МИНИСТЕРСТВО НАУКИ и высшего образования РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования «МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ»

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

филиал «РКТ» МАИ в г. Химки Московской области

**Специальность 09.02.03 «Программирование в компьютерных системах»**

**ОТЧЕТ ПО УЧЕБНОЙ ПРАКТИКЕ**

**ПМ.01 «Разработка программных модулей программного обеспечения для компьютерных систем»**

**Студент**

**группы МП31-20 / Жуков В.М.**

**Руководитель**

**практики от филиала / Жилина Т.А.**

**2023.**

**АТТЕСТАЦИОННЫЙ ЛИСТ ПО УЧЕБНОЙ ПРАКТИКЕ**

Студент Жуков Владислав Максимович , обучающийся на 3 курсе по специальности СПО 09.02.03 Программирование в компьютерных системах успешно прошел учебную практику в объеме 72 часов с «16»февраля2023г. по «01»марта 2023г.

В филиале «РКТ» МАИ в г. Химки Московской области

*Наименование предприятия/организации, юридический адрес*

**Виды и качество выполнения работ**

|  |  |  |
| --- | --- | --- |
| Виды работ, выполненных обучающимся во время практики | Объём работ  (час.) | Качество выполнения работ в соответствии с технологией и (или) требованиями организации, в которой походила практика |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Руководитель практики от филиала «РКТ» МАИ в г. Химки Московской области

Преподаватель Жилина Т.А

(должность, фамилия, имя, отчество)

|  |  |  |
| --- | --- | --- |
| Дата 01.03.2023 | *(подпись)* | Жилина Т.А.  *Расшифровка подписи* |

**ХАРАКТЕРИСТИКА**

Обучающийся Жуков Владислав Максимович группы МП31-20 по специальности СПО 09.02.03Программирование в компьютерных системах успешно прошел учебную практику в объеме

72  часов с «16»  февраля 2023г. по «01» марта 2023г.

в организации филиал «РКТ» МАИ в г. Химки Московской области

*Наименование предприятия /организации, юридический адрес*

За время работы проявил себя как ответственный/безответственный, коммуникабельный/замкнутый, исполнительный/неисполнительный, дисциплинированный/имеющий дисциплинарные замечания, доброжелательный/наглый сотрудник.

Обучающийся обладает общими компетенциями, включающими в себя способность:

ОК 1. Понимать сущность и социальную значимость своей будущей профессии, проявлять к ней устойчивый интерес.

ОК 2. Организовывать собственную деятельность, выбирать типовые методы и способы выполнения профессиональных задач, оценивать их эффективность и качество.

ОК 3. Принимать решения в стандартных и нестандартных ситуациях и нести за них ответственность.

ОК 4. Осуществлять поиск и использование информации, необходимой для эффективного выполнения профессиональных задач, профессионального и личностного развития.

ОК 5. Использовать информационно-коммуникационные технологии в профессиональной деятельности.

ОК 6. Работать в коллективе и в команде, эффективно общаться с коллегами, руководством, потребителями.

ОК 7. Брать на себя ответственность за работу членов команды (подчиненных), за результат выполнения заданий.

ОК 8. Самостоятельно определять задачи профессионального и личностного развития, заниматься самообразованием, осознанно планировать повышение квалификации.

ОК 9. Ориентироваться в условиях частой смены технологий в профессиональной деятельности.

В ходе выполнения всех видов работ обучающийся(аяся) показал(а) сформированность следующих профессиональных компетенций:

ПК 1.1. Выполнять разработку спецификаций отдельных компонент.

ПК 1.2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК 1.3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК 1.4. Выполнять тестирование программных модулей.

ПК 1.5. Осуществлять оптимизацию программного кода модуля.

ПК 1.6. Разрабатывать компоненты проектной и технической документации с использованием графических языков спецификаций.

**К работе относился**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

**Цели и задачи практики** достигнуты/ достигнуты не в полном объеме.

**Оценка за практику**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

Руководитель практики от организации «РКТ» МАИ в г. Химки Московской области

Преподаватель Жилина Т.А.

(должность, фамилия, имя, отчество)

|  |  |  |
| --- | --- | --- |
| Дата 01.03.2023 | *(подпись)* | Жилина Т.А.  *Расшифровка подписи* |

**Программа учебной практики**

По модулю ПМ.01 «Разработка программных модулей программного обеспечения для компьютерных систем»

По специальности СПО 09.02.03 Программирование в компьютерных системах.

Дата проведения с 16.02.2023 по 01.03.2023

|  |  |  |
| --- | --- | --- |
| №п/п | Наименование вида работ | Количество дней практики |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Руководитель практики от филиала «РКТ» МАИ в г. Химки Московской области

Преподаватель Жилина Т.А

(должность, фамилия, имя, отчество)

|  |  |  |
| --- | --- | --- |
| Дата 01.03.2023г. | *(подпись)* | Жилина Т.А  *Расшифровка подписи* |

**ДНЕВНИК ПРОХОЖДЕНИЯ ПРАКТИКИ**

По специальности СПО 09.02.03Программирование в компьютерных системах.

|  |  |  |
| --- | --- | --- |
| Дата | Наименование выполняемых работ | Подпись руководителя предприятия/ОУ |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Параметры учета выполнения работ по заданию**

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование вида работ | Количество дней практики | Прилагаемый материал в отчете |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

### **1 Задание первое. Диалоговые окна, реализация команд для вызова MSPaint и Calculator, свойства CheckBox**

Разработать код программного модуля создающего диалоговую панель в которой при вводе в строку редактирования Paint и Calculator запускаются MSPaint и Windows Calculator. На данной панели должны присутствовать чек CheckBox: видимость и блокировка. В активном состоянии строка видна и доступна для редактирования, при снятии галочек строка исчезает или блокируется.

Создается главное окно, далее поверх него создается диалоговое окно. В диалоговом присутствуют строка ввода, текст получается функцией GetWindowText, кнопка открыть и два check бокса. При нажатии кнопки, посылается сообщение IDMYOK, если в строке ввода введено «paint» - открывается paint, функцией Winexec, а если «calculator» открывается калькулятор, той же функцией, в случае ввода чего-то другого высвечивается messagebox. Вместо кнопки можно нажать просто enter. При снятии галочки с check бокса «Lock» в строку ввода нельзя вводить текст, вернуть обратно – поставить галочку. При снятии галочки с check бокса «Vision» строка ввода скрывается, вернуть обратно – поставить галочку.

Ниже приводится код реализации диалогового окна, кнопки, строки ввода и check боксов.

IDD\_MYDIALOG DIALOGEX 0, 0, 309, 176

STYLE DS\_SETFONT | DS\_MODALFRAME | DS\_FIXEDSYS | WS\_POPUP | WS\_CAPTION | WS\_SYSMENU

CAPTION "Open"

FONT 8, "MS Shell Dlg", 400, 0, 0x1

BEGIN

DEFPUSHBUTTON "ОК",IDMYOK,129,76,50,14

EDITTEXT IDC\_EDIT,97,57,118,14,ES\_AUTOHSCROLL

CONTROL "Visible",IDC\_CHECK1,"Button",BS\_AUTOCHECKBOX | WS\_TABSTOP,37,136,39,10

CONTROL "Lock",IDC\_CHECK2,"Button",BS\_AUTOCHECKBOX | WS\_TABSTOP,203,134,39,10

END

INT\_PTR CALLBACK DialogProc(HWND hDlg, UINT uMsg, WPARAM wParam, LPARAM lParam)

{

switch (uMsg)

{

case WM\_INITDIALOG:

{

auto hwnd\_check\_1 = GetDlgItem(hDlg, IDC\_CHECK1);

SendMessage(hwnd\_check\_1, BM\_SETCHECK, BST\_CHECKED, 0);

auto hwnd\_check\_2 = GetDlgItem(hDlg, IDC\_CHECK2);

SendMessage(hwnd\_check\_2, BM\_SETCHECK, BST\_CHECKED, 0);

}

case WM\_COMMAND:

{

if (wParam == IDMYOK)

{

auto hwnd = GetDlgItem(hDlg, IDC\_EDIT);

int len = GetWindowTextLength(hwnd) + 1;

std::vector<wchar\_t> buf(len);

GetWindowText(hwnd, &buf[0], len);

std::wstring command = &buf[0];

if (command == L"paint")

{

WinExec("mspaint.exe", 1);

}

else if (command == L"calculator")

WinExec("calc.exe", 1);

else

{

MessageBox(hDlg, L"Введите paint или calculator", L"Ошибка", MB\_OK);

}

}

else if (wParam == IDC\_CHECK1) //visible

{

auto hwnd = GetDlgItem(hDlg, IDC\_CHECK1);

auto res = SendMessage(hwnd, BM\_GETCHECK, 0, 0);

if (res == BST\_CHECKED)

{

auto hwnd\_edit = GetDlgItem(hDlg, IDC\_EDIT);

ShowWindow(hwnd\_edit, SW\_SHOW);

}

else if (res == BST\_UNCHECKED)

{

auto hwnd\_edit = GetDlgItem(hDlg, IDC\_EDIT);

ShowWindow(hwnd\_edit, SW\_HIDE);

}

}

else if (wParam == IDC\_CHECK2)

{

auto hwnd = GetDlgItem(hDlg, IDC\_CHECK2);

auto res = SendMessage(hwnd, BM\_GETCHECK, 0, 0);

if (res == BST\_CHECKED)

{

SendDlgItemMessage(hDlg, IDC\_EDIT, EM\_SETREADONLY, FALSE, 0);

}

else if (res == BST\_UNCHECKED)

{

SendDlgItemMessage(hDlg, IDC\_EDIT, EM\_SETREADONLY, TRUE, 0);

}

}

else if (wParam == IDCANCEL)

{

EndDialog(hDlg, 1);

}

}

}

return 0;

}

На рисунке 1 изображен результат выполнения кода. На нем показаны главное окно, и диалоговое окно с строкой ввода, кнопка «ок» и check боксы «Lock» и «Vision».

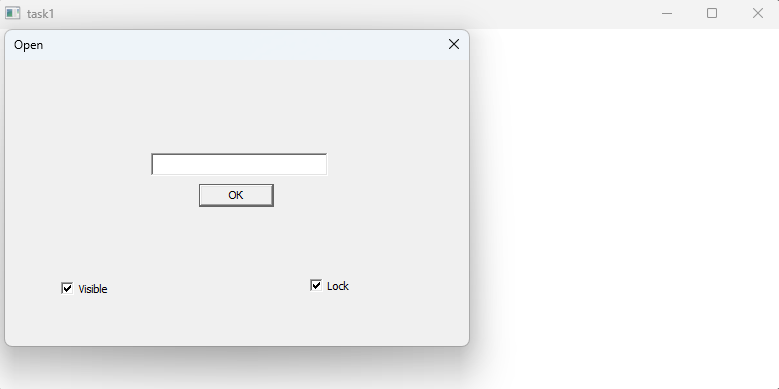
****

Рисунок 1 – Результат выполнения кода

На рисунке 2 представлен результат выполнения кода при отключенной checkbox(Видимость) – происходит скрытие textbox.

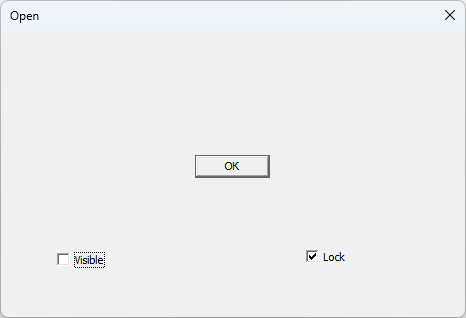


Рисунок 2 – Результат выполнения при отключенной видимости

На рисунке 3 представлен результат выполнения при выключенной checkbox (Блокировка) – происходит блокировка изменений textbox.

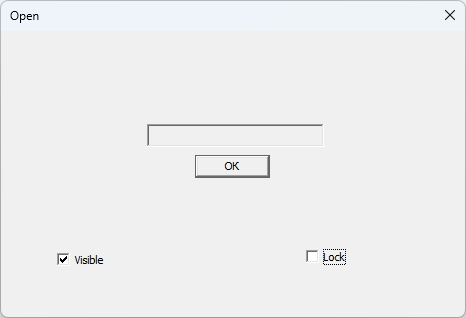


Рисунок 3 – Результат выполнения при выключенной блокировке

Данный вариант диалогового окна является верным, так как реализованы все необходимые методы и соблюдены условия задачи.

### **2 Задание второе. Рисование в окне курсором мыши, реализация смены цвета и толщины линии**

Разработать программный модуль, в котором предусмотрена возможность для рисования с помощью линий, установленного цвета и размера в соответствии с движением мыши. Условия: наличие возможности выбора минимум 3-х цветов (из списка), наличие возможности выбора толщины линий, наличие кнопки "выход".

Создаются два окна главное и дочернее. На одном можно рисовать, на втором выбирается цвет и толщина линий рисования. Выбор цвета и толщины реализован с помощью ListBox. Можно выбрать три цвета: красный, зеленый и желтый. Толщин линий тоже 3: 3px, 6px, 9 px. Из программы можно выйти благодаря реализованной кнопке Exit.

В приведенном ниже коде реализация метода смены толщины линии, цвета и выход из программы.

LRESULT CALLBACK windowprocessforwindow1(HWND handleforwindow, UINT msg, WPARAM wParam, LPARAM lParam)

{

static HPEN hPen = NULL;

static

BOOL fDraw = FALSE;

static POINT ptPrevious = { 0 };

HDC hdc;

switch (msg)

{

case WM\_PAINT:

{

PAINTSTRUCT ps;

HDC hdc = BeginPaint(handleforwindow, &ps);

SelectObject(hdc, GetStockObject(DC\_BRUSH));

FillRect(hdc, &ps.rcPaint, (HBRUSH)(COLOR\_WINDOW + 1));

EndPaint(handleforwindow, &ps);

break;

}

case WM\_LBUTTONDOWN:

fDraw = TRUE;

ptPrevious.x = LOWORD(lParam);

ptPrevious.y = HIWORD(lParam);

break;

case WM\_LBUTTONUP:

if (fDraw)

{

hdc = GetDC(handleforwindow);

MoveToEx(hdc, ptPrevious.x, ptPrevious.y, NULL);

LineTo(hdc, LOWORD(lParam), HIWORD(lParam));

ReleaseDC(handleforwindow, hdc);

fDraw = FALSE;

}

break;

case WM\_MOUSEMOVE:

if (fDraw)

{

hdc = GetDC(handleforwindow);

HPEN pen;

pen = CreatePen(PS\_SOLID, 5, RGB(255, 0, 0));

auto index = SendMessage(list\_box, LB\_GETCURSEL, 0, 0);

auto index\_width = SendMessage(width\_box, LB\_GETCURSEL, 0, 0);

auto data = SendMessage(width\_box, LB\_GETITEMDATA, index\_width, 0);

if (index == 0)

pen = CreatePen(PS\_SOLID, data, RGB(255, 0, 0));

else if (index == 1)

pen = CreatePen(PS\_SOLID, data, RGB(0, 128, 0));

else if (index == 2)

pen = CreatePen(PS\_SOLID, data, RGB(255, 255, 0));

SelectObject(hdc, pen);

MoveToEx(hdc, ptPrevious.x, ptPrevious.y, NULL);

LineTo

(

hdc,

ptPrevious.x = LOWORD(lParam),

ptPrevious.y = HIWORD(lParam)

);

ReleaseDC(handleforwindow, hdc);

}

break;

case WM\_COMMAND:

{

if ((buttons)wParam == buttons::exit)

PostQuitMessage(0);

break;

}

case WM\_DESTROY:

{

PostQuitMessage(0);

break;

}

}

return DefWindowProc(handleforwindow, msg, wParam, lParam);

}

На рисунке 4 изображен результат выполнения кода. На нем показаны окна рисования с кнопкой выхода и окно с параметрами линий: цвет и толщина.

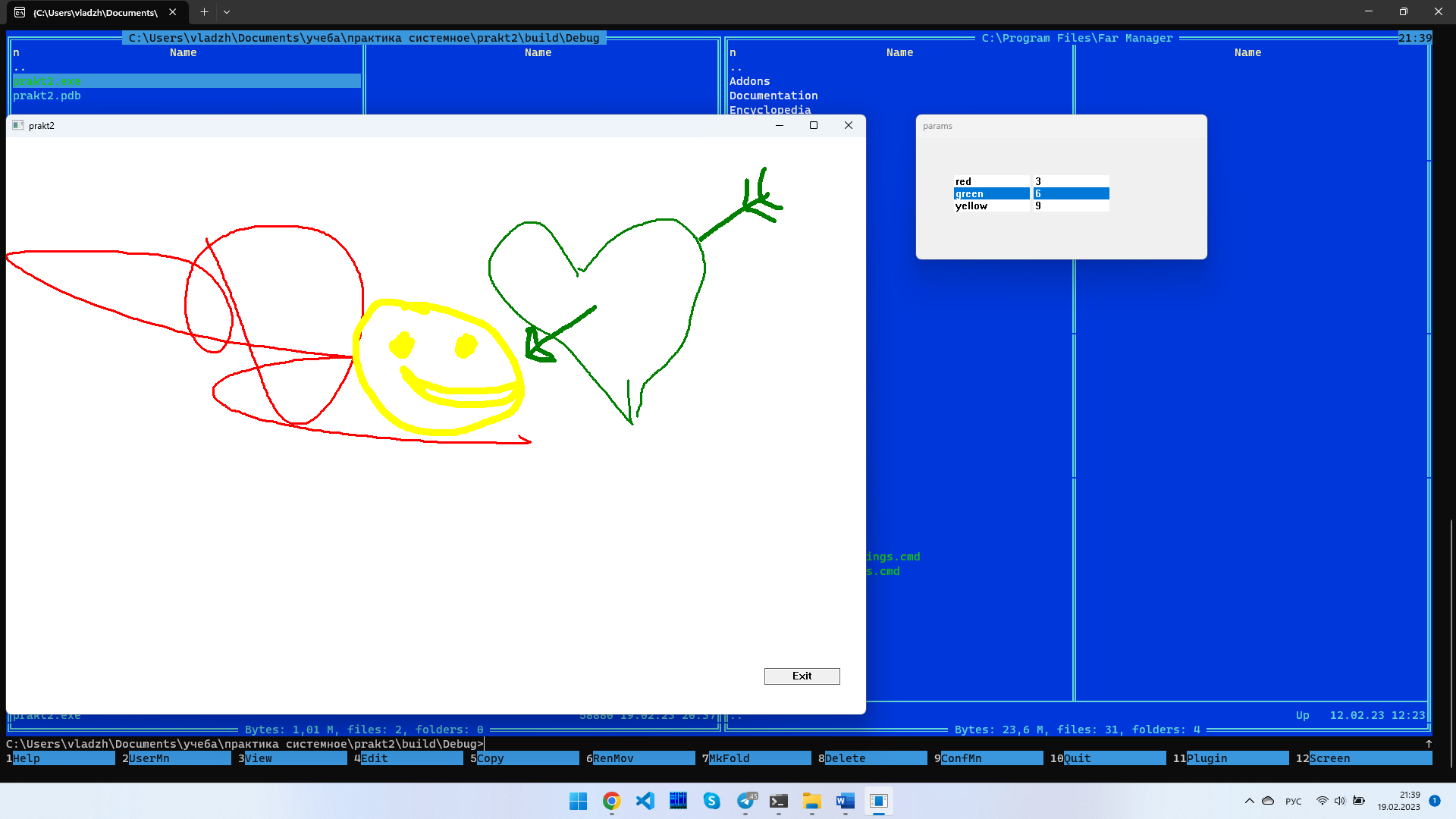
****

Рисунок 4 – Результат выполнения кода

Данный вариант диалогового окна является верным, так как реализованы все необходимые методы и соблюдены условия задачи.

### 

### **3 Задание третье. Вывод нажатой клавиши в дочернем окне, которое запускается на кнопку в главном окне**

Разработка программного модуля, создающего диалоговую панель, на которой должна быть кнопка запуска дочернего окна, в котором при нажатии на клавиатуру, показывается нажатый символ.

Создается главное окно с кнопкой открытия дочернего. При нажатии кнопки run открывается дочернее диалоговое окно. При нажатии клавиши на клавиатуре в диалоговом окне высвечивается текст значения этой клавиши.

В приведенном ниже коде показана реализация отображения дочернего окна, считывания клавиши и отображения ее в окне.

LRESULT CALLBACK windowprocessforwindow1(HWND handleforwindow, UINT msg, WPARAM wParam, LPARAM lParam)

{

switch (msg)

{

case WM\_COMMAND:

{

if ((buttons)wParam == buttons::run)

{

HWND params\_handle = CreateWindowEx(NULL,

name,

L"output",

WS\_OVERLAPPEDWINDOW,

1200,

150,

400,

200,

main\_handle,

NULL,

(HINSTANCE)GetModuleHandle(NULL),

NULL);

ShowWindow(params\_handle, SW\_NORMAL);

static\_cont = CreateWindow(L"STATIC", L"", WS\_VISIBLE | WS\_CHILD, CW\_USEDEFAULT, CW\_USEDEFAULT,

100, 25, params\_handle, NULL, NULL, NULL);

}

break;

}

case WM\_KEYDOWN:

{

std::wstring key;

GetKeyNameText(lParam, (LPWSTR)key.c\_str(), 100);

SetWindowText(static\_cont, (LPWSTR)key.c\_str());

break;

}

case WM\_DESTROY:

{

PostQuitMessage(0);

break;

}

}

return DefWindowProc(handleforwindow, msg, wParam, lParam);

}

На рисунке 5 показан результат выполнения кода. На нем изображены главное окно с кнопкой открытия дочернего диалогового окна.

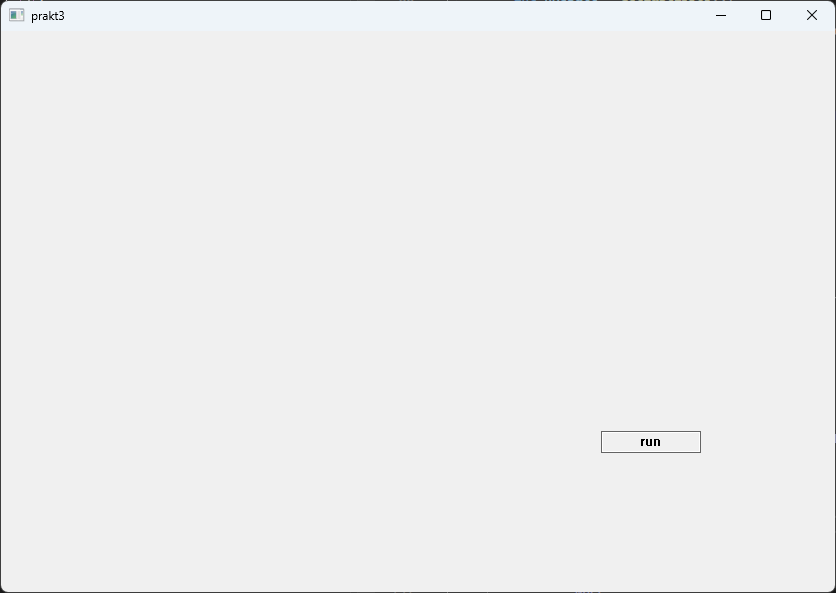


Рисунок 5 – Результат выполнения кода

На рисунке 6 изображен результат выполнения кода при нажатии кнопки открытия дочернего диалогового окна.

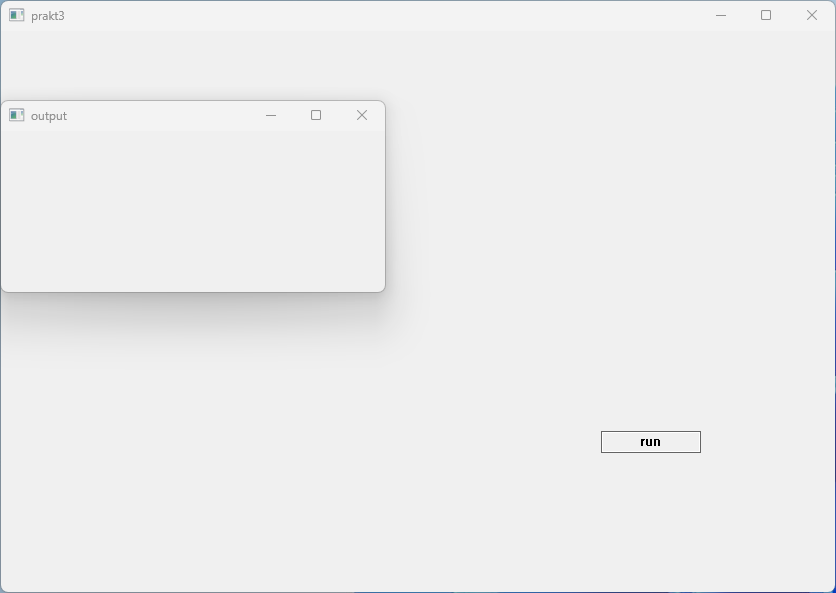


Рисунок 6 – Результат выполнения кода при нажатии кнопки «run»

На рисунке 7 изображен результат при нажатии клавиши на клавиатуре «Right Shift».

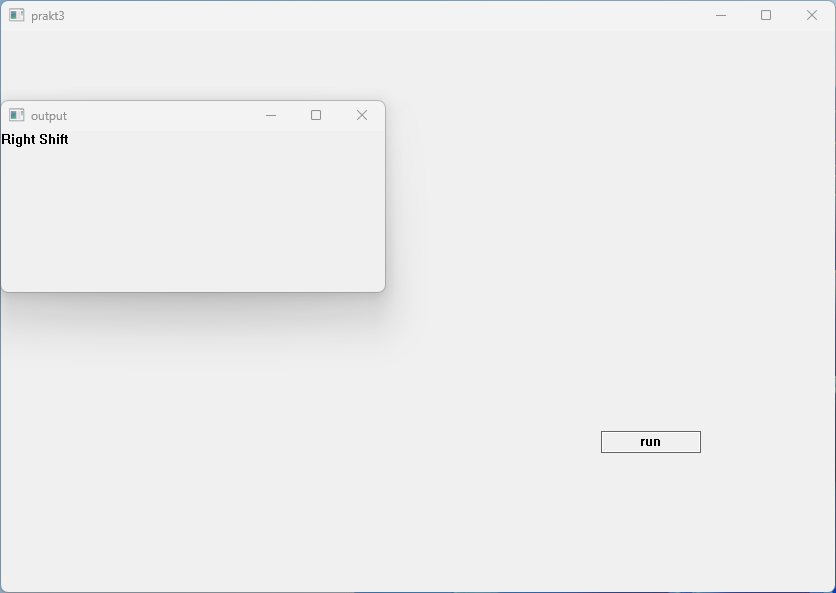


Рисунок 7 – Результат выполнения кода при нажатии «Right shift» на клавиатуре

Данный вариант диалогового окна является верным, так как реализованы все необходимые методы и соблюдены условия задачи.

### **4 Задание четвертое. Ввод только чисел через окно редактирования и вывод введенного в окне сообщений**

Разработать модуль, содержащий меню из двух кнопок "файл" и "помощь". В рабочей области расположено окно редактирования (текстовое поле в середине главного окна), которое позволяет вводить только числа без знаков препинания, букв, спец клавиш. Пункт меню "файл" содержит только одну команду "показать число", эта команда вызывает окно с сообщением, в которое передаётся информация из окна редактирования.

Создается главное окно с меню, состоящим из двух кнопок «File» и «Help», также окно редактирования в середине главного окна. В окно редактирования можно вводить только числа. При нажатии на «File» открывается дополнительное меню с пунктом «Show number». При нажатии «Show number» выводится окно сообщения с числом, которое было введено.

В приведенном ниже коде показана реализация создания строки редактирования и меню, а также получения чисел из строки редактирования и вывод в окно сообщения.

int WINAPI WinMain(HINSTANCE hInst, HINSTANCE hPrevInstance, LPSTR IpszCmdParam, int cmdShow)

{

MSG msg;

myRegistryClass(hInst, cmdShow);

InitInstance(hInst, cmdShow);

edit = CreateWindow(L"edit", L"", WS\_CHILD | WS\_VISIBLE | ES\_NUMBER, 142, 260, 300, 20, hwndA, NULL, hInst, NULL);

HMENU hmenu1;

hmenu1 = CreateMenu();

HMENU popmenu;

popmenu = CreatePopupMenu();

AppendMenu(hmenu1, MF\_STRING | MF\_POPUP, (UINT\_PTR)popmenu, L"&File");

AppendMenu(hmenu1, MF\_STRING, 1, L"&Help");

AppendMenu(popmenu, MF\_STRING, (UINT)menu::file, L"Show number");

SetMenu(hwndA, hmenu1);

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

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return 0;

}

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

{

switch (message)

{

case WM\_GETMINMAXINFO:

{

LPMINMAXINFO lpMMI = (LPMINMAXINFO)lParam;

lpMMI->ptMinTrackSize.x = GetSystemMetrics(SM\_CXSCREEN) / 2;

lpMMI->ptMinTrackSize.y = GetSystemMetrics(SM\_CYSCREEN) / 2;

break;

}

case WM\_SIZE:

{

int width = LOWORD(lParam);

int height = HIWORD(lParam);

MoveWindow(edit, ((width / 2) - 150), ((height / 2) - 10), 300, 20, TRUE);

break;

}

case WM\_COMMAND:

{

if ((menu)wParam == menu::file)

{

TCHAR command[1024];

GetWindowText(edit, command, 1024);

MessageBox(hwndA, command, L"Number", MB\_OK | MB\_ICONINFORMATION);

}

break;

}

case WM\_DESTROY: PostQuitMessage(0); break;

default: return DefWindowProc(hwnd, message, wParam, lParam); break;

}

}

На рисунке 8 изображен результат выполнения кода. На нем изображены главное окно с строкой ввода и меню.

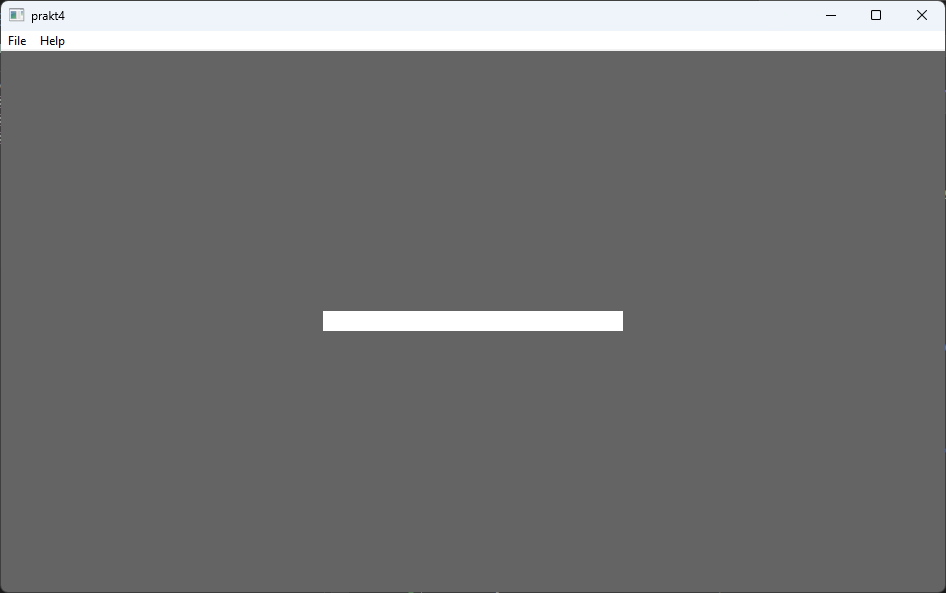


Рисунок 8 – Результат выполнения кода

На рисунке 9 изображен результат выполнения кода при вводе чисел в строку редактирования.

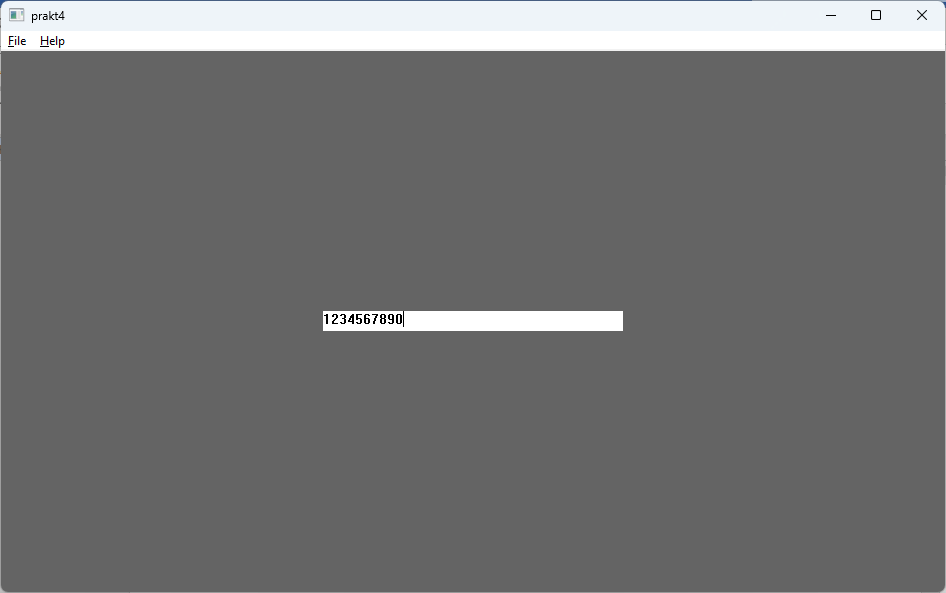


Рисунок 9 – Результат выполнения кода при вводе чисел в строку редактирования

На рисунке 10 изображен результат выполнения кода при выборе пункта «File» в меню.

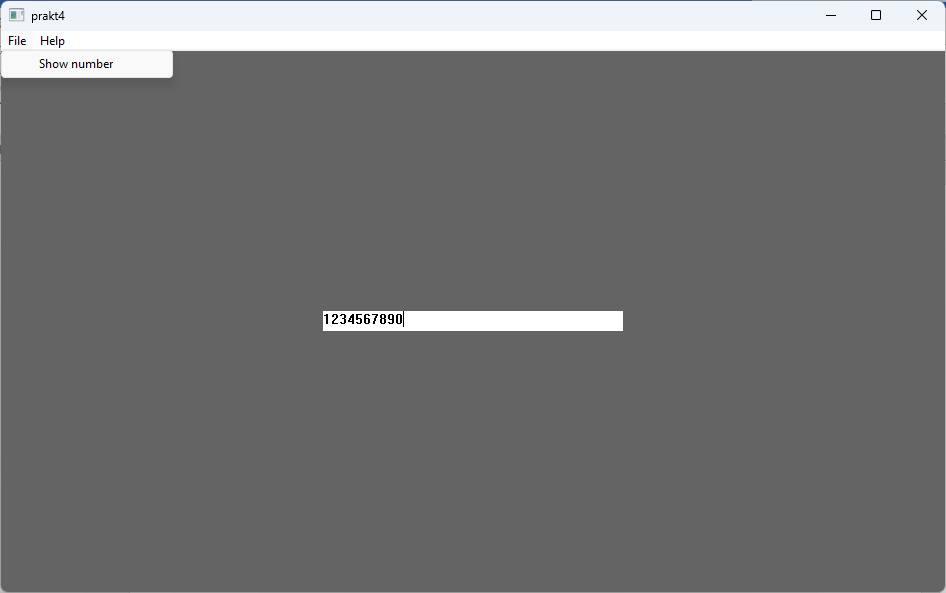


Рисунок 10 – Результат выполнения кода при выборе пункта «File» в меню

На рисунке 11 изображен результат выполнения кода при выборе пункта «Show number» в меню.

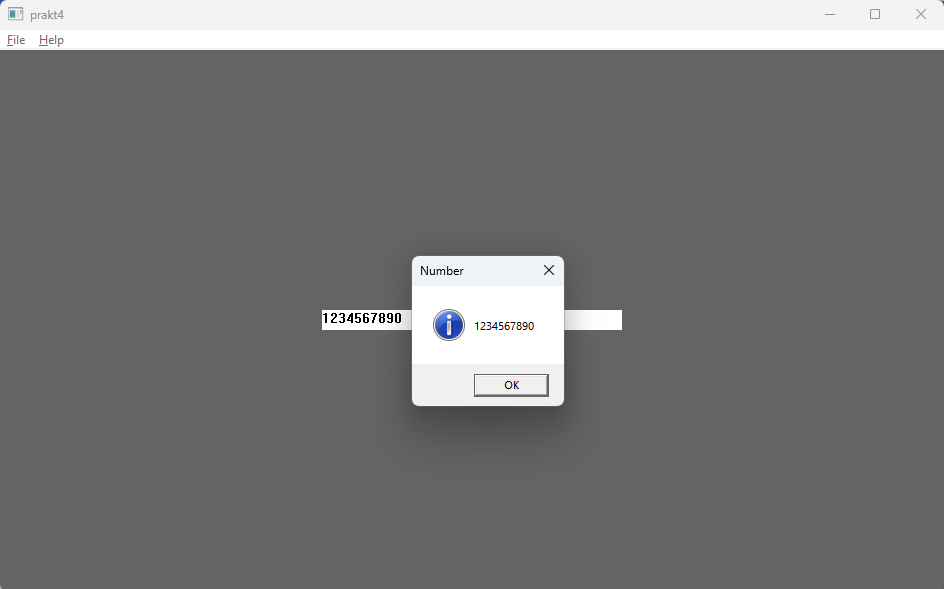


Рисунок 11 - результат выполнения кода при выборе пункта «Show number» в меню

Данный вариант диалогового окна является верным, так как реализованы все необходимые методы и соблюдены условия задачи.

### **5 Задание пятое. Выбор вопросов с возможностью выбора ответов.**

Разработать программный модуль, в главном окне которого расположено 3 кнопки «вопрос 1», «вопрос 2», «вопрос 3», каждая кнопка вызывает отдельное окно, в котором расположен некоторый вопрос и две кнопки ответа. При выборе кнопки с неправильным ответом, выдаётся сообщение о том, что ответ неверный и закрывается окно. При выборе кнопки с правильным ответом появляется сообщение о том, что ответ верный и вопрос "выбрать ли другой вопрос?", где будет 3 варианта - две кнопки с оставшимися вопросами и кнопка "нет", закрывающая окно. Для того, чтобы закрыть главное окно используется пункт меню «Файл» ‎‎‎‎‎→ «‎‎Выход». При выборе пункта меню "помощь" должно открываться справочное окно, где написано, что надо делать в этой программе.

Создается главное окно с пунктами в меню «Файл» и «Помощь». В пользовательской зоне окна также представлены кнопки «Вопрос 1», «Вопрос 2», «Вопрос 3». При нажатии кнопки «Файл» открывается выпадающее меню с пунктом «Выход», при нажатии которой будет осуществлен выход из программы. При нажатии «Помощь» открывается справочное окно, где написано, что делать в программе. При нажатии кнопок с номерами вопросов открываются вопросы и варианты ответов «Да» и «Нет». При правильном ответе открывается окно с надписью «Ответ верный. Выберите действие» и кнопками следующих вопросов и кнопкой отмены. При неправильном ответе открывается окно сообщения с информацией о том, что ответ неверный.

Ниже представлен код, в котором реализовано создание и обработка окон и всех необходимых кнопок.

LRESULT CALLBACK WndProc2(HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)

{

if (message == WM\_COMMAND)

{

ShowWindow(hwndB, SW\_HIDE);

DestroyWindow(but1);

DestroyWindow(but2);

DestroyWindow(but3);

DestroyWindow(cancel);

if ((menu)wParam == menu::but1)

{

SendMessage(hwndA, WM\_COMMAND, (WPARAM)menu::button1, NULL);

}

else if ((menu)wParam == menu::but2)

{

SendMessage(hwndA, WM\_COMMAND, (WPARAM)menu::button2, NULL);

}

else if ((menu)wParam == menu::but3)

{

SendMessage(hwndA, WM\_COMMAND, (WPARAM)menu::button3, NULL);

}

}

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

}

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

{

switch (message)

{

case WM\_GETMINMAXINFO:

{

LPMINMAXINFO lpMMI = (LPMINMAXINFO)lParam;

lpMMI->ptMinTrackSize.x = GetSystemMetrics(SM\_CXSCREEN) / 2;

lpMMI->ptMinTrackSize.y = GetSystemMetrics(SM\_CYSCREEN) / 2;

break;

}

case WM\_SIZE:

{

int width = LOWORD(lParam);

int height = HIWORD(lParam);

break;

}

case WM\_COMMAND:

{

if ((menu)wParam == menu::exit)

{

PostQuitMessage(0);

}

else if ((menu)wParam == menu::help)

{

std::wstring text = L"В этой программе можно выбрать вопросы и отвечать на них.\n"\

L"При нажатии кнопки Выход в пункте меню Файл можно закрыть программу.\n"\

L"При нажатии кнопки Помощь открывается данное окно.\n"\

L"Чтобы выбрать вопрос необходимо нажать на одну из кнопкок с номером вопроса.";

MessageBox(hwndA, text.c\_str(), L"Помощь", MB\_OK | MB\_ICONINFORMATION);

}

else if ((menu)wParam == menu::button1)

{

auto id = MessageBox(hwndA, L"Великая отественная война началась в 1941?", L"Вопрос 1", MB\_YESNO | MB\_ICONQUESTION);

if (id == IDYES)

{

ShowWindow(hwndB, SW\_SHOW);

but2 = CreateWindow(L"button", L"Вопрос 2", WS\_CHILD | WS\_VISIBLE | BS\_FLAT | BS\_VCENTER | BS\_PUSHBUTTON, 20, 200, 100, 22, hwndB, (HMENU)menu::but2, NULL, NULL);

but3 = CreateWindow(L"button", L"Вопрос 3", WS\_CHILD | WS\_VISIBLE | BS\_FLAT | BS\_VCENTER | BS\_PUSHBUTTON, 120, 200, 100, 22, hwndB, (HMENU)menu::but3, NULL, NULL);

cancel = CreateWindow(L"button", L"Отмена", WS\_CHILD | WS\_VISIBLE | BS\_FLAT | BS\_VCENTER | BS\_PUSHBUTTON, 220, 200, 100, 22, hwndB, (HMENU)menu::cancel, NULL, NULL);

}

else if (id == IDNO)

{

MessageBox(hwndA, L"Ответ неверный!", L"", MB\_OK | MB\_ICONERROR);

}

}

else if ((menu)wParam == menu::button2)

{

auto id = MessageBox(hwndA, L"На значке BMW изображен зонт?", L"Вопрос 2", MB\_YESNO | MB\_ICONQUESTION);

if (id == IDYES)

{

MessageBox(hwndA, L"Ответ неверный!", L"", MB\_OK | MB\_ICONERROR);

}

else if (id == IDNO)

{

ShowWindow(hwndB, SW\_SHOW);

but1 = CreateWindow(L"button", L"Вопрос 1", WS\_CHILD | WS\_VISIBLE | BS\_FLAT | BS\_VCENTER | BS\_PUSHBUTTON, 20, 200, 100, 22, hwndB, (HMENU)menu::but1, NULL, NULL);

but3 = CreateWindow(L"button", L"Вопрос 3", WS\_CHILD | WS\_VISIBLE | BS\_FLAT | BS\_VCENTER | BS\_PUSHBUTTON, 120, 200, 100, 22, hwndB, (HMENU)menu::but3, NULL, NULL);

cancel = CreateWindow(L"button", L"Отмена", WS\_CHILD | WS\_VISIBLE | BS\_FLAT | BS\_VCENTER | BS\_PUSHBUTTON, 220, 200, 100, 22, hwndB, (HMENU)menu::cancel, NULL, NULL);

}

}

else if ((menu)wParam == menu::button3)

{

auto id = MessageBox(hwndA, L"Трава зеленая?", L"Вопрос 3", MB\_YESNO | MB\_ICONQUESTION);

if (id == IDYES)

{

ShowWindow(hwndB, SW\_SHOW);

but1 = CreateWindow(L"button", L"Вопрос 1", WS\_CHILD | WS\_VISIBLE | BS\_FLAT | BS\_VCENTER | BS\_PUSHBUTTON, 20, 200, 100, 22, hwndB, (HMENU)menu::but1, NULL, NULL);

but2 = CreateWindow(L"button", L"Вопрос 2", WS\_CHILD | WS\_VISIBLE | BS\_FLAT | BS\_VCENTER | BS\_PUSHBUTTON, 120, 200, 100, 22, hwndB, (HMENU)menu::but2, NULL, NULL);

cancel = CreateWindow(L"button", L"Отмена", WS\_CHILD | WS\_VISIBLE | BS\_FLAT | BS\_VCENTER | BS\_PUSHBUTTON, 220, 200, 100, 22, hwndB, (HMENU)menu::cancel, NULL, NULL);

}

else if (id == IDNO)

{

MessageBox(hwndA, L"Ответ неверный!", L"", MB\_OK | MB\_ICONERROR);

}

}

break;

}

case WM\_DESTROY: PostQuitMessage(0); break;

default: return DefWindowProc(hwnd, message, wParam, lParam); break;

}

}

BOOL InitInstance(HINSTANCE hInst, int Mode)

{

hwndA = CreateWindow(

czClassName,

czFormName,

WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT,

CW\_USEDEFAULT,

900,

600,

NULL,

NULL,

hInst,

NULL);

WNDCLASS cw;

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

cw.cbClsExtra = 0;

cw.cbWndExtra = 0;

cw.hbrBackground = (HBRUSH)(COLOR\_WINDOW + 2);

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

cw.hIcon = LoadIcon(NULL, IDI\_APPLICATION);

cw.hInstance = hInst;

cw.lpfnWndProc = WndProc2;

cw.lpszClassName = class\_name;

cw.lpszMenuName = NULL;

RegisterClass(&cw);

hwndB = CreateWindow(

class\_name,

czFormName,

NULL,

CW\_USEDEFAULT,

CW\_USEDEFAULT,

500,

300,

NULL,

NULL,

hInst,

NULL);

HWND static\_text = CreateWindow(L"static", L"Ответ верный. Выберите действие", WS\_CHILD | WS\_VISIBLE, 20, 100, 240, 20, hwndB, NULL, hInst, NULL);

ShowWindow(hwndB, SW\_HIDE);

button1 = CreateWindow(L"button", L"Вопрос 1", WS\_CHILD | WS\_VISIBLE | BS\_FLAT | BS\_VCENTER | BS\_PUSHBUTTON, 300, 400, 100, 22, hwndA, (HMENU)menu::button1, NULL, NULL);

button2 = CreateWindow(L"button", L"Вопрос 2", WS\_CHILD | WS\_VISIBLE | BS\_FLAT | BS\_VCENTER | BS\_PUSHBUTTON, 500, 400, 100, 22, hwndA, (HMENU)menu::button2, NULL, NULL);

button3 = CreateWindow(L"button", L"Вопрос 3", WS\_CHILD | WS\_VISIBLE | BS\_FLAT | BS\_VCENTER | BS\_PUSHBUTTON, 700, 400, 100, 22, hwndA, (HMENU)menu::button3, NULL, NULL);

ShowWindow(hwndA, Mode);

UpdateWindow(hwndA);

return TRUE;

}

/\*Главная функция\*/

int WINAPI WinMain(HINSTANCE hInst, HINSTANCE hPrevInstance, LPSTR IpszCmdParam, int cmdShow)

{

MSG msg;

myRegistryClass(hInst, cmdShow);

InitInstance(hInst, cmdShow);

HMENU hmenu1;

hmenu1 = CreateMenu();

HMENU popmenu;

popmenu = CreatePopupMenu();

AppendMenu(hmenu1, MF\_STRING | MF\_POPUP, (UINT\_PTR)popmenu, L"&Файл");

AppendMenu(hmenu1, MF\_STRING, (UINT)menu::help, L"&Помощь");

AppendMenu(popmenu, MF\_STRING, (UINT)menu::exit, L"Выход");

SetMenu(hwndA, hmenu1);

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

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return 0;

}

На рисунке 12 представлен результат выполнения кода. На нем изображено главное окно, меню и все кнопки.

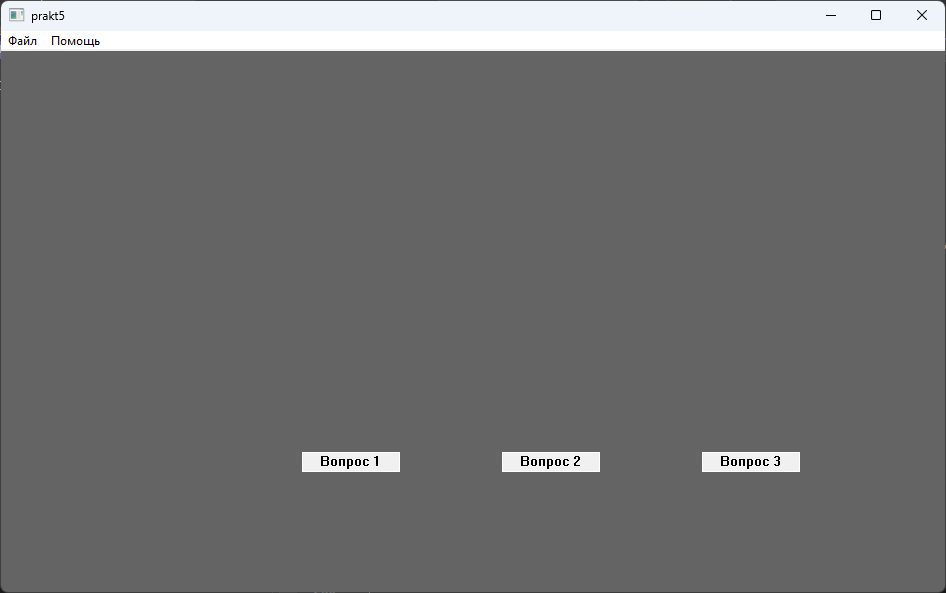


Рисунок 12 – Результат выполнения кода при открытии программы

На рисунке 13 представлен результат выполнения кода при нажатии пункта меню «Файл».



Рисунок 13 – Результат выполнения кода нажатия кнопки «Файл»

На рисунке 14 изображен результат выполнения кода, при нажатии пункта меню «Помощь».

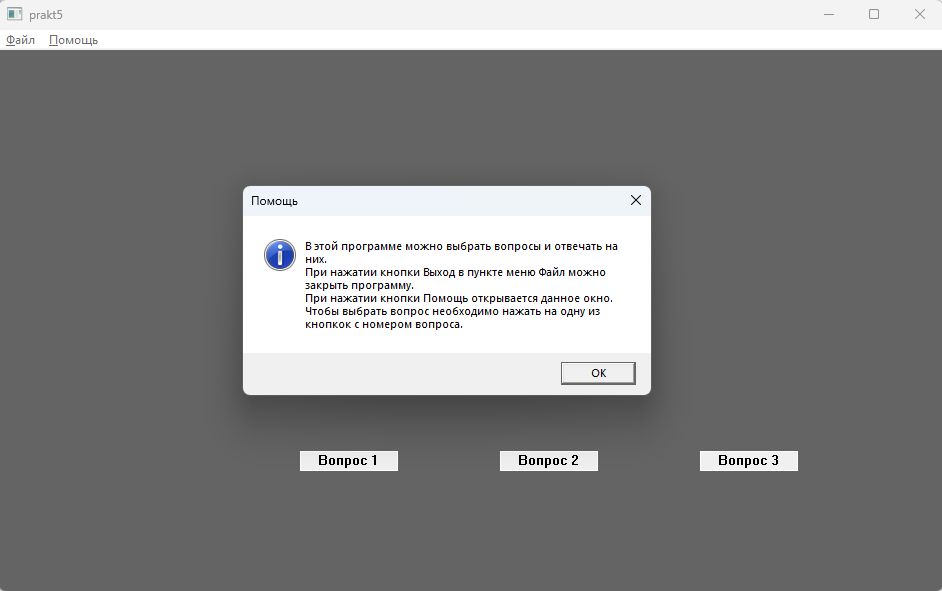


Рисунок 14 – Результат выполнения кода

На рисунке 15 изображен результат выполнения кода, при выборе любого вопроса.

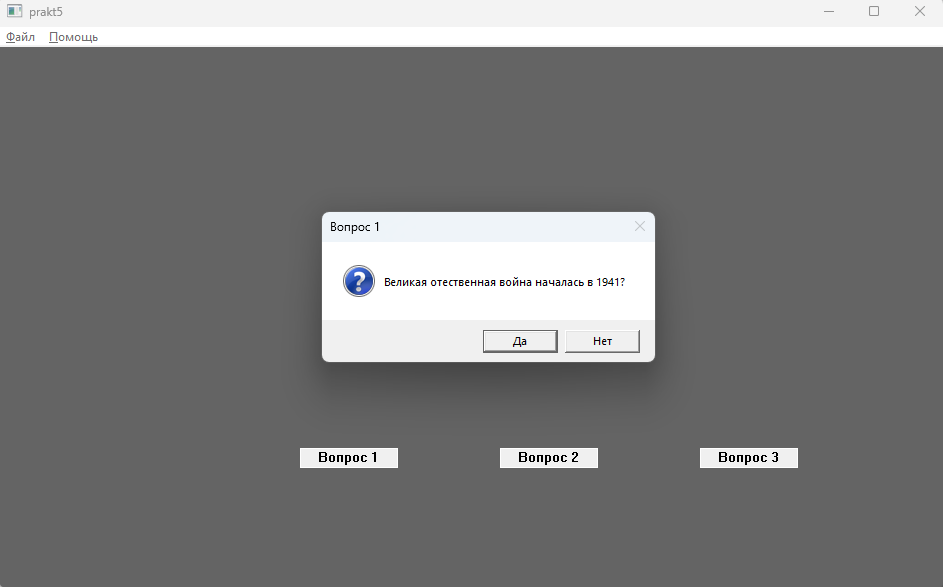


Рисунок 15 – Результат выполнения кода

На рисунке 16 изображен результат выполнения кода при выборе правильного ответа.

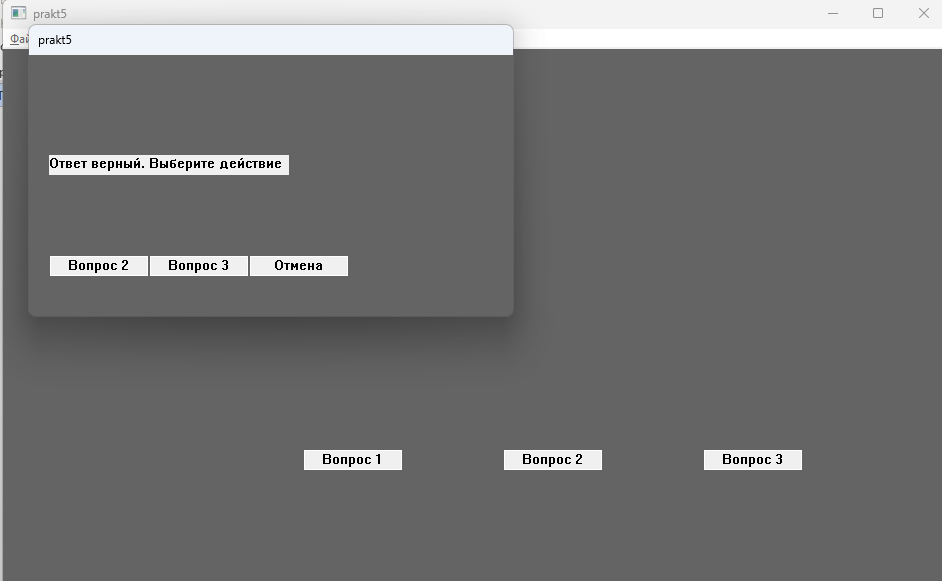


Рисунок 16 – Результат выполнения кода

На рисунке 17 представлен результат выполнения кода при выборе следующего вопроса.

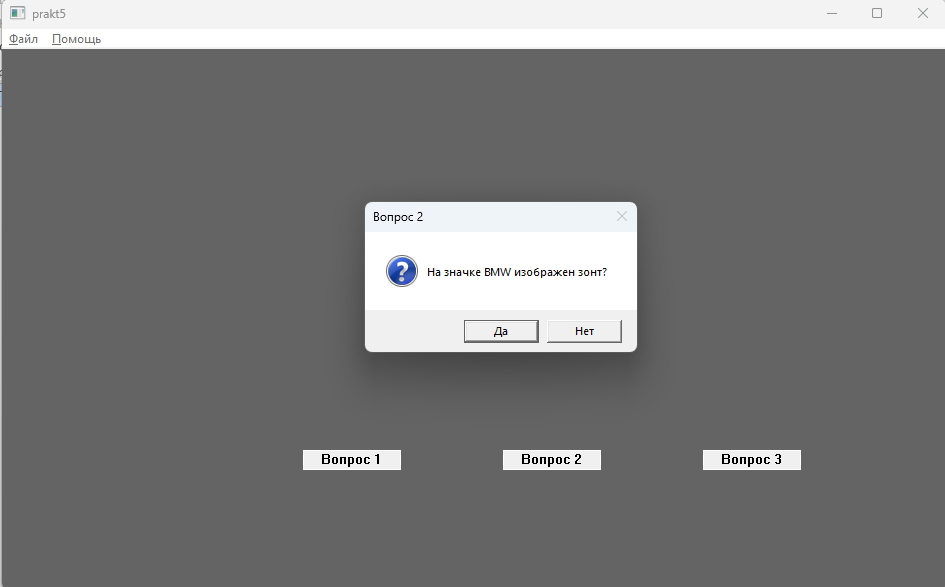


Рисунок 17 – Результат выполнения кода

На рисунке 18 представлен результат выполнения кода при выборе неправильного варианта ответа.

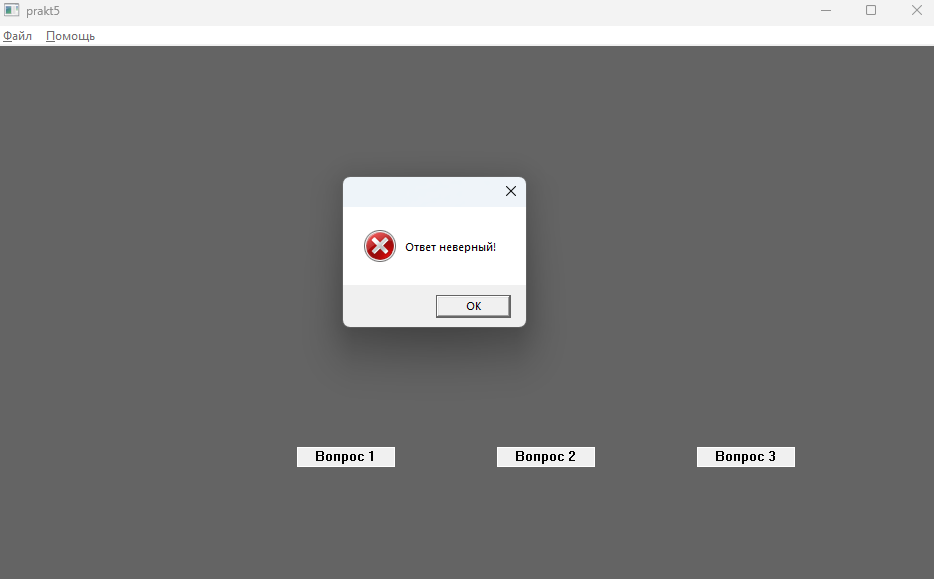


Рисунок 18 – Результат выполнения кода

Данный вариант диалогового окна является верным, так как реализованы все необходимые методы и соблюдены условия задачи.