

Министерство образования и науки Российской Федерации  
Федеральное агентство по образованию  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Вятский государственный университет»

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Разработка многооконного приложения средствами WinAPI

Отчет по лабораторной работе №3 дисциплины  
«Технологии программирования»

Выполнил студент группы ИВТ-22\_\_\_\_\_ /Крючков И. С/  
Проверил \_\_\_\_\_ /Долженкова М. Л./

Киров 2022

## 1. Задание

Написать программу с использованием WinApi для преобразования матрицы к треугольному виду и нахождения суммы элементов главной диагонали. На главном окне программы реализовать ввод матрицы, а также выбор необходимой операции – нахождение суммы элементов главной диагонали, либо преобразование матрицы к треугольному виду. В дочернем окне вывести результаты работы программы в соответствии с выбранной операцией.

### Особенности

Windows-приложение состоит как минимум из двух основных процедур. **WinMain()**, составляющей основу приложения. Данная функция является точкой входа в приложение. Выполняется при запуске программы, инициализирует все необходимые данные приложения, определяет его окна и запускает главное окно приложения. После создания главного окна запускается цикл обработки сообщений, извлекаемых из очереди. При получении сообщения вызывается процедура **wndProc()**. Представляет собой case структуру, позволяющую выбрать обработчик на основании id сообщения. Цикл обработки сообщения так же поддерживается на основе вызова winAPI.

## 2. Результаты работы программы

Lab3

Размер матрицы матрицы:

1	2
4	3

☐ Треугольный вид  
☒ Сумма эл-в гл. диагонали

Рисунок 1 – Главное окно программы

Result

**Сумма элементов главной диагонали: 4**

Рисунок 2 – Результат суммы элементов главной диагонали в дочернем окне

Result

**Треугольная матрица:**

4	3
0	1.25

Рисунок 3 – Результат приведения матрицы к треугольному виду в дочернем окне



```

        if (a[h][i] != 0)
            swap(a[z], a[h]);
    }

    if (f == 1) {
        return 1;
    }

    // прямой ход
    for (int j = i + 1; j < n; j++)
    {
        double m = -a[j][i] / a[i][i];
        for (int k = i; k < n; k++)
            a[j][k] += a[i][k] * m;
    }
    return 0;
}

double diagonal_sum(double** matr, int n)
{
    double sum = 0;
    for (int i = 0; i < n; i++)
    {
        sum += matr[i][i];
    }
    return sum;
}

// Точка входа
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdline, int nShowCmd)
{
    MSG msg;
    memset(&msg, 0, sizeof(msg));

    // главное окно
    WNDCLASSEX wndMain;
    memset(&wndMain, 0, sizeof(wndMain));
    wndMain.cbSize = sizeof(wndMain); // размер структуры
    wndMain.style = CS_VREDRAW | CS_WREDRAW; // перерисовка при изменении размеров окна
    wndMain.lpfnWndProc = (WNDPROC)WndMainProc; // обработчик окна
    wndMain.hInstance = hInstance; // дескриптор экземпляра
    wndMain.hbrBackground = (HBRUSH)COLOR_WINDOW; // фон окна
    wndMain.lpszClassName = L"MainWndClass";

    if (!RegisterClassEx(&wndMain))
    {
        int nResult = GetLastError();
        MessageBox(NULL, L"Класс главного окна не был создан!", L"Ошибка", MB_ICONERROR);
    }

    mainWindow = CreateWindowEx(
        NULL,
        L"MainWndClass",
        L"Lab3",
        WS_DLGFRAME | WS_SYSMENU | WS_MINIMIZEBOX,
        CW_USEDEFAULT, // x
        0, // y
        widthMainWnd, // width
        heightMainWnd, // height
        NULL, // id родительского окна
        NULL,
        hInstance,
        NULL
    );

    if (!mainWindow)
    {
        MessageBox(NULL, L"Главное окно не было создано!", L"Ошибка", MB_ICONERROR);
    }

    // дочернее окно
    WNDCLASSEX wndChild;
    memset(&wndChild, 0, sizeof(wndChild));

```

```

wndChild.cbSize = sizeof(wndChild); // размер структуры
wndChild.style = CS_VREDRAW | CS_HREDRAW; // перерисовка при изменении размеров окна
wndChild.lpfnWndProc = (WNDPROC)WndChildProc; // обработчик окна
wndChild.hInstance = hInstance; // дескриптор экземпляра
wndChild.hbrBackground = (HBRUSH)COLOR_WINDOW; // фон окна
wndChild.lpszClassName = L"ChildWndClass";

if (!RegisterClassEx(&wndChild))
{
    int nResult = GetLastError();
    MessageBox(NULL, L"Класс дочернего окна не был создан!", L"Ошибка", MB_ICONERROR);
}

childWindow = CreateWindowEx(
    NULL,
    L"ChildWndClass",
    L"Result",
    WS_DLGFRAME,
    CW_USEDEFAULT, // x
    0, // y
    widthChildWnd, // width
    heightChildWnd, // height
    mainWindow, // id родительского окна
    NULL,
    hInstance,
    NULL
);

if (!childWindow)
{
    MessageBox(NULL, L"Дочернее окно не было создано!", L"Ошибка", MB_ICONERROR);
}

ShowWindow(mainWindow, SW_SHOW);
UpdateWindow(mainWindow);

while (GetMessage(&msg, NULL, 0, 0))
{
    TranslateMessage(&msg);
    DispatchMessage(&msg);
}

return 0;
}

LRESULT CALLBACK WndMainProc(HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    unsigned int tLength;
    char *editValue = nullptr;
    char *err = NULL;

    switch (message)
    {
        case WM_CREATE:
            sizeMatrixText = CreateWindow(L"STATIC", L"Размер матрицы матрицы:", WS_VISIBLE |
WS_CHILD, 20, 20, 200, 20, hwnd, NULL, NULL, NULL);
            editSizeMatrix = CreateWindow(L"EDIT", L"", WS_VISIBLE | WS_CHILD | WS_BORDER |
ES_NUMBER, 260, 20, 40, 20, hwnd, NULL, NULL, NULL);
            Edit_LimitText(editSizeMatrix, 1); // ограничение кол-ва символов в поле ввода

            submitButton1 = CreateWindow(L"BUTTON", L"Принять", WS_VISIBLE | WS_CHILD |
WS_BORDER, 320, 20, 60, 20, hwnd, (HMENU)1, NULL, NULL);

            radioButton1 = CreateWindow(L"BUTTON", L"Треугольный вид", WS_CHILD | WS_VISIBLE |
BS_AUTORADIOBUTTON,
                20, 300, 150, 20, hwnd, (HMENU)3, NULL, NULL);
            radioButton2 = CreateWindow(L"BUTTON", L"Сумма эл-в гл. диагонали", WS_CHILD |
WS_VISIBLE | BS_AUTORADIOBUTTON,
                20, 320, 200, 20, hwnd, (HMENU)4, NULL, NULL);

            resultButton = CreateWindow(L"BUTTON", L"Результат", WS_VISIBLE | WS_CHILD |
WS_BORDER, 300, 320, 80, 20, hwnd, (HMENU)5, NULL, NULL);

            break;
    }
}

```

```

case WM_COMMAND:
    switch (LOWORD(wParam))
    {
    case 1: // принять размеры матрицы
        oldm = m;
        tLength = GetWindowTextLength(editSizeMatrix) + 1;
        editValue = new char[tLength];
        GetWindowTextA(editSizeMatrix, editValue, tLength + 1);
        m = atoi(editValue);
        delete[] editValue;

        if (m < 1)
        {
            MessageBox(hwnd, L"Допустимое значение: от 1 до 9", L"Ошибка",
                MB_OK);

            return 0;
        }

        clearMatrix(masMatrix, oldm);

        for (int i = 0; i < oldm; i++)
        {
            for (int j = 0; j < oldm; j++)
            {
                DestroyWindow(matrixEdit[i][j]);
            }
        }
        DestroyWindow(submitButton2);

        //Создание таблиц
        for (int i = 0; i < m; i++)
        {
            for (int j = 0; j < m; j++)
            {
                matrixEdit[i][j] = CreateWindow(L"EDIT", L"0", WS_VISIBLE
                    | WS_CHILD | WS_BORDER,
                    j * 40 + 20, i * 25 + 55, 40, 25, hwnd, NULL, NULL,
                    NULL);
                Edit_LimitText(matrixEdit[i][j], 3); // ограничение кол-ва
                символов в поле ввода
            }
        }

        submitButton2 = CreateWindow(L"BUTTON", L"Принять", WS_VISIBLE | WS_CHILD
            | WS_BORDER,
            m * 20, m * 25 + 60, 80, 20, hwnd, (HMENU)2, NULL, NULL);

        matrState = 0;

        break;
    case 2: // принять матрицу
        if (matrState == 0)
        {
            try
            {
                masMatrix = new double* [m];
                for (int i = 0; i < m; i++)
                {
                    masMatrix[i] = new double[m];
                    for (int j = 0; j < m; j++)
                    {
                        tLength =
                            GetWindowTextLength(matrixEdit[i][j]) + 1;

                        editValue = new char[tLength];
                        GetWindowTextA(matrixEdit[i][j],
                            editValue, tLength + 1);

                        masMatrix[i][j] = strtod(editValue, &err);
                        Edit_Enable(matrixEdit[i][j], false);
                        if (*err)
                        {
                            throw 100;
                        }
                        delete[] editValue;
                    }
                }
                SetWindowText(submitButton2, L"Изменить");
                matrState = 1;
            }
            catch (...)
            {
                // обработка исключений
            }
        }
    }
}

```

```

        }
        catch (...)
        {
            for (int i = 0; i < m; i++)
            {
                for (int j = 0; j < m; j++)
                {
                    Edit_Enable(matrixEdit[i][j], true);
                }
            }

            MessageBox(hwnd, L"Вводите числа!", L"Ошибка", MB_OK);
            return -1;
        }
    }
    else
    {
        clearMatrix(masMatrix, m);
        matrState = 0;
        SetWindowText(submitButton2, L"Принять");
        for (int i = 0; i < m; i++)
        {
            for (int j = 0; j < m; j++)
            {
                Edit_Enable(matrixEdit[i][j], true);
            }
        }
    }

    break;

case 3: // radio 1;
    switch (Button_GetCheck(radioButton1))
    {
        case BST_CHECKED:
            operationType = 0;
            break;
    }

    break;

case 4: // radio 2;
    switch (Button_GetCheck(radioButton2))
    {
        case BST_CHECKED:
            operationType = 1;
            break;
    }

    break;

case 5: // результаты
    if (operationType == -1){
        MessageBox(hwnd, L"Необходимо выбрать операцию", L"Ошибка", MB_OK);
    }
    else if (matrState == 0)
    {
        MessageBox(hwnd, L"Необходимо принять матрицу", L"Ошибка", MB_OK);
    }
    else
    {
        EnableWindow(mainWindow, false);
        ShowWindow(childWindow, SW_SHOW);
        SetFocus(childWindow);
    }
    break;
}
break;
case WM_DESTROY:
    PostQuitMessage(0);
    break;
default:
    return DefWindowProc(hwnd, message, wParam, lParam);
}
}

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

```



```

switch (message)
{
case WM_SHOWWINDOW:
    for (int i = 0; i < m; i++)
    {
        for (int j = 0; j < m; j++)
        {
            DestroyWindow(matrixEdit2[i][j]);
        }
    }
    DestroyWindow(closeButton);

    if (operationType == 0){
        double** resMatrix = new double* [m];
        copyMatrix(resMatrix, masMatrix, m);

        char trer = get_trangle_matrix(masMatrix, resMatrix, m);
        if (trer == 0)
        {
            SetWindowPos(hwnd, NULL, 0, 0, widthChildWnd, heightChildWnd,
SWP_NOMOVE);

            resultText = CreateWindow(L"STATIC", L"Треугольная матрица:",
WS_VISIBLE | WS_CHILD, 20, 20, 300, 20, hwnd, NULL, NULL, NULL);
            closeButton = CreateWindow(L"BUTTON", L"Заккрыть", WS_VISIBLE |
WS_CHILD | WS_BORDER, 300, 320, 80, 20, hwnd, (HMENU)1, NULL, NULL);

            //Создание таблицы
            for (int i = 0; i < m; i++)
            {
                for (int j = 0; j < m; j++)
                {
                    wstringstream wss;
                    wss << floor(resMatrix[i][j] * 1000.0 + 0.5) /
1000.0;

                    matrixEdit2[i][j] = CreateWindow(
                        L"EDIT", wss.str().c_str(), WS_VISIBLE |
WS_CHILD | WS_BORDER,
                        j * 40 + 20, i * 25 + 55, 40, 25, hwnd,
                        NULL, NULL, NULL);

                    Edit_Enable(matrixEdit2[i][j], false);
                }
            }
        }
        else
        {
            SetWindowPos(hwnd, NULL, 0, 0, widthChildWnd, heightChildWndSm,
SWP_NOMOVE);

            resultText = CreateWindow(L"STATIC", L"Невозможно получить
треугольную матрицу", WS_VISIBLE | WS_CHILD, 20, 20, 300, 20, hwnd, NULL, NULL, NULL);
            closeButton = CreateWindow(L"BUTTON", L"Заккрыть", WS_VISIBLE |
WS_CHILD | WS_BORDER, 300, 50, 80, 20, hwnd, (HMENU)1, NULL, NULL);
        }
        clearMatrix(resMatrix, m);
    }
    else
    {
        SetWindowPos(hwnd, NULL, 0, 0, widthChildWnd, heightChildWndSm,
SWP_NOMOVE);

        resultText = CreateWindow(L"STATIC", L"Сумма элементов главной диагонали:",
WS_VISIBLE | WS_CHILD, 20, 20, 300, 20, hwnd, NULL, NULL, NULL);
        closeButton = CreateWindow(L"BUTTON", L"Заккрыть", WS_VISIBLE | WS_CHILD |
WS_BORDER, 300, 50, 80, 20, hwnd, (HMENU)1, NULL, NULL);

        double resSum = diagonal_sum(masMatrix, m);

        wstringstream wss;
        wss << resSum;
        resultSumText = CreateWindow(L"STATIC", wss.str().c_str(), WS_VISIBLE |
WS_CHILD, 300, 20, 50, 20, hwnd, NULL, NULL, NULL);
    }
    break;
case WM_COMMAND:
    switch (LOWORD(wParam))
    {
    case 1: // закрыть
        DestroyWindow(resultText);
        ShowWindow(childWindow, SW_HIDE);
    }
}
}
}

```

```

        EnableWindow(mainWindow, true);
        SetFocus(mainWindow);
        break;
    }

    break;
case WM_DESTROY:
    PostQuitMessage(0);
    return 0;
    break;
default:
    return DefWindowProc(hwnd, message, wParam, lParam);
}
}

```

#### 4. Вывод

В ходе выполнения лабораторной работы была разработана многооконная программа с использованием функций WinApi, реализующая ввод матрицы, преобразование матрицы к треугольному виду и нахождение суммы элементов главной диагонали.