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

ОТЧЁТ

к лабораторной работе №1

на тему

Оконный интерфейс и оконные приложения Windows, обработка оконных сообщений, базовая графика GDI

Выполнил: студент группы 153501

Павлюткин Павел Сергеевич

Проверил: Сиротко Сергей Иванович

Минск 2023

СОДЕРЖАНИЕ

[1 Постановка задачи 3](#_Toc146728177)

[2 Краткие теоретические сведения 4](#_Toc146728178)

[2 Результаты выполнения лабораторной работы 7](#_Toc146728179)

[Выводы 9](#_Toc146728180)

[Список использованных источников 10](#_Toc146728181)

[Приложение А (обязательное) Листинг кода 11](#_Toc146728182)

## 1 ПОСТАНОВКА ЗАДАЧИ

Целью выполнения лабораторной работы является создание оконного приложения на Win32 API и использованию Win32 API с использованием GDI и формированию сложных изображений, обладающее минимальным функционалом, позволяющим отработать базовые навыки написания программы на Win32 API, использования виджетов и обработки оконных сообщений (как базовых, так и пользовательских).

## 2 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Интерфейс прикладного программирования Win32 API представляет собой набор функций и подпрограмм, предоставляющих программный доступ к возможностям операционной системы Windows. Этот API предоставляет средства для управления и взаимодействия с операционной системой, а также для создания приложений, которые работают в среде Windows.

Win32 API включает в себя более 3000 функций, которые позволяют реализовать различные сервисы операционной системы. Эти функции могут быть вызваны из приложений для выполнения разнообразных задач, таких как создание процессов (например, CreateProcess) и обработка оконных сообщений (например, GetMessage).

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

­– Win32 API (Application Programming Interface): Win32 API — это набор функций и подпрограмм, предоставляемых операционной системой Windows для взаимодействия с приложениями. Он предоставляет доступ к различным функциональным возможностям Windows, таким как создание окон, обработка сообщений, работа с файлами и др. В данной лабораторной работе Win32 API использован для создания графического интерфейса калькулятора и обработки событий.

– Обработка событий и сообщений: в Windows-приложениях взаимодействие с пользователем осуществляется через обработку событий и сообщений. В коде используются функции обработки сообщений, такие как SoftwareMainProcedure, которая отвечает за обработку событий, происходящих в приложении, таких как нажатия кнопок и меню.

– Создание пользовательского интерфейса: для создания графического интерфейса приложения были использованы стандартные элементы управления Windows, такие как окна, кнопки и текстовые поля. Эти элементы были созданы и настроены с использованием Win32 API функций.

– Математические операции: для выполнения математических операций, таких как сложение, вычитание, умножение и деление, а также расчет синуса, косинуса, тангенса и котангенса, использовались стандартные функции библиотеки cmath (например, sin, cos, tan) и операции с плавающей точкой для сохранения и обработки чисел.

– Работа с окнами и элементами управления: для создания оконного приложения была использована функция CreateWindow, которая создает окно с указанными характеристиками, такими как размер, положение и стиль.

Также использовались функции для работы с элементами управления, такие как кнопки и текстовые поля.

– 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.

– Обработка ошибок: для обработки ошибок, таких как деление на ноль или некорректный ввод, использовались функции вывода сообщений с помощью MessageBoxA.

– Интерфейс пользователя: графический интерфейс приложения разработан с учетом стандартных принципов пользовательского интерфейса, таких как расположение элементов и размещение текста.

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.

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.

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

– GDI (Graphics Device Interface): GDI — это подсистема Windows для рисования графики на экране и на печать. В данном приложении функции GDI, такие как Ellipse и Rectangle, использовались для рисования объектов на окне.

– Таймеры: для обновления позиции объекта и перерисовки окна с определенной частотой были использованы таймеры с помощью функций SetTimer и KillTimer. Это обеспечило анимацию в приложении.

– Меню: создано пользовательское меню с использованием функций CreateMenu и AppendMenu. Это меню позволяет пользователю выбирать тип фигуры, направление движения и цвет фигуры, что делает его важным элементом графического интерфейса пользователя (GUI).

– Выбор параметров: пользователь имеет возможность настраивать различные параметры анимации, такие как тип траектории, скорость, тип фигуры и цвет. Это демонстрирует концепцию интерактивности в приложении.

– 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.

## 2 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

В ходе выполнения лабораторной работы было разработано оконное приложение отрисовывающее круг либо квадрат (при нажатии на кнопку) и позволяющее пользователю выбирать соответствующий цвет для данной фигуры (Рисунок 1).

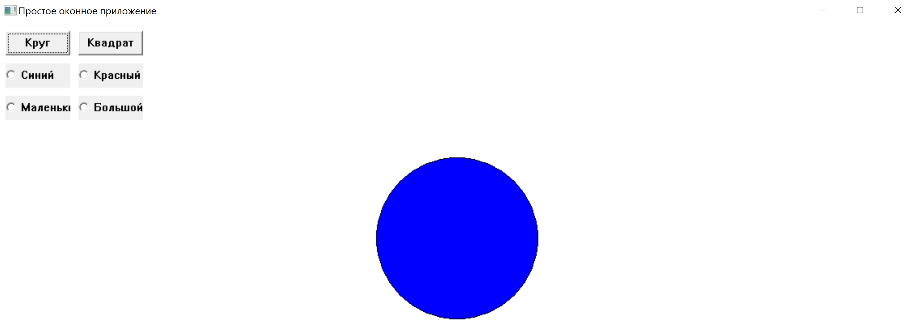


Рисунок 1 – Круг с параметрами (синий + большой)

Пользователь может изменять фигуру (Рисунок 2).

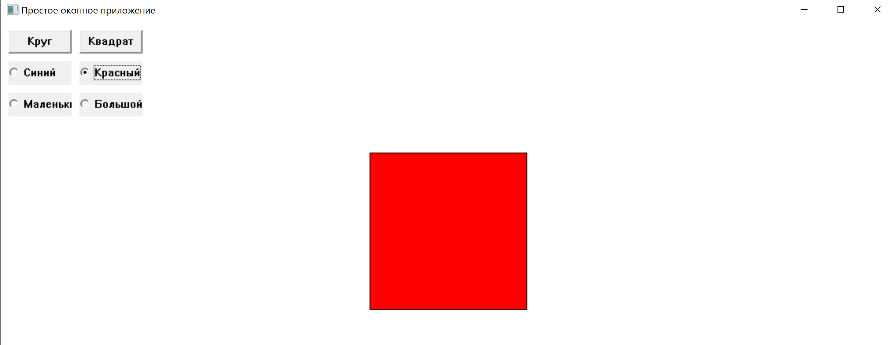


Рисунок 2 ─ Квадрат с параметрами (большой + красный)

При нажатии на клавишу изменения размера фигуры (маленький либо большой) – квадрат изменяет свой размер соответственно (Рисунок 3).

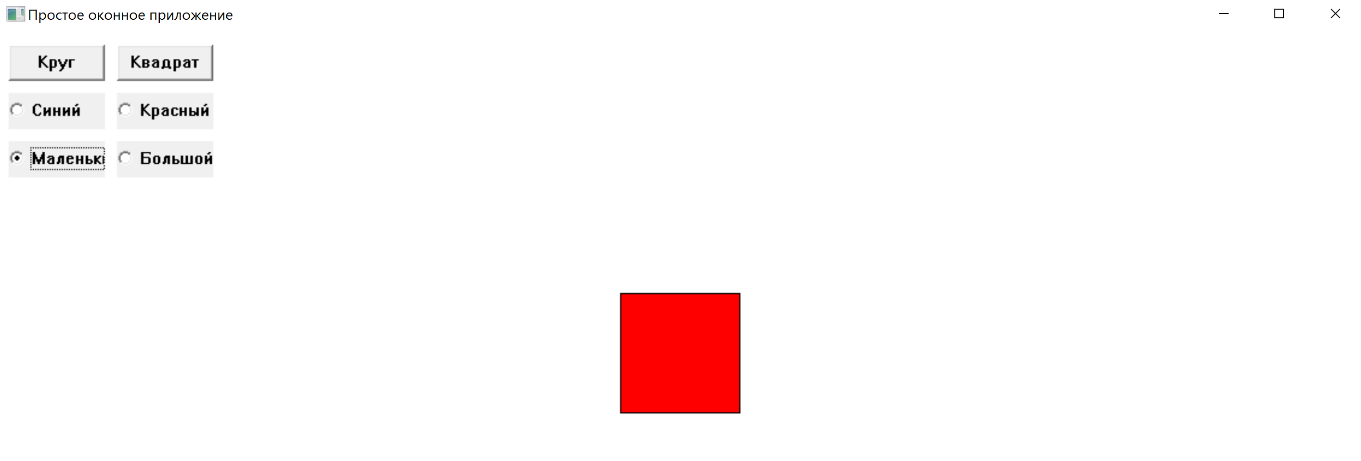


Рисунок 3 – Квадрат с параметрами (маленький + красный)

Круг с параметрами синий и маленький (Рисунок 4).

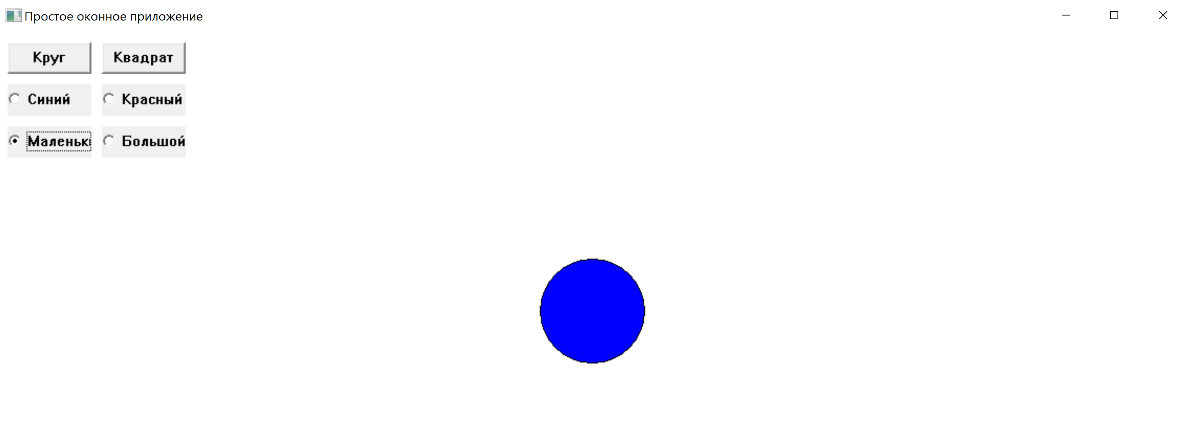


Рисунок 4 – Круг с параметрами (синий + маленький)

## ВЫВОДЫ

В ходе выполнения данной лабораторной работы было разработано оконное приложение, позволяющее пользователю рисовать фигуры при нажатии на соответствующую клавишу и позволяющее пользователю изменять цвет и размер фигуры.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

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

[2] Создание классических приложений для Windows с использованием API Win32 [Электронный ресурс]. ─ Режим доступа: https://learn.microsoft.com/ru-ru/windows/win32/api ─ Дата доступа 17.09.2023

## ПРИЛОЖЕНИЕ А

## (обязательное)

## Листинг кода

**Lab1.cpp**

#include <windows.h>

// Глобальные переменные

HINSTANCE hInst;

LPCTSTR szWindowClass = L"ПримерОкна";

LPCTSTR szTitle = L"Простое оконное приложение";

// Объявление функции обработки сообщений

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

// Тип для определения фигуры для рисования

enum ShapeType {

CIRCLE,

SQUARE

};

// Текущая выбранная фигура

ShapeType currentShape = CIRCLE;

// Текущий выбранный цвет и размер фигуры

COLORREF currentColor = RGB(0, 0, 255); // Синий цвет по умолчанию

int shapeSize = 100; // Размер фигур по умолчанию

// Текущая кисть для рисования

HBRUSH brush;

// Новые идентификаторы для кнопок

#define IDC\_BLUE 101

#define IDC\_RED 102

#define IDC\_SMALL 103

#define IDC\_LARGE 104

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow) {

UNREFERENCED\_PARAMETER(hPrevInstance);

UNREFERENCED\_PARAMETER(lpCmdLine);

hInst = hInstance;

// Регистрация класса окна

WNDCLASSEX wcex = { sizeof(WNDCLASSEX) };

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

wcex.lpfnWndProc = WndProc;

wcex.cbClsExtra = 0;

wcex.cbWndExtra = 0;

wcex.hInstance = hInstance;

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

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

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

wcex.lpszMenuName = NULL;

wcex.lpszClassName = szWindowClass;

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

if (!RegisterClassEx(&wcex)) {

MessageBox(NULL, L"Не удалось зарегистрировать класс окна.", L"Ошибка", MB\_ICONERROR);

return 1;

}

// Создание окна

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

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

if (!hWnd) {

MessageBox(NULL, L"Не удалось создать окно.", L"Ошибка", MB\_ICONERROR);

return 1;

}

// Создание кнопок для выбора фигуры

CreateWindow(L"BUTTON", L"Круг", WS\_CHILD | WS\_VISIBLE, 10, 10, 80, 30, hWnd, (HMENU)CIRCLE, hInstance, NULL);

CreateWindow(L"BUTTON", L"Квадрат", WS\_CHILD | WS\_VISIBLE, 100, 10, 80, 30, hWnd, (HMENU)SQUARE, hInstance, NULL);

// Создание кнопок для изменения цвета и размера фигуры

CreateWindow(L"BUTTON", L"Синий", WS\_CHILD | WS\_VISIBLE | BS\_AUTORADIOBUTTON, 10, 50, 80, 30, hWnd, (HMENU)IDC\_BLUE, hInstance, NULL);

CreateWindow(L"BUTTON", L"Красный", WS\_CHILD | WS\_VISIBLE | BS\_AUTORADIOBUTTON, 100, 50, 80, 30, hWnd, (HMENU)IDC\_RED, hInstance, NULL);

CreateWindow(L"BUTTON", L"Маленький", WS\_CHILD | WS\_VISIBLE | BS\_AUTORADIOBUTTON, 10, 90, 80, 30, hWnd, (HMENU)IDC\_SMALL, hInstance, NULL);

CreateWindow(L"BUTTON", L"Большой", WS\_CHILD | WS\_VISIBLE | BS\_AUTORADIOBUTTON, 100, 90, 80, 30, hWnd, (HMENU)IDC\_LARGE, hInstance, NULL);

// Инициализация кисти

brush = CreateSolidBrush(currentColor);

// Отображение окна

ShowWindow(hWnd, nCmdShow);

UpdateWindow(hWnd);

// Основной цикл сообщений

MSG msg;

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

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return (int)msg.wParam;

}

// Функция обработки сообщений

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

PAINTSTRUCT ps;

HDC hdc;

hdc = BeginPaint(hWnd, &ps);

RECT rect;

GetClientRect(hWnd, &rect);

int centerX = (rect.right - rect.left) / 2;

int centerY = (rect.bottom - rect.top) / 2;

switch (message) {

case WM\_PAINT:

if (currentShape == CIRCLE) {

SelectObject(hdc, brush); // Выбор текущей кисти

Ellipse(hdc, centerX - shapeSize, centerY - shapeSize, centerX + shapeSize, centerY + shapeSize);

}

else if (currentShape == SQUARE) {

SelectObject(hdc, brush); // Выбор текущей кисти

Rectangle(hdc, centerX - shapeSize, centerY - shapeSize, centerX + shapeSize, centerY + shapeSize);

}

EndPaint(hWnd, &ps);

break;

case WM\_DESTROY:

DeleteObject(brush); // Освобождение кисти

PostQuitMessage(0);

break;

case WM\_COMMAND:

switch (LOWORD(wParam)) {

case CIRCLE:

currentShape = CIRCLE;

InvalidateRect(hWnd, NULL, TRUE);

break;

case SQUARE:

currentShape = SQUARE;

InvalidateRect(hWnd, NULL, TRUE);

break;

case IDC\_BLUE:

currentColor = RGB(0, 0, 255); // Синий

DeleteObject(brush); // Освобождение предыдущей кисти

brush = CreateSolidBrush(currentColor); // Создание новой кисти

InvalidateRect(hWnd, NULL, TRUE);

break;

case IDC\_RED:

currentColor = RGB(255, 0, 0); // Красный

DeleteObject(brush); // Освобождение предыдущей кисти

brush = CreateSolidBrush(currentColor); // Создание новой кисти

InvalidateRect(hWnd, NULL, TRUE);

break;

case IDC\_SMALL:

shapeSize = 50; // Установка маленького размера

InvalidateRect(hWnd, NULL, TRUE);

break;

case IDC\_LARGE:

shapeSize = 100; // Установка большого размера

InvalidateRect(hWnd, NULL, TRUE);

break;

}

break;

default:

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

}

return 0;

}