todo\_data[curDate] = vector<wstring>();

return TRUE;

}

case WM\_COMMAND:

{

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

{

int id = LOWORD(wParam);

if (id == IDC\_LIST)

{

HWND hwndList = GetDlgItem(hWnd, IDC\_LIST);

curItem = SendMessage(hwndList, LB\_GETCURSEL, curItem, NULL);

}

}

int wmId = LOWORD(wParam);

switch (wmId)

{

case IDC\_ADD:

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

HWND hwndEdit = GetDlgItem(hWnd, IDC\_EDIT);

TCHAR buff[100];

GetWindowText(hwndEdit, buff, 100);

todo\_data[curDate].push\_back(buff);

HWND hwndList = GetDlgItem(hWnd, IDC\_LIST);

SendMessage(hwndList, LB\_RESETCONTENT, 0, 0);

for (int i = 0; i < todo\_data[curDate].size(); i++) {

SendMessage(hwndList, LB\_ADDSTRING, 0, (LPARAM)todo\_data[curDate][i].c\_str());

}

if (timerTime != 0) {

HWND hwndText = GetDlgItem(hWnd, IDC\_TEXT);

timer\_msg.push\_back(todo\_data[curDate][todo\_data[curDate].size() - 1]);

SetTimer(hWnd, tId, timerTime \* 1000, (TIMERPROC)NULL);

tId += 1;

}

break;

}

case IDM\_ABOUT:

DialogBox(hInst, MAKEINTRESOURCE(IDD\_ABOUTBOX), hWnd, About);

break;

case IDM\_SAVE:

{

ofn.lStructSize = sizeof(ofn);

ofn.hwndOwner = hWnd;

ofn.lpstrFile = szFile;

ofn.lpstrFile[0] = '\0';

ofn.nMaxFile = sizeof(szFile);

ofn.lpstrFilter = L"All\0\*.\*\0Text\0\*.TXT\0";

ofn.nFilterIndex = 1;

ofn.lpstrFileTitle = NULL;

ofn.nMaxFileTitle = 0;

ofn.lpstrInitialDir = NULL;

ofn.Flags = 0;

bool r = GetSaveFileName(&ofn);

if (r) {

wofstream myfile;

myfile.open(szFile);

for (auto it = todo\_data.begin(); it != todo\_data.end(); it++) {

myfile << it->first.c\_str() << L"\t";

for (auto i = it->second.begin(); i < it->second.end(); i++)

{

myfile << (\*i).c\_str() << L" ";

}

myfile << L"\n";

}

myfile.close();

}

break;

}

case IDM\_REPLACE:

{

HWND hdlg = NULL;

fr.hwndOwner = hWnd;

fr.lpstrFindWhat = szFindWhat;

fr.lpstrReplaceWith = szReplaceWith;

fr.wFindWhatLen = 100;

fr.wReplaceWithLen = 100;

fr.Flags = FR\_HIDEUPDOWN | FR\_HIDEMATCHCASE | FR\_HIDEWHOLEWORD | FR\_NOMATCHCASE | FR\_NOUPDOWN | FR\_NOWHOLEWORD;

hdlg = ReplaceText(&fr);

if (hdlg == NULL) print\_log("%i", CommDlgExtendedError());

break;

}

case IDM\_EXIT:

DestroyWindow(hWnd);

break;

default:

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

}

}

break;

case WM\_NOTIFY:

{

LPNMHDR lpnmhdr = (LPNMHDR)lParam;

switch (lpnmhdr->code)

{

case MCN\_SELECT: {

HWND hwndList = GetDlgItem(hWnd, IDC\_LIST);

LPNMSELCHANGE lpNMSelChange = (LPNMSELCHANGE)lParam;

wstring date = to\_wstring(lpNMSelChange->stSelStart.wYear) + to\_wstring(lpNMSelChange->stSelStart.wMonth) + to\_wstring(lpNMSelChange->stSelStart.wDay);

if (todo\_data.count(date) < 1) {

todo\_data[date] = vector<wstring>();

}

curItem = 0;

SendMessage(hwndList, LB\_RESETCONTENT, 0, 0);

for (int i = 0; i < todo\_data[date].size(); i++) {

SendMessage(hwndList, LB\_ADDSTRING, 0, (LPARAM)todo\_data[date][i].c\_str());

}

curDate = date;

return TRUE;

}

case UDN\_DELTAPOS:

{

LPNMUPDOWN lpnmud = (LPNMUPDOWN)lParam;

timerTime -= lpnmud->iDelta;

if (timerTime < 0) timerTime = 0;

HWND hwndText = GetDlgItem(hWnd, IDC\_TEXT);

wchar\_t str[1000];

swprintf(str, L"%d", timerTime);

SetWindowText(hwndText, str);

break;

}

default:

break;

}

}

break;

case WM\_SIZING:

{

RECT\* rect = (RECT\*)lParam;

RECT win\_rect;

HWND hwndList = GetDlgItem(hWnd, IDC\_LIST);

GetWindowRect(hwndList, &win\_rect);

SetWindowPos(hwndList, NULL, 270, 10, 350, rect->bottom - rect->top - 80, NULL);

return TRUE;

}

case WM\_MOUSEWHEEL:

{

HWND hwndList = GetDlgItem(hWnd, IDC\_LIST);

int delta = GET\_WHEEL\_DELTA\_WPARAM(wParam) / WHEEL\_DELTA;

curItem -= delta;

if (curItem < 0) curItem = 0;

int count = ListBox\_GetCount(hwndList);

if (curItem > count) curItem = count-1;

SendMessage(hwndList, LB\_SETCURSEL, curItem, NULL);

return 0;

}

case WM\_SYSCHAR:

{

wchar\_t c = wParam;

if (c == L's' || c == L'S') {

DialogBox(hInst, MAKEINTRESOURCE(IDD\_DIALOG1), hWnd, Save);

}

break;

}

case WM\_SYSKEYUP:

{

wchar\_t c = wParam;

if (c == L'f' || c == L'F') {

DialogBox(hInst, MAKEINTRESOURCE(IDD\_DIALOG1), hWnd, Save);

}

break;

}

case WM\_LBUTTONDBLCLK:

{

HWND hwndList = GetDlgItem(hWnd, IDC\_LIST);

ListBox\_DeleteString(hwndList, curItem);

todo\_data[curDate].erase(todo\_data[curDate].begin() + curItem);

curItem -= 1;

if (curItem < 0) curItem = 0;

break;

}

case WM\_LBUTTONDOWN:

if (bSelecting)

ReleaseCapture();

bSelecting = TRUE;

rcSelection.left = (short)LOWORD(lParam);

rcSelection.top = (short)HIWORD(lParam);

rcSelection.right = (short)LOWORD(lParam);

rcSelection.bottom = (short)HIWORD(lParam);

DrawSelectionRect(hWnd, &rcSelection);

SetCapture(hWnd);

return 0;

case WM\_LBUTTONUP:

if (bSelecting)

{

DrawSelectionRect(hWnd, &rcSelection);

bSelecting = FALSE;

ReleaseCapture();

if (rcSelection.left > rcSelection.right)

{

int nTemp = rcSelection.left;

rcSelection.left = rcSelection.right;

rcSelection.right = nTemp;

}

if (rcSelection.top > rcSelection.bottom)

{

int nTemp = rcSelection.top;

rcSelection.top = rcSelection.bottom;

rcSelection.bottom = nTemp;

}

InvalidateRect(hWnd, NULL, TRUE);

}

return 0;

case WM\_MOUSEMOVE:

if (bSelecting)

{

RECT rcClient;

DrawSelectionRect(hWnd, &rcSelection);

rcSelection.right = (short)LOWORD(lParam);

rcSelection.bottom = (short)HIWORD(lParam);

GetClientRect(hWnd, &rcClient);

if (rcSelection.right < 0)

rcSelection.right = 0;

if (rcSelection.right > rcClient.right)

rcSelection.right = rcClient.right;

if (rcSelection.bottom < 0)

rcSelection.bottom = 0;

if (rcSelection.bottom > rcClient.bottom)

rcSelection.bottom = rcClient.bottom;

DrawSelectionRect(hWnd, &rcSelection);

}

return 0;

case WM\_TIMER:

{

int id = (int)wParam;

KillTimer(hWnd, id);

wstring str = timer\_msg.at(id);

const wchar\_t\* wcs = str.c\_str();

print\_log("%w", str);

MessageBox(hWnd, (LPCWSTR)wcs, L"Сообщение", MB\_OK);

break;

}

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

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

}

return 0;

}

INT\_PTR CALLBACK About(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)

{

UNREFERENCED\_PARAMETER(lParam);

switch (message)

{

case WM\_INITDIALOG:

return (INT\_PTR)TRUE;

case WM\_COMMAND:

if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)

{

EndDialog(hDlg, LOWORD(wParam));

return (INT\_PTR)TRUE;

}

break;

}

return (INT\_PTR)FALSE;

}

INT\_PTR CALLBACK Save(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)

{

UNREFERENCED\_PARAMETER(lParam);

switch (message)

{

case WM\_INITDIALOG:

return (INT\_PTR)TRUE;

case WM\_COMMAND:

if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)

{

EndDialog(hDlg, LOWORD(wParam));

return (INT\_PTR)TRUE;

}

break;

}

return (INT\_PTR)FALSE;

}

**Вывод**

В ходе лабораторной работы получение практические навыки создания оконных приложений Windows на языке C++ с применением Win32 API.