**Министерство науки и высшего образования РФ**

**Федеральное государственное бюджетное образовательное**

**учреждение высшего образования**

**Московский авиационный институт**

**(национальный исследовательский университет)**

**«МАИ»**

Кафедра 304

Пояснительная записка к курсовой работе

по дисциплине: «Конструирование программного обеспечения»

на тему: «Разработка 2D-графиков в классическом приложении Windows»

Выполнила:

студентка группы М3О-211Б-22 Мороз А.И. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Принял:

к.т.н., доц. каф. 304 Новиков П.В.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва

2024

Содержание

[Задание 3](#_Toc168734413)

[Теоретическая часть 3](#_Toc168734414)

[Классическое приложение Windows 3](#_Toc168734415)

[Создание окон 3](#_Toc168734416)

[Нажатие кнопок и создание новых окон 4](#_Toc168734417)

[Рисование графиков 5](#_Toc168734418)

[Классическое приложение Windows С++ для построения 2D-графиков 6](#_Toc168734419)

[Код программы 6](#_Toc168734420)

[Результат работы программы 16](#_Toc168734421)

[Вывод 19](#_Toc168734422)

[Список источников 19](#_Toc168734423)

# **Задание**

Разработать 2D-графики в классическом приложении Windows.

# **Теоретическая часть**

В данной курсовой работе рассматривается создание приложения на языке программирования C++ с использованием библиотеки WinAPI, которое позволяет пользователю выбирать и отображать различные графики математических функций. Программа состоит из двух окон: главное окно для выбора графиков и окно для отображения выбранного графика. Важное внимание уделяется созданию окон, обработке событий и отрисовке графиков.

# **Классическое приложение Windows**

Классическое приложение в C++ - приложение, которое может получить доступ к полному набору API Windows, а также выполняется в окне или в системной консоли.

Windows API (Application Programming Interface) - набор функций и интерфейсов, предоставляемых операционной системой Microsoft Windows для разработки приложений. WinAPI обеспечивает доступ к функциям операционной системы Windows, таким как создание окон, управление ресурсами, работа с файлами, реагирование на события и многое другое.

# **Создание окон**

Создание окон в Windows-программе осуществляется с помощью API функций, предоставляемых операционной системой. Основными компонентами при создании окон являются регистрация класса окна, создание и отображение окна, а также обработка сообщений.

Регистрация класса окна

Прежде чем создать окно, необходимо зарегистрировать его класс. Класс окна определяет основные характеристики окна, включая функцию обработки сообщений, стили окна и прочие атрибуты.

Пример:

wc.lpfnWndProc = GraphDisplayWindowProc; // Функция обработки сообщений //для этого окна

wc.hInstance = hInstance; // Экземпляр приложения

wc.lpszClassName = CLASS\_NAME; // Имя класса

// Регистрируем класс окна

RegisterClass(&wc);

Создание и отображение окна

После регистрации класса окна, создается само окно с помощью функции CreateWindowEx, которая принимает множество параметров, включая имя класса окна, заголовок, стили окна, координаты и размеры окна.

Пример:

hwnd = CreateWindowEx(

0, //Расширенный стиль окна

CLASS\_NAME, // Имя класса окна

L"Выбор графика", // Заголовок окна

WS\_OVERLAPPEDWINDOW, // Стандартный стиль окна

CW\_USEDEFAULT, CW\_USEDEFAULT, // Координаты окна

400, 400, // Размеры окна

NULL, // Дескриптор родительского окна

NULL, // Дескриптор меню окна

hInstance, // Дескриптор экземпляра приложения

NULL // Дополнительные параметры

);

ShowWindow(hwnd, SW\_SHOW); // Отображение окна

Обработка сообщений

Основная логика приложения реализована в функции обработки сообщений окна, которая реагирует на различные события, такие как создание окна, нажатие кнопок, перерисовка окна и его закрытие. Для обработки сообщений используется цикл сообщений.

Пример:

MSG msg = { };

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

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

# **Нажатие кнопок и создание новых окон**

Когда пользователь нажимает на одну из кнопок для выбора графика, программа создает новое окно для отображения соответствующего графика. Это реализуется в функции обработки сообщений:

case WM\_COMMAND:

switch (LOWORD(wParam))

{

case 1: // Выбор графика линейной функции

CreateGraphWindow(LINEAR);

break;

case 2: // Выбор графика синуса

CreateGraphWindow(SINUS);

break;

// Другие случаи

}

break;

Функция CreateGraphWindow создает новое окно с графиком:

void CreateGraphWindow(GraphType type)

{

GraphDisplayWindow\* graphDisplayWindow = new GraphDisplayWindow(g\_hInstance, type);

graphDisplayWindow->Create();

}

# **Рисование графиков**

Рисование графиков осуществляется в функции обработки сообщений окна графика, когда приходит сообщение WM\_PAINT. Для каждого типа графика используются свои математические формулы, которые определяют координаты точек графика.

Линейная функция

График линейной функции задается уравнением вида y = kx + b. В программе для этого используется следующая логика:

MoveToEx(hdc, 0, Y - (int)(k \* (-X) + b), NULL);

for (int x = -X + 1; x < X; x++) {

int y = k \* x + b;

LineTo(hdc, X + x, Y - y);

}

Функция синуса

График синуса задается уравнением y = sin(x). Для его рисования используется следующая логика:

MoveToEx(hdc, 0, Y - (int)(sin(-X) \* scale), NULL);

for (int x = -X + 1; x < X; x++) {

int y = sin(x \* step) \* scale;

LineTo(hdc, X + x, Y - y);

}  
\* scale используется для масштабирования графика.

Парабола

График параболы задается уравнением y = x^2. В программе для рисования используется:

MoveToEx(hdc, 0, Y - (int)(pow(-X / scale, 2) \* scale), NULL);

for (int x = -X + 1; x < X; x++) {

int y = pow(x / scale, 2) \* scale;

LineTo(hdc, X + x, Y - y);

}

Кубическая парабола

График кубической параболы задается уравнением вида y = x^3. В программе для этого используется следующая логика:

MoveToEx(hdc, 0, Y - (int)(pow(-X / scale, 3) \* scale), NULL);

for (int x = -X + 1; x < X; x++) {

int y = pow(x / scale, 3) \* scale;

LineTo(hdc, X + x, Y - y);

}

## **Классическое приложение Windows С++ для построения 2D-графиков**

### **Код программы**

#include <windows.h> // Для работы с оконными функциями в Windows

#include <cmath> // Для математических функций - pow, sin

#include <wchar.h> // Добавляем заголовочный файл для функции swprintf\_s

// Прототипы функций обработки сообщений

LRESULT CALLBACK GraphSelectWindowProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam);

LRESULT CALLBACK GraphDisplayWindowProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam);

HINSTANCE g\_hInstance; // Экземпляр приложения

HWND g\_hCurrentGraphWindow = NULL; // Указатель на текущее окно

//графика

// Перечисление различных графиков

enum GraphType { LINEAR, SINUS, PARABOLA, PARABOLA2};

// Класс окна для выбора графика

class GraphSelectWindow

{

public:

// Конструктор принимает экземпляр приложения

GraphSelectWindow(HINSTANCE hInstance) : hInstance(hInstance) {}

// Метод для создания окна

void Create()

{

const wchar\_t CLASS\_NAME[] = L"Класс окна выбора графика";

WNDCLASS wc = { };

wc.lpfnWndProc = GraphSelectWindowProc; // Функция обработки

//сообщений для окна

wc.hInstance = hInstance; // Экземпляр приложения

wc.lpszClassName = CLASS\_NAME; // Имя класса

// Регистрируем класс окна

RegisterClass(&wc);

// Создаем окно

hwnd = CreateWindowEx(

0, // Отсутствие дополнительных стилей

CLASS\_NAME, // Имя класса

L"Выбор графика", // Заголовок окна

WS\_OVERLAPPEDWINDOW, // Стандартный стиль окна

50, // Координата x

50, // Координата y

400, // Ширина окна

400, // Высота окна

NULL, // Окно не имеет родительского окна

NULL, // Отсутствие меню

hInstance, // Дескриптор экземпляра приложения, которому

//принадлежит окно

NULL // Пользовательские данные отсутствуют

);

ShowWindow(hwnd, SW\_SHOW); // Показываем окно

}

// Метод для получения дескриптора окна

HWND GetHandle() const { return hwnd; }

private:

HINSTANCE hInstance; // Экземпляр приложения, необходимый для

//создания окна

HWND hwnd; // Дескриптор окна - используется для идентификации

//созданного окна

};

// Класс окна для отображения графика

class GraphDisplayWindow

{

public:

// Конструктор принимает экземпляр приложения, имя графика и тип

//графика

GraphDisplayWindow(HINSTANCE hInstance, LPCWSTR graphName, GraphType type)

: hInstance(hInstance), graphName(graphName), graphType(type) {}

// Метод для создания окна

void Create()

{

const wchar\_t CLASS\_NAME[] = L"Класс окна отображения графика";

WNDCLASS wc = { };

wc.lpfnWndProc = GraphDisplayWindowProc; // Функция обработки   
//сообщений для этого окна

wc.hInstance = hInstance; // Экземпляр приложения

wc.lpszClassName = CLASS\_NAME; // Имя класса

// Регистрируем класс окна

RegisterClass(&wc);

// Создаем окно

hwnd = CreateWindowEx(

0, // Отсутствие дополнительных стилей

CLASS\_NAME, // Имя класса

graphName, // Заголовок окна - меняется в зависимости от графика

WS\_OVERLAPPEDWINDOW, // Стандартный стиль окна

500, // Координата x

50, // Координата y

800, // Ширина окна

600, // Высота окна

NULL, // Окно не имеет родительского окна

NULL, // Отсутствие меню

hInstance, // Дескриптор экземпляра приложения, которому   
//принадлежит окно

this // Передаем указатель на объект окна

);

ShowWindow(hwnd, SW\_SHOW);// Показываем окно

}

// Метод для получения дескриптора окна

HWND GetHandle() const { return hwnd; }

// Возвращает тип графика, который был передан при создании объекта

GraphType GetType() const { return graphType; }

private:

HINSTANCE hInstance; // Экземпляр приложения

LPCWSTR graphName; // Имя графика

HWND hwnd; // Дескриптор окна

GraphType graphType; // Тип графика

};

// Главная функция приложения - точка входа для приложения Windows

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

{

// Сохраняем экземпляр приложения в глобальной переменной

g\_hInstance = hInstance;

// Создаем окно для выбора графика

GraphSelectWindow graphSelectWindow(hInstance);

graphSelectWindow.Create(); //Вызов метода Create() объекта //graphSelectWindow, который создает и отображает окно для выбора графика

// Запуск цикла сообщений

MSG msg = { }; // Переменная для хранения сообщений

while (GetMessage(&msg, NULL, 0, 0)) //Пока не будет получено сообщение //о завершении приложения

DispatchMessage(&msg); // Функция отправляет сообщение

//соответствующей оконной процедуре для дальнейшей обработки

return 0;

}

// Обработчик сообщений окна выбора графика

LRESULT CALLBACK GraphSelectWindowProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam)

{

switch (uMsg)

{

// Кейс обрабатывает сообщение о создании окна

case WM\_CREATE:

{

// Создание кнопок выбора графика

CreateWindow(L"BUTTON", L"График линейной функции", WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON,

50, 50, 250, 50, hwnd, (HMENU)1, NULL, NULL);

CreateWindow(L"BUTTON", L"График синуса", WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON,

50, 120, 250, 50, hwnd, (HMENU)2, NULL, NULL);

CreateWindow(L"BUTTON", L"График квадратичной функции", WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON,

50, 190, 250, 50, hwnd, (HMENU)3, NULL, NULL);

CreateWindow(L"BUTTON", L"График степенной функции", WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON,

50, 260, 250, 50, hwnd, (HMENU)4, NULL, NULL);

}

break;

// Кейс обрабатывает сообщения о командах, такие как нажатие на кнопку в //окне

case WM\_COMMAND:

switch (LOWORD(wParam)) // Для определения кнопки, которую нажали

{

case 1: // Выбор прямой

{

if (g\_hCurrentGraphWindow != NULL) // Проверка на существования //окна графика

{

DestroyWindow(g\_hCurrentGraphWindow); // Уничтожение окна

g\_hCurrentGraphWindow = NULL;

}

// Создаем и отображаем окно графика линейной функции

GraphDisplayWindow\* graphDisplayWindow = new GraphDisplayWindow(g\_hInstance, L"График линейной функции - прямая", LINEAR); // Создание нового экземпляра класса

graphDisplayWindow->Create(); // Создание и отображение

g\_hCurrentGraphWindow = graphDisplayWindow->GetHandle();

// Сохранения для дальнейшего управления

}

break;

case 2: // Выбор синусоиды

{

if (g\_hCurrentGraphWindow != NULL)

{

DestroyWindow(g\_hCurrentGraphWindow);

g\_hCurrentGraphWindow = NULL;

}

// Создаем и отображаем окно графика синуса

GraphDisplayWindow\* graphDisplayWindow = new GraphDisplayWindow(g\_hInstance, L"График синуса - синусоида", SINUS);

graphDisplayWindow->Create();

g\_hCurrentGraphWindow = graphDisplayWindow->GetHandle();

}

break;

case 3: // Выбор параболы

{

if (g\_hCurrentGraphWindow != NULL) {

DestroyWindow(g\_hCurrentGraphWindow);

g\_hCurrentGraphWindow = NULL;

}

// Создаем и отображаем окно графика параболы

GraphDisplayWindow\* graphDisplayWindow = new GraphDisplayWindow(g\_hInstance, L"График квадратичной функции - квадратная парабола", PARABOLA);

graphDisplayWindow->Create();

g\_hCurrentGraphWindow = graphDisplayWindow->GetHandle();

}

break;

case 4: // Выбор кубической параболы

{

if (g\_hCurrentGraphWindow != NULL) {

DestroyWindow(g\_hCurrentGraphWindow);

g\_hCurrentGraphWindow = NULL;

}

// Создаем и отображаем окно графика кубической параболы

GraphDisplayWindow\* graphDisplayWindow = new GraphDisplayWindow(g\_hInstance, L"График степенной функции - кубическая парабола", PARABOLA2);

graphDisplayWindow->Create();

g\_hCurrentGraphWindow = graphDisplayWindow->GetHandle();

}

break;

}

break;

// Кейс обрабатывает сообщение о закрытии окна

case WM\_DESTROY:

// Завершение приложения

PostQuitMessage(0);

return 0;

/\*Если обработчик не распознает сообщение или команду, оно передается

функции DefWindowProc, которая выполняет стандартную обработку сообщения\*/

default:

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

}

return 0;

}

void line(HDC hdc, int Xs, int Ys, int Xf, int Yf) // Рисование отрезка прямой //линии

{

if (hdc != NULL)

{

MoveToEx(hdc, Xs, Ys, 0); // Перемещение из точки (Xs, Ys)

LineTo(hdc, Xf, Yf); // В точку (Xt, Yf)

}

}

// Обработчик сообщений окна отображения графика

LRESULT CALLBACK GraphDisplayWindowProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam)

{

if (uMsg == WM\_CREATE) // Было ли сообщение о создании окна

{

// Сохраняем указатель на объект окна

LPCREATESTRUCT pCreate = (LPCREATESTRUCT)lParam; // Получаем //доп. данные, которые содержат информацию о создании окна

SetWindowLongPtr(hwnd, GWLP\_USERDATA, (LONG\_PTR)pCreate->lpCreateParams); // Сохраняем указатель на объект окна

return 0;

}

// Получаем указатель на объект окна

GraphDisplayWindow\* pGraphWindow = (GraphDisplayWindow\*)GetWindowLongPtr(hwnd, GWLP\_USERDATA);

if (pGraphWindow) // Существует ли указатель на объект окна

{

switch (uMsg)

{

case WM\_DESTROY:

// Не выходим из программы при закрытии окна графика

return 0;

// Кейс активируется, когда окно требует перерисовки

case WM\_PAINT:

{

/\*Структура в Windows API, которая используется для хранения

//информации о процессе рисования окна в функции BeginPaint и EndPaint\*/

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hwnd, &ps); // Функция начала рисования

// hwnd - дескриптор окна, в котором будет происходить рисование

// Получаем размеры клиентской области окна

// ps - указатель на структуру PAINTSTRUCT, в которую будет записана //информация о процессе рисования

RECT rect;

GetClientRect(hwnd, &rect); // Функция Windows API, которая получает //размеры клиентской области указанного окна и записывает их в структуру //RECT

int width = rect.right - rect.left; //длина клиентской области

int height = rect.bottom - rect.top; //высота

// Координаты центра клиентской области

int X = width / 2;

int Y = height / 2;

//Полная отрисовка оси х

line(hdc, 10, Y, width - 10, Y); //рисование горизонтальной оси

line(hdc, width - 10, Y, width - 15, Y - 5); //верхнее гориз-ое оперение

line(hdc, width - 10, Y, width - 15, Y + 5); //нижнее гориз-ое оперение

//Полная отрисовка оси у

line(hdc, X, 10, X, height - 10); //рисование вертикальной оси

line(hdc, X, 10, X + 5, 15); //правое верт-ое оперение

line(hdc, X, 10, X - 5, 15); //левое верт-ое оперение

TCHAR buffer[32]; // Для хранения строковых данных

TextOut(hdc, width - 10, Y + 10, L"X", 1); // Подпись оси x

TextOut(hdc, X + 10, 5, L"Y", 1); // Подпись оси y

TextOut(hdc, X + 5, Y + 5, L"0", 1); // 0 на пересечении осей

// Определяем количество делений на оси X и Y

int numDivisionsX = width / 50; // 50 - расстояние между делениями

int numDivisionsY = height / 50;

// Рисуем деления на оси X

for (int i = 1; i < numDivisionsX; i++) {

// Для положительных значений X

int xCoordPositive = X + i \* (width / numDivisionsX);

// Для отрицательных значений X

int xCoordNegative = X - i \* (width / numDivisionsX);

line(hdc, xCoordPositive, Y - 5, xCoordPositive, Y + 5);

line(hdc, xCoordNegative, Y - 5, xCoordNegative, Y + 5);

// Подписываем деление на оси X

TCHAR buffer[32];

swprintf\_s(buffer, L"%d", i \* 50);

TextOut(hdc, xCoordPositive - 10, Y + 10, buffer, wcslen(buffer));

swprintf\_s(buffer, L"-%d", i \* 50);

TextOut(hdc, xCoordNegative - 15, Y + 10, buffer, wcslen(buffer));

}

// Рисуем деления на оси Y

for (int i = 1; i < numDivisionsY; i++) {

// Для положительных значений Y

int yCoordPositive = Y - i \* (height / numDivisionsY);

// Для отрицательных значений Y

int yCoordNegative = Y + i \* (height / numDivisionsY);

line(hdc, X - 5, yCoordPositive, X + 5, yCoordPositive);

line(hdc, X - 5, yCoordNegative, X + 5, yCoordNegative);

// Подписываем деление на оси Y

TCHAR buffer[32];

swprintf\_s(buffer, L"%d", i \* 100);

TextOut(hdc, X + 10, yCoordPositive - 10, buffer, wcslen(buffer));

swprintf\_s(buffer, L"-%d", i \* 100);

TextOut(hdc, X + 10, yCoordNegative - 10, buffer, wcslen(buffer));

}

// Переменные для управления точностью и масштабом при отрисовке

double step = 0.1; // Определяет шаг, с которым будут вычисляться точки //графика функции

double scale = 50.0; // Определяет масштаб, с которым значения

//функции будут преобразованы в пиксели на экране

if (pGraphWindow->GetType() == LINEAR) // Проверка типа графика

{

// Установка цвета пера для отрисовки графика

HPEN Pen = CreatePen(PS\_SOLID, 2, RGB(255, 0, 0)); // Красный цвет

SelectObject(hdc, Pen);

// Отрисовка графика линейной функции

// Идем из крайней левой точки окна в правую

MoveToEx(hdc, 0, Y - (int)(0.6 \* (-X) + 3), NULL); // Начальная точка //- крайнее левое положение окна по оси X и соответствующее значение Y

for (int x = - X + 1; x < X; x++) {

int y = 0.6 \* x + 3; // Вычисление значения линейной функции

LineTo(hdc, X + x, Y - y);

}

DeleteObject(Pen); // Освобождение ресурсов пера

}

else if (pGraphWindow->GetType() == SINUS) {

HPEN Pen = CreatePen(PS\_SOLID, 2, RGB(0, 255, 0)); // Зеленый цвет

SelectObject(hdc, Pen);

scale = 70.0;

// Отрисовка графика синуса

MoveToEx(hdc, 0, Y - (int)(sin(-X) \* scale), NULL);

for (int x = -X + 1; x < X; x++) {

int y = sin(x \* step) \* scale; // Значение функции синуса

LineTo(hdc, X + x, Y - y);

}

DeleteObject(Pen);

}

else if (pGraphWindow->GetType() == PARABOLA) {

HPEN Pen = CreatePen(PS\_SOLID, 2, RGB(0, 0, 255)); // Синий цвет

SelectObject(hdc, Pen);

// Отрисовка графика квадратичной функции

MoveToEx(hdc, 0, Y - (int)(pow(-X / scale, 2) \* scale), NULL);

for (int x = -X + 1; x < X; x++) {

int y = pow(x / scale, 2) \* scale; // Значение функции параболы

LineTo(hdc, X + x, Y - y);

}

DeleteObject(Pen);

}

else if (pGraphWindow->GetType() == PARABOLA2) {

HPEN Pen = CreatePen(PS\_SOLID, 2, RGB(255, 0, 255)); // Розовый //цвет

SelectObject(hdc, Pen);

// Отрисовка графика степенной функции

MoveToEx(hdc, 0, Y - (int)(pow(-X / scale, 3) \* scale), NULL);

for (int x = -X + 1; x < X; x++) {

int y = pow(x / scale, 3) \* scale; // Значение функции кубической //параболы

LineTo(hdc, X + x, Y - y);

}

DeleteObject(Pen);

}

EndPaint(hwnd, &ps); // Завершение процесса рисования

}

return 0;

default:

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

}

}

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

}

# **Результат работы программы**

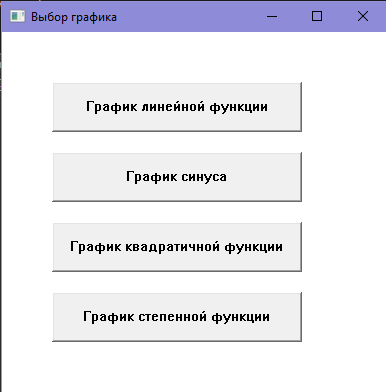
****

Рисунок 1. Окно выбора графика

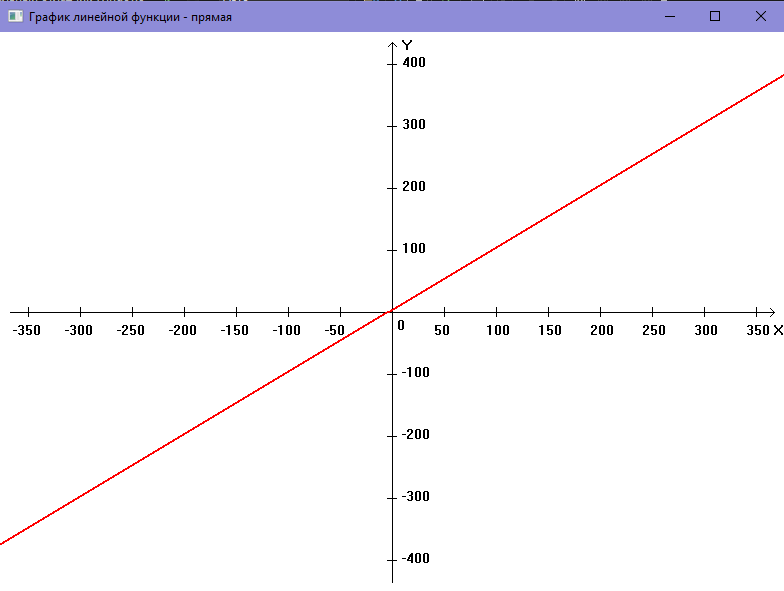


Рисунок 2. Окно с графиком линейной функции (высветилось после нажатия на соответствующую кнопку)

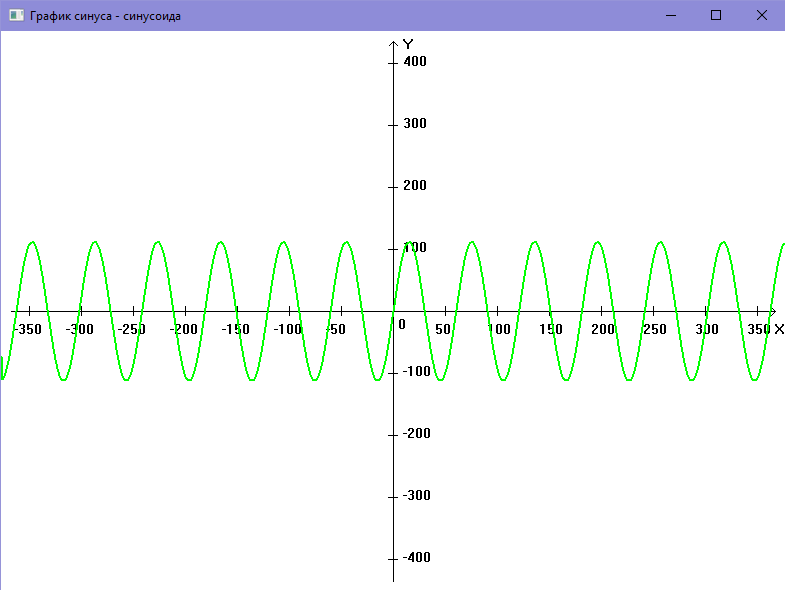


Рисунок 3. Окно с графиком синуса (высветилось после нажатия на соответствующую кнопку)

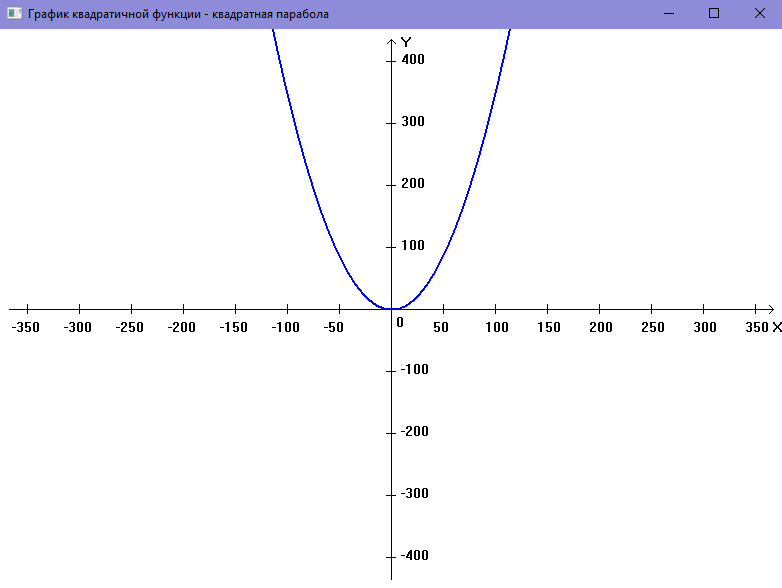


Рисунок 4. Окно с графиком квадратичной функции (высветилось после нажатия на соответствующую кнопку)

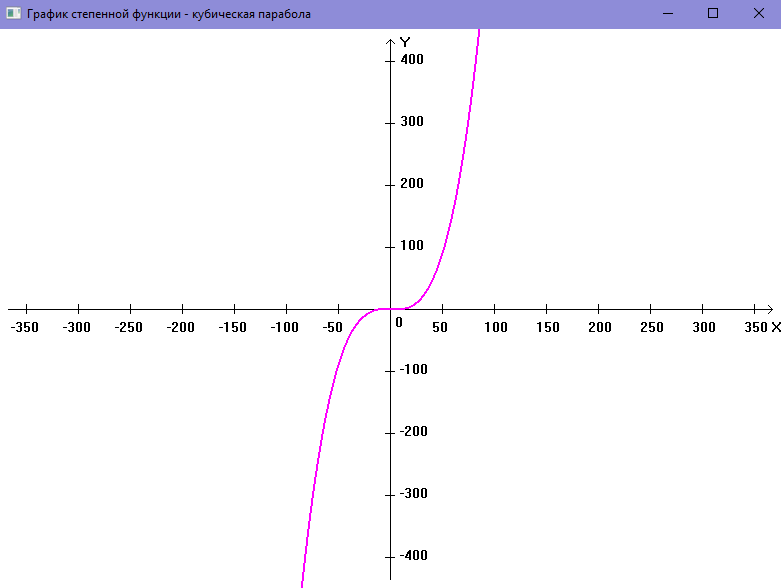


Рисунок 5. Окно с графиком степенной функции (высветилось после нажатия на соответствующую кнопку)

# **Вывод**

В ходе выполнения курсовой работы были реализованы алгоритмы отображения графиков различных функций. Было создано окно, предназначенное для выбора интересующего графика, а также окно для отображения выбранного графика.

Программы позволяет пользоваться удобным меню графиков, в котором от пользователя требуется нажать на кнопку, на которой написано соответствующее название графика.

В дальнейшем можно без труда добавить в программу новые графики.

В общем, классическое приложение Windows предоставляет возможность написания программы, которая реализует графические функции, а также позволяет использовать функции для создания и управления несколькими графическими окнами с собственными характеристиками.

# **Список источников**

1. Щупак Ю.А. Win32 API. Разработка приложений для Windows. – Спб.: Питер, 2008. – 592с.
2. Определение классического приложения Windows

https://learn.microsoft.com/ru-RU/cpp/windows/overview-of-windows-programming-in-cpp?view=msvc-170&viewFallbackFrom=vs-2019