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 № 2

на тему «Расширенное использование оконного интерфейса Win 32 и GDI. Формирование сложных изображений, создание и использование элементов управления, обработка различных сообщений, механизм перехвата сообщений»

Выполнил:

студент гр. 153503

Филипеня А.Д.

Проверил:

Гриценко Н.Ю.

Минск 2023

СОДЕРЖАНИЕ

[1 Постановка задачи 3](#_Toc146631498)

[2 Краткие теоретические сведения 4](#_Toc146631499)

[3 Результаты выполнения лабораторной работы 6](#_Toc146631500)

[Выводы 8](#_Toc146631501)

[Список использованных источников 9](#_Toc146631502)

[Приложение А (обязательное) Листинг кода 10](#_Toc146631503)

## 1 ПОСТАНОВКА ЗАДАЧИ

Целью выполнения лабораторной работы является создание оконного приложения на Win32 API с использованием GDI, созданием и использованием элементов управления, обработке различных сообщений, механизм перехвата сообщений (winhook). Для этого мы дополним наш имеющийся текстовый редактор, добавив кнопку для изменения цвета текста, кнопку для изменения цвета фона, кнопку для изменения шрифта, поля для ввода цвета.

## 2 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Win32 API (Windows API) представляет собой набор функций и интерфейсов, предоставляемых операционной системой Windows для разработки приложений. Этот мощный набор инструментов обеспечивает доступ к различным функциональным возможностям Windows, включая создание и управление окнами, обработку сообщений, работу с файлами и реестром, а также многие другие операции. Win32 API играет ключевую роль в разработке приложений для Windows и обеспечивает высокую степень контроля над поведением приложений.

GDI (Graphics Device Interface) — это часть Win32 API, отвечающая за графический вывод и взаимодействие с графическим оборудованием. GDI предоставляет функции для рисования графических объектов, создания изображений, управления цветами и шрифтами, а также реализации различных эффектов. Этот компонент позволяет разработчикам создавать интерфейсы пользовательских приложений, включая окна, кнопки, текст и другие графические элементы. GDI обеспечивает визуальное качество и интерактивность приложений, делая их более привлекательными и функциональными для пользователей. [1]

Совместное использование Win32 API и GDI позволяет разработчикам создавать мощные и интерактивные графические приложения под операционной системой Windows, обеспечивая широкие возможности по работе с графикой, окнами и элементами управления.

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

– Win32 API (Application Programming Interface): это набор функций и структур, предоставляемых операционной системой Windows для разработки приложений под Windows. Для создания приложения было использовано множество функций Win32 API для создания окна, обработки сообщений, управления таймерами и других задач. [1]

– Процедура обработки сообщений (SoftwareMainProcedure): это ключевой элемент приложений Windows. SoftwareMainProcedure — это функция, которая обрабатывает сообщения, отправляемые операционной системой приложению. В данном приложении SoftwareMainProcedure была реализована для обработки различных сообщений, таких как создание окна, команды меню, перерисовка и таймер.

– Меню: создано пользовательское меню с использованием функций CreateMenu и AppendMenu. Это меню позволяет пользователю сохранять или загружать файл, а также закрывать программу.

– Выбор параметров: пользователь имеет возможность настраивать различные параметры текста, такие как цвет, шрифт.

– Управление окнами: приложение управляет окном, его размерами, заголовком и стилями с использованием функций Win32 API, таких как CreateWindow и ShowWindow.

– Обработка событий: код обрабатывает различные события, такие как нажатия кнопок, изменение текста в поле ввода и закрытие окна. Это обеспечивает реакцию приложения на действия пользователя.

## 3 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

В ходе выполнения лабораторной работы был усовершенствован текстовый редактор с возможностью сохранения и загрузки файлов. При запуске программы пользователь видит базовый интерфейс текстового редактора с полем для ввода текста и кнопкой для его полного удаления, полями ввода цвета и кнопками для изменения фона поля ввода, цвета текста шрифта. Результат работы программы показан на рисунке 3.1.



Рисунок 3.1 – Результат работы программы

Для наглядного примера на рисунке 3.2 будет предоставлен результат работы программы с измененным цветом текста, цветом фона и шрифтом.

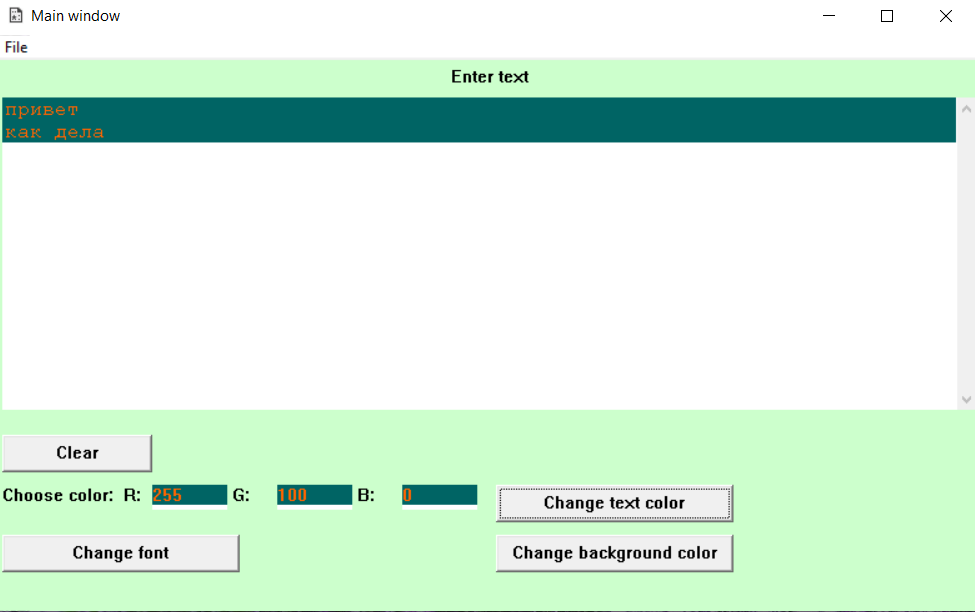


Рисунок 3.2 – Результат работы программы с измененными параметрами

## ВЫВОДЫ

В ходе выполнения данной лабораторной работы был создан простой графический анимационный пример с использованием Win32 API и GDI. Эта работа позволила ознакомиться с основными концепциями программирования под Windows, включая создание окна, обработку сообщений, работу с графической библиотекой GDI.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Щупак Ю. Win32 API. Разработка приложений для Windows. – СПб: Питер, 2008. – 592 с.: ип.

[2] [Электронный ресурс]. – Режим доступа: [https://learn.microsoft.com/ru-ru/windows/win32/api – Дата доступа 20.09.2023](https://learn.microsoft.com/ru-ru/windows/win32/api%20–%20Дата%20доступа%2020.09.2023)

[3] [Электронный ресурс]. – Режим доступа: [https://stackoverflow.com/questions/17187265/how-to-group-radio-box-buttons-using-win32-api – Дата доступа 23.09.2023](https://stackoverflow.com/questions/17187265/how-to-group-radio-box-buttons-using-win32-api%20–%20Дата%20доступа%2023.09.2023)

[4] [Электронный ресурс]. – Режим доступа: <https://prog-cpp.ru/winelements/> – Дата доступа 23.09.2023

[5] [Электронный ресурс]. – Режим доступа: [https://learn.microsoft.com/en-us/windows/win32/gdi/windows-gdi – Дата доступа 24.09.2023](https://learn.microsoft.com/en-us/windows/win32/gdi/windows-gdi%20–%20Дата%20доступа%2024.09.2023)

[6] [Электронный ресурс]. – Режим доступа: <https://learn.microsoft.com/en-us/windows/win32/controls/button-messages> – Дата доступа 24.09.2023

## ПРИЛОЖЕНИЕ А

## (обязательное)

## Листинг кода

**SoftwareMain.cpp**

#include <Windows.h>

#include <string>

#include "resource.h"

#include "SoftwareDefinitions.h"

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

HBRUSH backgroundColor = CreateSolidBrush(RGB(204, 255, 204));

font = CreateFontA(70, 30, 0, 0, FW\_MEDIUM, FALSE, FALSE, FALSE, DEFAULT\_CHARSET, OUT\_OUTLINE\_PRECIS, CLIP\_DEFAULT\_PRECIS, ANTIALIASED\_QUALITY, FF\_DECORATIVE, "MyFont");

WNDCLASS SoftwareMainClass = NewWindowClass(backgroundColor, LoadCursor(NULL, IDC\_ARROW),

hInst, LoadIcon(hInst, MAKEINTRESOURCE(IDI\_ICON1)), L"MainWndClass", SoftwareMainProcedure);

CreateSolidBrush(RGB(0, 0, 255));

if (!RegisterClassW(&SoftwareMainClass)) {

return -1;

}

MSG SoftwareMainMessage = { 0 };

CreateWindow(L"MainWndClass", L"Main window", WS\_OVERLAPPEDWINDOW | WS\_VISIBLE, 100, 100, 800, 500, NULL, NULL, NULL, NULL);

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

TranslateMessage(&SoftwareMainMessage);

DispatchMessage(&SoftwareMainMessage);

}

return 0;

}

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

WNDCLASS WindowClass = { 0 };

WindowClass.hbrBackground = BdColor;

WindowClass.hCursor = Cursor;

WindowClass.hInstance = hInst;

WindowClass.hIcon = Icon;

WindowClass.lpszClassName = Name;

WindowClass.lpfnWndProc = Procedure;

return WindowClass;

}

//CALlLBACK - спецификатор для функции обратного вызова, т.е. функции, которую вызывает сама операционная система

//hWnd - дескриптор окна, которому адресовано сообщение

//msg - код сообщения

//wp, lp - параметры сообщения

//WndProc - для главного окна приложения

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

switch (msg)

{

case WM\_COMMAND:

switch (wp)

{

case OnButtonClicked:

SetWindowTextA(hEditControl, "");

textColor = RGB(0, 0, 0);

backgroundColor = RGB(255, 255, 255);

break;

case OnChangeTextColor:

textColor = RGB(GetDlgItemInt(hWnd, indexColorR, NULL, false),

GetDlgItemInt(hWnd, indexColorG, NULL, false),

GetDlgItemInt(hWnd, indexColorB, NULL, false));

RedrawWindow(hWnd, NULL, NULL, RDW\_UPDATENOW | RDW\_INVALIDATE);

break;

case OnChangeBackgroundColor:

backgroundColor = RGB(GetDlgItemInt(hWnd, indexColorR, NULL, false),

GetDlgItemInt(hWnd, indexColorG, NULL, false),

GetDlgItemInt(hWnd, indexColorB, NULL, false));

RedrawWindow(hWnd, NULL, NULL, RDW\_UPDATENOW | RDW\_INVALIDATE);

break;

case OnChangeFont:

SendMessage(hEditControl,

WM\_SETFONT,

(WPARAM)hFont,

MAKELPARAM(TRUE, 0)

);

break;

case OnSaveFile:

if (GetSaveFileNameA(&file)) {

SaveData(path);

};

break;

case OnLoadFile:

if (GetOpenFileNameA(&file)) {

LoadData(path);

};

break;

case OnExitClicked:

PostQuitMessage(0);

break;

default:

break;

}

break;

case WM\_CTLCOLORSTATIC:

{

HDC hdcStatic = (HDC)wp;

SetBkColor(hdcStatic, RGB(204, 255, 204));

return (INT\_PTR)CreateSolidBrush(RGB(204, 255, 204));

}

case WM\_CTLCOLOREDIT:

{

HDC hdc = (HDC)wp;

SetTextColor(hdc, textColor);

SetBkColor(hdc, backgroundColor);

return 0;

}

break;

case WM\_SETFONT:

SendMessage(hEditControl, WM\_SETFONT, (WPARAM)font, MAKELPARAM(TRUE, 0));

case WM\_CREATE:

AddMenu(hWnd);

AddWidgets(hWnd);

SetOpenFileParams(hWnd);

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

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

}

}

void AddMenu(HWND hWnd) {

HMENU RootMenu = CreateMenu();

HMENU SubMenu = CreateMenu();

AppendMenu(SubMenu, MF\_STRING, OnSaveFile, L"Save");

AppendMenu(SubMenu, MF\_STRING, OnLoadFile, L"Load");

//разделитель меню,используем флаг separator

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

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

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

SetMenu(hWnd, RootMenu);

}

void AddWidgets(HWND hWnd) {

hStaticControl = CreateWindowA("static", "Enter text", WS\_VISIBLE | WS\_CHILD| ES\_CENTER, 5, 5, 780, 20, hWnd, NULL, NULL, NULL);

hEditControl = CreateWindowA("edit", "", ES\_MULTILINE | WS\_VSCROLL | WS\_VISIBLE | WS\_CHILD, 5, 30, 780, 250, hWnd, NULL, NULL, NULL);

CreateWindowA("button", "Clear", WS\_VISIBLE | WS\_CHILD, 5, 300, 120, 30, hWnd, (HMENU)OnButtonClicked, NULL, NULL);

//Цвета

CreateWindowA("static", "Choose color: R:", WS\_VISIBLE | WS\_CHILD, 5, 340, 120, 20, hWnd, NULL, NULL, NULL);

CreateWindowA("edit", NULL, ES\_MULTILINE | WS\_VISIBLE | WS\_CHILD, 125, 340, 60, 20, hWnd, (HMENU)indexColorR, NULL, NULL);

CreateWindowA("static", " G:", WS\_VISIBLE | WS\_CHILD, 185, 340, 40, 20, hWnd, NULL, NULL, NULL);

CreateWindowA("edit", NULL, ES\_MULTILINE | WS\_VISIBLE | WS\_CHILD, 225, 340, 60, 20, hWnd, (HMENU)indexColorG, NULL, NULL);

CreateWindowA("static", " B:", WS\_VISIBLE | WS\_CHILD, 285, 340, 40, 20, hWnd, NULL, NULL, NULL);

CreateWindowA("edit", NULL, ES\_MULTILINE | WS\_VISIBLE | WS\_CHILD, 325, 340, 60, 20, hWnd, (HMENU)indexColorB, NULL, NULL);

CreateWindowA("button", "Change text color", WS\_VISIBLE | WS\_CHILD, 400, 340, 190, 30, hWnd, (HMENU)OnChangeTextColor, NULL, NULL);

CreateWindowA("button", "Change background color", WS\_VISIBLE | WS\_CHILD, 400, 380, 190, 30, hWnd, (HMENU)OnChangeBackgroundColor, NULL, NULL);

CreateWindowA("button", "Change font", WS\_VISIBLE | WS\_CHILD, 5, 380, 190, 30, hWnd, (HMENU)OnChangeFont, NULL, NULL);

}

void SaveData(LPCSTR path) {

HANDLE file = CreateFileA(path, GENERIC\_WRITE, 0, NULL, CREATE\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL);

int length = GetWindowTextLength(hEditControl)+1;

char\* data = new char[length];

length = GetWindowTextA(hEditControl, data, length);

DWORD bytesIterated; //кол-во считанных символов

WriteFile(file, data, length, &bytesIterated, NULL);

CloseHandle(file);

delete[] data;

}

void LoadData(LPCSTR path) {

HANDLE file = CreateFileA(path, GENERIC\_READ, 0, NULL, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, NULL);

DWORD bytesIterated;

ReadFile(file, Buffer, TextBufferSize, &bytesIterated, NULL);

SetWindowTextA(hEditControl, Buffer);

CloseHandle(file);

}

void SetOpenFileParams(HWND hWnd) {

ZeroMemory(&file, sizeof(file));

file.lStructSize = sizeof(file);

file.hwndOwner = hWnd;

file.lpstrFile = path;

file.nMaxFile = sizeof(path);

file.lpstrFilter = ".txt";

file.lpstrFileTitle = NULL;

file.nMaxFileTitle = 0;

file.lpstrInitialDir = "D:\\5 sem\\SystemProgramming\\Lab1\\example.txt";

file.Flags = OFN\_PATHMUSTEXIST | OFN\_FILEMUSTEXIST;

}

**SoftwareMain.h**

#pragma once

#define OnMenuClicked 1

#define OnExitClicked 2

#define OnButtonClicked 3

#define OnReadField 4

#define OnSaveFile 5

#define OnLoadFile 6

#define OnNewWindow 7

#define indexColorR 8

#define indexColorG 9

#define indexColorB 10

#define OnChangeTextColor 11

#define OnChangeBackgroundColor 12

#define OnChangeFont 13

#define TextBufferSize 256

char Buffer[TextBufferSize];

int countOfReadChars;

HWND hEditControl;

HWND hStaticControl;

COLORREF textColor = RGB(0,0,0);

COLORREF backgroundColor = RGB(255,255,255);

PAINTSTRUCT ps;

HFONT font;

char path[200];

OPENFILENAMEA file;

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

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

HFONT hFont = CreateFont(18, 0, 0, 0, FW\_DONTCARE, FALSE, FALSE, FALSE, DEFAULT\_CHARSET, OUT\_OUTLINE\_PRECIS,

CLIP\_DEFAULT\_PRECIS, ANTIALIASED\_QUALITY, VARIABLE\_PITCH, TEXT("Courier New"));

void AddMenu(HWND hWnd);

void AddWidgets(HWND hWnd);

void SaveData(LPCSTR path);

void LoadData(LPCSTR path);

void SetOpenFileParams(HWND hWnd);