Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Операционные системы и системное программирование

Отчёт

к лабораторной работе

на тему

Расширенное использование оконного интерфейса Win 32 и GDI. Формирование сложных изображений, создание и использование элементов управления, обработка различных сообщений, механизм перехвата сообщений (winhook)

Студент: гр.153504

Пригожий К. А.

Проверил: Гриценко Н.Ю.

Минск 2023

**СОДЕРЖАНИЕ**

[1 Цель работы 3](#_Toc146635545)

[2 Теоретические сведения 4](#_Toc146635546)

[3 Результат выполнения программы 5](#_Toc146635547)

[Список использованных источников 8](#_Toc146635548)

[Приложение А 9](#_Toc146635549)

# 1 ЦЕЛЬ РАБОТЫ

Разработать текстовый редактор с поддержкой изменения шрифта, цвета, размера текста и изменения цвета заднего фона поля ввода.

# 2 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Интерфейс графических устройств *Microsoft* *Windows* (*GDI*) предоставляет приложениям возможность манипулировать графическими итекстовыми элементами на разнообразных устройствах вывода, включая видеодисплеи и принтеры. Однако в контексте операционной системы *Windows*, взаимодействие приложений с графическим оборудованием осуществляется не напрямую. Вместо этого, *GDI* выступает как посредник и взаимодействует с драйверами устройств от имени приложений.

Одной из ключевых аспектов работы с графикой является цветовая палитра. Эта палитра представляет собой набор цветовых значений, которые определяют цвета, используемые для отображения и рисования на конкретном устройстве вывода. Цветовые палитры имеют особое значение для устройств, способных воспроизводить множество цветов, но при этом ограниченных в числе цветов, которые могут быть одновременно отображены. Для таких устройств система поддерживает системную палитру, которая отслеживает текущие цвета устройства и управляет ими. Приложения не имеют прямого доступа к системной палитре; вместо этого, каждый контекст устройства ассоциируется с палитрой по умолчанию.

Для обеспечения более гибкого форматирования текста, приложения могут использовать структуру *CHARFORMAT*. *CHARFORMAT* позволяет определять различные параметры текста, включая характеристики шрифта, цвет текста и другие атрибуты. Это предоставляет приложениям возможность динамически изменять шрифты и их размеры для текстовых элементов, что позволяет конечным пользователям настраивать визуальное оформление текста внутри приложения.

Для того чтобы пользователи могли выбирать цвета и настраивать параметры шрифта и его размер в приложении, можно использовать функцию *ChooseColor* и манипулировать структурой *CHARFORMAT*. Если пользователь подтверждает свой выбор, нажав кнопку "*ОК*" в диалоговом окне *ChooseColor*, то выбранный цвет может быть применен к тексту. С использованием *CHARFORMAT* можно также устанавливать параметры шрифта, включая его размер. Эти возможности придает приложению большую гибкость и позволяет пользователям настраивать внешний вид текста в соответствии с их индивидуальными предпочтениями.

# 3 РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОГРАММЫ

В результате выполнения лабораторной работы получилась улучшенная версия предыдущей лабораторной: добавились элементы в меню для стилизации текста и фона во время редактирования (рисунок 1).

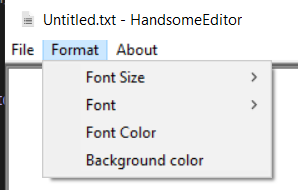


Рисунок 1 – Меню изменения формата

При наведении на кнопку «*Font Size*» всплывает окно выбора размера шрифта (рисунок 2).

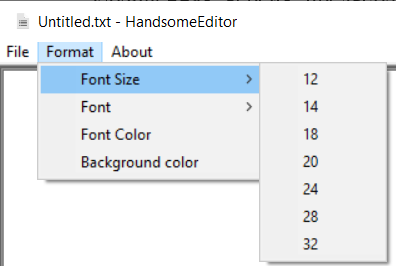


Рисунок 2 – Раздел выбора размера шрифта



Рисунок 3 – Поле ввода с выбранным размером шрифта равным 32

При наведении на кнопку «*Font*» всплывает окно с доступным шрифтами (рисунок 4).

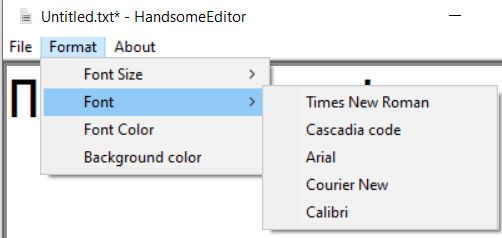


Рисунок 4 – Раздел выбора шрифта

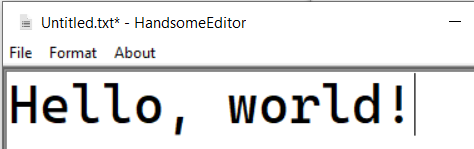


Рисунок 5 – Поле ввода после выбора шрифта *Cascadia* *Code*

При нажатии на кнопки «*Font Color*» и «*Background color*» всплывает окно выбора цвета (рисунок 6).

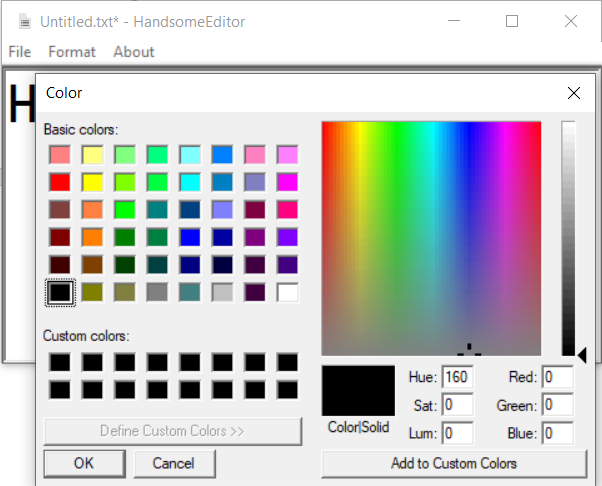


Рисунок 6 – Окно выбора цвета

Если пользователь выберет цвет и нажмет кнопку «*Ok*» цвет применится к соответсвующему элементу.



Рисунок 7 – Поле ввода после применения цветов

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Windows GDI [Электронный ресурс]. – Режим доступа: <https://learn.microsoft.com/ru-ru/windows/win32/gdi/windows-gdi>.

[2] Цветовые палитры (Windows GDI) [Электронный ресурс]. – Режим доступа: <https://learn.microsoft.com/ru-ru/windows/win32/gdi/color-palettes>.

[3] Системная палитра [Электронный ресурс]. – Режим доступа: <https://learn.microsoft.com/ru-ru/windows/win32/gdi/system-palette>.

[4] Использование перехватчиков [Электронный ресурс]. – Режим доступа: <https://learn.microsoft.com/ru-ru/windows/win32/winmsg/using-hooks>.

# ПРИЛОЖЕНИЕ А

**(обязательное)**

**Исходный код программы**

**Файл actions.cpp**

#include "actions.h"

void ProcessWMCommand(HWND hWnd, WPARAM wParam, LPARAM lParam)

{

switch (LOWORD(wParam)) {

case ID\_FONT\_TIMESNEWROMAN:

case ID\_FONT\_CASCADIA\_CODE:

case ID\_FONT\_ARIAL:

case ID\_FONT\_COURIERNEW:

case ID\_FONT\_CALIBRI: {

font = PickFont(LOWORD(wParam));

UpdateFont();

break;

}

case ID\_FONTSIZE\_12:

case ID\_FONTSIZE\_14:

case ID\_FONTSIZE\_18:

case ID\_FONTSIZE\_20:

case ID\_FONTSIZE\_22:

case ID\_FONTSIZE\_28:

case ID\_FONTSIZE\_32: {

fontSize = PickFontSize(LOWORD(wParam));

UpdateFont();

break;

}

case ID\_FORMAT\_FONTCOLOR: {

ChooseFontColor(hWnd);

break;

}

case ID\_FORMAT\_BACKGROUNDCOLOR: {

ChooseBackgroundColor(hWnd);

break;

}

case ID\_FILE\_SAVE:

SaveFile(hWnd);

break;

case ID\_FILE\_LOAD:

if (contentAlreadyChanged and MessageBox(hWnd, L"Do you want to save your changes?", L"Attention", MB\_YESNO) == IDYES) {

SaveFile(hWnd);

}

LoadFile(hWnd);

break;

case ID\_ABOUT:

MessageBoxA(hWnd, "Text editor created by Kiryl Pryhozhy. \nGroup №153504\n\n© Handsome Inc.", "About", MB\_OK);

break;

case ID\_EXIT:

PostQuitMessage(0);

break;

}

}

void UpdateFont()

{

charFormat.yHeight = fontSize \* 20;

lstrcpy(charFormat.szFaceName, std::wstring(font.begin(), font.end()).c\_str());

charFormat.crTextColor = fontColor;

SendMessage(hTextField, EM\_SETCHARFORMAT, SCF\_ALL, (LPARAM)&charFormat);

}

std::string PickFont(int cellId)

{

switch (cellId) {

case ID\_FONT\_TIMESNEWROMAN: return "Times New Roman";

case ID\_FONT\_CASCADIA\_CODE: return "Cascadia Code";

case ID\_FONT\_ARIAL: return "Arial";

case ID\_FONT\_COURIERNEW: return "Courier New";

case ID\_FONT\_CALIBRI: return "Calibri";

default: return "Georgia";

}

}

int PickFontSize(int cellId)

{

switch (cellId) {

case ID\_FONTSIZE\_12: return 12;

case ID\_FONTSIZE\_14: return 14;

case ID\_FONTSIZE\_18: return 18;

case ID\_FONTSIZE\_20: return 20;

case ID\_FONTSIZE\_22: return 22;

case ID\_FONTSIZE\_28: return 28;

case ID\_FONTSIZE\_32: return 32;

default: return 18;

}

}

void ChooseFontColor(HWND hWnd)

{

CHOOSECOLOR chooseColor = ConfigureChooseColor(hWnd, fontColor);

if (ChooseColor(&chooseColor)) {

charFormat.crTextColor = fontColor = chooseColor.rgbResult;

}

SendMessage(hTextField, EM\_SETCHARFORMAT, SCF\_ALL, (LPARAM)&charFormat);

}

void ChooseBackgroundColor(HWND hWnd)

{

CHOOSECOLOR chooseColor = ConfigureChooseColor(hWnd, backgroundColor);

if (ChooseColor(&chooseColor)) {

backgroundColor = chooseColor.rgbResult;

}

SendMessage(hTextField, EM\_SETBKGNDCOLOR, 0, backgroundColor);

}

CHOOSECOLOR ConfigureChooseColor(HWND hWnd, COLORREF initialColor)

{

static COLORREF customColors[16];

CHOOSECOLOR cc{ };

cc.lStructSize = sizeof(CHOOSECOLOR);

cc.hwndOwner = hWnd;

cc.lpCustColors = customColors;

cc.rgbResult = initialColor;

cc.Flags = CC\_FULLOPEN | CC\_RGBINIT;

return cc;

}