**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ  
 РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФГБОУ ВО «БРЯНСКИЙ ГОСУДАРСТВЕННЫЙ   
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**Кафедра** «Информатика и программное обеспечение»

**КУРСОВАЯ РАБОТА ПО ДИСЦИПЛИНЕ**

**«Программирование приложений Windows»**

«Создание демонстрационно-обучающей программы по методам интерполяции функций (многочлены Ньютона и Лагранжа)»

Выполнил студ. гр. 15-ИВТ2

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Шалом В.Ю.

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2018 г.

Руководитель

\_\_\_\_\_\_\_\_\_\_\_к.т.н., доц. Белов Е.А.

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2018 г.

Брянск 2018**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ 3

ТЕХНИЧЕСКОЕ ЗАДАНИЕ 4

1. АНАЛИТИЧЕСКАЯ ЧАСТЬ 5

1.1. Анализ предметной области 5

1.2. Современные аналоги 5

1.3. Выбор языка и платформы 5

2. КОНСТРУКТОРСКАЯ ЧАСТЬ 7

2.1. Программирование приложений Windows 7

2.2. Проектирование 8

3. ЭКСПЕРИМЕНТАЛЬНАЯ ЧАСТЬ 16

3.1. Руководство пользователя 16

3.2. Тестирование 18

ЗАКЛЮЧЕНИЕ 19

СПИСОК ЛИТЕРАТУРЫ 20

ПРИЛОЖЕНИЕ 1 21

ВВЕДЕНИЕ

Педагоги всегда уделяли большое внимание проблеме активизации деятельности учащихся в процессе обучения, поскольку сознательное и прочное усвоение знаний и осознанное формирование способов действий происходит, прежде всего, в процессе активной умственной деятельности обучаемых. В последнее время всё большее внимание уделяется интерактивным методам обучения. Понятие «интеракции» трактуется как способность взаимодействия с чем-либо (компьютер), или кем-либо (человек).

В рамках данного курсового проекта будет разработан программный продукт, который позволит в интерактивной форме изучить методы интерполяции математических функций с помощью интерполяционных полиномов Ньютона и Лагранжа.

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

Цель курсовой работы по дисциплине «Программирование приложений Windows» состоит в закреплении и углублении знаний и навыков, полученных в ходе изучения курса. Курсовая работа представляет собой индивидуальное задание повышенной сложности на изучаемом языке высокого уровня. Результатом разработки должна стать демонстрационно-обучающая программа по методам аппроксимации функций.

**Требования к функциональным характеристикам:**

* Программа должна иметь удобный и интуитивно понятный интерфейс.
* Наличие необходимого теоретического минимума, рассматривающего интерполяционные многочлены Ньютона и Лагранжа.
* Возможность вычислить значение заданного полинома в конкретной точке.
* Раздел с заданиями для самостоятельной работы и возможностью самопроверки.

# 1. АНАЛИТИЧЕСКАЯ ЧАСТЬ

## 1.1. Анализ предметной области

Аспекты интерактивного обучения на сегодняшний день хорошо изучены, однако компьютеризация этого процесса все ещё находится в процессе становления. На сегодняшний день существует множество сервисов, позволяющих производить математические вычисления различной сложности и направленности. В то же время, лишь единицы дают отдельные теоретические знания и помогают в вопросе.

Так же стоит отметить, что данная область широко распространена в веб-среде в ущерб настольным приложениям. Поэтому актуальность данной курсовой работы оправдана.

## 1.2. Современные аналоги

Единственным крупным, на мой взгляд, сервисом интерактивного обучения на сегодняшний день выступает следа дистанционного обучения Moodle. Moodle на равных соперничает с мировыми флагманами рынка СДО. Над системой уже более 10 лет работает международная команда разработчиков, под руководством фонда Moodle в Австралии. Благодаря этому, Moodle сочетает в себе богатство функционала, гибкость, надежность и простоту использования. Система широко известна в мире, имеет более 60 тысяч инсталляций более чем в 100 странах, переведена на несколько десятков языков. Система хорошо масштабируется: существуют инсталляции, обслуживающие до миллиона пользователей.  
СДО Moodle предназначена для создания и проведения качественных дистанционных курсов.

## 1.3. Выбор языка и платформы

В качестве языка программирования для реализации данного курсового проекта был выбран С++. Язык С++ — универсальный язык общего назначения, область приложений которого — программирование систем в самом широком смысле. Кроме этого, С++ успешно используется как во многих приложениях, так и в мощных операционных системах.

Концепция операционной системы Windows состоит в том, что программа взаимодействует с пользователем в интерактивном режиме. Это реализуется с помощью графических интерфейсов. Windows имеет набор стандартно выглядящих элементов управления и отлаженный механизм работы с ними. Но часто возникает нужда добавить необходимый несуществующий элемент управления или просто изменить внешний вид элемента. Многие программы такие как проигрыватели мультимедиа еще с времен Windows 95 используют свой графический интерфейс пользователя для придания большей привлекательности со стороны пользователей. Каждый разработчик пытается вложить в программу что-то свое, необычное, не похожее на остальных. Конечно, функциональность играет основную роль, но удобный и красивый интерфейс не менее важен и проблема это довольно серьезная.

Для реализации поставленной задачи мною была выбрана среда разработки Microsoft Visual Studio 2015, так как она наиболее подходит для создания данного проекта.

Visual Studio имеет понятный интерфейс, большой набор функций и инструментов.

# 2. КОНСТРУКТОРСКАЯ ЧАСТЬ

## 2.1. Программирование приложений Windows

### ***2.1.1. Функция окна***

Все Windows-программы должны содержать специальную функцию, которая не используется в самой программе, но вызывается самой операционной системой. Эту функцию обычно называют функцией окна, или процедурой окна. Она вызывается Windows, когда системе необходимо передать сообщение в программу.

Именно через нее осуществляется взаимодействие между программой и системой. Функция окна передает сообщение в своих аргументах. Согласно терминологии Windows, функции, вызываемые системой, называются функциями обратного вызова. Таким образом, функция окна является функцией обратного вызова.

Помимо принятия сообщения от Windows, функция окна должна вызывать выполнение действия, указанного в сообщении.

Конечно, программа не обязана отвечать на все сообщения, посылаемые Windows. Поскольку их могут быть сотни, то большинство сообщений обычно обрабатывается самой системой, а программе достаточно поручить выполнять действия, предусмотренные по умолчанию.

В большинстве Windows-программ задача создания функции окна ложится на плечи программиста. Но в любом случае, если сообщение получено, то программа должна выполнить некоторое действие. Хотя она может вызывать для этого одну или несколько API-функций, само действие было инициировано Windows. Поэтому именно способ взаимодействия с операционной системой через сообщения диктует общий принцип построения всех программ для Windows.

### 2.1.2. Цикл сообщений

Каждое окно в Windows-приложении характеризуется определенными атрибутами, называемыми классом окна. (Здесь понятие “класс” не идентично используемому в C++. Оно, скорее, означает стиль или тип.) В традиционной программе класс окна должен быть определен и зарегистрирован прежде, чем будет создано окно. При регистрации необходимо сообщить Windows, какой вид должно иметь окно и какую функцию оно выполняет. В то же время регистрация класса окна еще не означает создание самого окна. Для этого требуется выполнить дополнительные действия.

### 2.1.3. Типы данных в Windows

В Windows-программах не слишком широко применяются стандартные типы данных из С или С++, такие как int или char\*. Вместо них используются типы данных, определенные в различных библиотечных (header) файлах.

Наиболее часто используемыми типами являются HANDLE, HWND, BYTE, WORD, DWORD, UNIT, LONG, BOOL, LPSTR и LPCSTR. Тип HANDLE обозначает 32-разрядное целое, используемое в качестве дескриптора. Есть несколько похожих типов данных, но все они имеют ту же длину, что и HANDLE, и начинаются с литеры Н.

Дескриптор – это просто число, определяющее некоторый ресурс. Например, тип HWND обозначает 32-разрядное целое – дескриптор окна. Тип BYTE обозначает 8-разрядное беззнаковое символьное значение, тип WORD – 16-разрядное беззнаковое короткое целое, тип DWORD – беззнаковое длинное целое, тип UNIT - беззнаковое 32-разрядное целое. Тип LONG эквивалентен типу long. Тип BOOL обозначает целое и используется, когда значение может быть либо истинным, либо ложным. Тип LPSTR определяет указатель на строку, а LPCSTR – константный (const) указатель на строку.

## 2.2. Проектирование

### 2.2.1. Общий интерфейс программы

Интерфейс программы проектировался с учетом установленных ранее требований и представляет собой набор окно с элементами управления, создающимися в соответствии с текущим сценарием работы.

### 2.2.2. Основные функции приложения

Основные функции, описанные в данной программе, отвечают за инициализацию окна и рисование элементов управления для создания тематических экранов.

Таблица 1 – «Обработчики окон»

|  |  |  |
| --- | --- | --- |
| **№** | **Название метода** | **Описание метода** |
| 1 | WndPrc(HWND,UNIT,WPARAM,LPARAM) | Обработчик главного окна приложения. |
| 2 | WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int) | Главная функция программы. |

Таблица 2 – «Функции»

|  |  |  |
| --- | --- | --- |
| **№** | **Название метода** | **Описание метода** |
| 1 | drawHome(HDC hDC, HWND hWnd) | Отрисовка главного окна. |
| 2 | drawInfo(HDC hDC, HWND hWnd, int method) | Отрисовка окна с теоретическим минимумом. |
| 3 | drawGraph(HDC hDC, HWND hWnd, int method) | Отрисовка окна вычисления значения полинома. |
| 4 | drawExample(HDC hDC, HWND hWnd, int method) | Отрисовка окна с примером. |
| 5 | drawExercises(HDC hDC, HWND hWnd, int method) | Отрисовка окна с заданиями. |
| 6 | reset() | Функция удаления элементов управления и задания необходимых значений по-умолчанию. |
| 7 | resetTable() | Функция освобождения памяти под массив дескрипторов полей ввода для таблицы значений. |

### 2.2.3. Окна приложения

Приложение состоит из главного окна, элементы управления на котором создаются в зависимости от текущего сценария работы, выбранного пользователем. Будем называть такие наборы элементов управления страницами.

На домашней странице расположен текст приветствия и краткая информация об интерполяции. Ниже размещены два элемента управления BUTTON, соответствующие рассматриваемым методам.

Создание главного окна:

Листинг 2.1. Создание главного окна

hWnd = CreateWindow (ClassName, "Интерполяционные многочлены", WS\_OVERLAPPEDWINDOW, CW\_USEDEFAULT, CW\_USEDEFAULT, 1000, 800, 0, hMenu, hInst, NULL);

Домашняя страница отрисовывается с помощью функции drawHome(). Текст на экран выводится двумя способами. Для однострочных надписей используется функция TextOut(), в то время как многострочный текст рисуется при помощи DrawText с соответствующим параметром.

Листинг 2.2. Вывод текста

string Buf = "Добро пожаловать в обучающую программу";

TextOut(hDC, 100, 60, Buf.c\_str(), strlen(Buf.c\_str()));

SelectObject(hDC, font25);

Buf = "Тема: Методы аппроксимации функций. Многочлены Ньютона и Лагранжа";

TextOut(hDC, 120, 120, Buf.c\_str(), strlen(Buf.c\_str()));

………………………………………………..

r.top = 180;

r.left = 200;

r.bottom = 260;

r.right = 800;

SelectObject(hDC, font20);

Buf = " Интерполяция, интерполирование (от лат. inter–polis — \"разглаженный, подновлённый, обновлённый; преобразованный\") — в вычислительной математике способ нахождения промежуточных значений величины по имеющемуся дискретному набору известных значений.";

DrawText(hDC, Buf.c\_str(), Buf.length(), &r, DT\_WORDBREAK);

Две кнопки BUTTON используются для перехода к соответствующим разделам, рассматривающим изучаемые интерполяционные многочлены. Создание этих кнопок представлено в листинге 2.3.

Листинг 2.3. Создание окон EDIT

hBtnLagr = CreateWindow("BUTTON", "Интерполяционный полином Лагранжа", WS\_CHILD | WS\_VISIBLE | BS\_DEFPUSHBUTTON,

200, 420, 280, 120,

hMainWin, (HMENU)ID\_BUTTON\_LAGRANZ, hInst, NULL);

hBtnNewt = CreateWindow("BUTTON", "Интерполяционный полином Ньютона", WS\_CHILD | WS\_VISIBLE | BS\_DEFPUSHBUTTON,

520, 420, 280, 120,

hMainWin, (HMENU)ID\_BUTTON\_NEWTON, hInst, NULL);

Три кнопки BUTTON используются для открытия диалогового окна просмотра сегодняшней новости, для открытия диалогового окна поиска новостей по дате и для выхода из программы.

Листинг 2.4. Создание BUTTON

HWND hBtn = CreateWindow(

"BUTTON", "События на сегодня",

WS\_TABSTOP | WS\_VISIBLE | WS\_CHILD | BS\_DEFPUSHBUTTON,

550, 40, 140, 50, hWnd, (HMENU)10997, hInst, NULL);

HWND hBtn2 = CreateWindow(

"BUTTON", "События по дате",

WS\_TABSTOP | WS\_VISIBLE | WS\_CHILD | BS\_DEFPUSHBUTTON,

550, 100, 140, 50, hWnd, (HMENU)10998, hInst, NULL);

HWND hBtn3 = CreateWindow(

"BUTTON", "Выход",

WS\_TABSTOP | WS\_VISIBLE | WS\_CHILD | BS\_DEFPUSHBUTTON,

550, 470, 140, 50, hWnd, (HMENU)10999, hInst, NULL);

### 2.2.4. Информационные страницы

Функции программы, которые отрисовывают остальные страницы, универсальны для каждого из двух рассматриваемых метода интерполяции. Обосновать это можно тем, что большинство элементов управления имеют одинаковое расположение и содержание. В тех же случаях, когда необходимо точно указать отдельные параметры, используется условный блок, который, в зависимости от переданного специального параметра, определяет содержимое отдельных элементов или используемый алгоритм. Рассмотрим несколько примером данных функций.

В листинге 2.4 представлен код функции drawInfo(), отвечающей за отображение страницы с теоретическим минимумом для выбранного раздела. Как видим, сначала создаются кнопки, которые служат навигационными элементами управления и являются идентичными для всех страниц. После этого происходит обработка ветвления и, в зависимости от рассматриваемого метода, да экран выводится изображение с информацией, соответствующей изучаемому полиному. Такой подход позволяет сократить код и повышает его расширяемость.

*Листинг 2.5.* Функции drawInfo()

BOOL drawInfo(HDC hDC, HWND hWnd, int method)

{

hGraphBtn = CreateWindow("BUTTON", "Вычислить", WS\_CHILD | WS\_VISIBLE | BS\_DEFPUSHBUTTON,

20, 40, 160, 40,

hMainWin, (HMENU)ID\_BUTTON\_GRAPH, hInst, NULL);

hExampBtn = CreateWindow("BUTTON", "Пример", WS\_CHILD | WS\_VISIBLE | BS\_DEFPUSHBUTTON,

20, 90, 160, 40,

hMainWin, (HMENU)ID\_BUTTON\_EXAMP, hInst, NULL);

hExercBtn = CreateWindow("BUTTON", "Упражнения", WS\_CHILD | WS\_VISIBLE | BS\_DEFPUSHBUTTON,

20, 140, 160, 40,

hMainWin, (HMENU)ID\_BUTTON\_EXERC, hInst, NULL);

hBackBtn = CreateWindow("BUTTON", "Назад", WS\_CHILD | WS\_VISIBLE | BS\_DEFPUSHBUTTON,

20, 680, 160, 40,

hMainWin, (HMENU)ID\_BUTTON\_BACK, hInst, NULL);

if (method == LAGR)

{

SelectObject(hDC, fontBig);

string Buf = "Интерполяционный полином Лагранжа";

TextOut(hDC, 210, 40, Buf.c\_str(), strlen(Buf.c\_str()));

BITMAP bitmap;

HDC hdcMem;

HGDIOBJ oldBitmap;

hdcMem = CreateCompatibleDC(hDC);

oldBitmap = SelectObject(hdcMem, hBitmapLagrInfo);

GetObject(hBitmapLagrInfo, sizeof(bitmap), &bitmap);

BitBlt(hDC, 210, 150, bitmap.bmWidth, bitmap.bmHeight, hdcMem, 0, 0, SRCCOPY);

SelectObject(hdcMem, oldBitmap);

DeleteDC(hdcMem);

}

else if (method == NEWT)

{

SelectObject(hDC, fontBig);

string Buf = "Интерполяционный полином Ньютона";

TextOut(hDC, 210, 40, Buf.c\_str(), strlen(Buf.c\_str()));

BITMAP bitmap;

HDC hdcMem;

HGDIOBJ oldBitmap;

hdcMem = CreateCompatibleDC(hDC);

oldBitmap = SelectObject(hdcMem, hBitmapNewtInfo);

GetObject(hBitmapNewtInfo, sizeof(bitmap), &bitmap);

BitBlt(hDC, 210, 150, bitmap.bmWidth, bitmap.bmHeight, hdcMem, 0, 0, SRCCOPY);

SelectObject(hdcMem, oldBitmap);

DeleteDC(hdcMem);

}

return true;

}

Отдельно стоит рассмотреть функцию drawGraph, как наиболее сложную в плане создания необходимых элементов управления. Опустим часть кода, которая создаёт уже знакомые нам навигационные кнопки. Поясним значение флагов runGraph, inWork и result. Флаг inWork отвечает за отрисовку таблицы полей ввода для координат точек исследуемой функции. Он устанавливается лишь тогда, когда успешно получено количество точек в соответствующее поле ввода.

Далее, когда все координаты точек успешно введены, поднимается флаг runGraph, который сигнализирует о том, что можно создавать элементы управления для ввода координаты Х, для которой мы ходим вычислить значение интерполяционного многочлена.

В завершение переменная result принимает значение true и выводится результат вычисления значения полинома в точке. С помощью условного ветвления подбирается соответствующая функция для его вычисления.

*Листинг 2.6. Функция drawGraph*

BOOL drawGraph(HDC hDC, HWND hWnd, int method)

{

RECT r;

GetClientRect(hWnd, &r);//Получает координаты клиентской области окна.

……………………………………………………………………………………………..

hEditCount = CreateWindow("edit", "", WS\_VISIBLE | WS\_CHILD | WS\_BORDER, 320, 220, 60, 40, hMainWin, (HMENU)ID\_EDIT\_COUNT, hInst, NULL);

hCountBtn = CreateWindow("BUTTON", "ОК", WS\_CHILD | WS\_VISIBLE | BS\_DEFPUSHBUTTON,

390, 220, 60, 40,

hMainWin, (HMENU)ID\_BUTTON\_SET\_COUNT, hInst, NULL);

TextOut(hDC, 220, 230, "Кол-во точек:", strlen("Кол-во точек:"));

if (runGraph)

{

TextOut(hDC, 220, 410, "X:", strlen("X:"));

hEditCheck = CreateWindow("edit", "", WS\_VISIBLE | WS\_CHILD | WS\_BORDER, 240, 400, 60, 40, hMainWin, (HMENU)ID\_EDIT\_CHECK, hInst, NULL);

hCheckBtn = CreateWindow("BUTTON", "Вычислить", WS\_CHILD | WS\_VISIBLE | BS\_DEFPUSHBUTTON,

320, 400, 100, 40,

hMainWin, (HMENU)ID\_BUTTON\_CHECK, hInst, NULL);

}

if (inWork)

{

TextOut(hDC, 220, 290, "X:", strlen("X:"));

TextOut(hDC, 220, 330, "Y:", strlen("Y:"));

table = new HWND\*[2];

for (int i = 0; i < 2; i++)

{

table[i] = new HWND[valCount];

}

int tWidth = 40;

int tHeight = 40;

int xPos = 250;

int yPos = 280;

int tX;

for (int i = 0; i < 2; i++)

{

for (int j = 0; j < valCount; j++)

{

table[i][j] = CreateWindow("edit", "", WS\_VISIBLE | WS\_CHILD | WS\_BORDER, xPos, yPos, tWidth, tHeight, hMainWin, NULL, hInst, NULL);

xPos += tWidth;

}

tX = xPos;

xPos = 250;

yPos += tHeight;

}

runTable = true;

hDrawBtn = CreateWindow("BUTTON", "Многочлен", WS\_CHILD | WS\_VISIBLE | BS\_DEFPUSHBUTTON,

tX + 10, 300, 90, 40,

hMainWin, (HMENU)ID\_BUTTON\_DRAW, hInst, NULL);

}

if (method == LAGR)

{

SelectObject(hDC, fontBig);

string Buf = "Интерполяционный полином Лагранжа";

TextOut(hDC, 210, 40, Buf.c\_str(), strlen(Buf.c\_str()));

r.top = 120;

r.left = 220;

r.bottom = 200;

r.right = 820;

SelectObject(hDC, font20);

Buf = " Здесь вы можете узнать значение интерполяционного полинома Лагранжа в конкретной точке. Для начала введите количество точек, определяющих функцию, затем введите координаты и нажмите на кнопку \"Многочлен\". В появившееся поле введите значение Х и нажмите кнопку \"Вычислить\".";

DrawText(hDC, Buf.c\_str(), Buf.length(), &r, DT\_WORDBREAK);

if (result)

{

TextOut(hDC, 220, 500, "Полином Лагранжа в данной точке равен:", strlen("Полином Лагранжа в данной точке равен:"));

string p = to\_string(Lagranz(X, Y, pointX));

TextOut(hDC, 520, 500, p.c\_str(), strlen(p.c\_str()));

}

}

else if (method == NEWT)

{

SelectObject(hDC, fontBig);

string Buf = "Интерполяционный полином Ньютона";

TextOut(hDC, 210, 40, Buf.c\_str(), strlen(Buf.c\_str()));

r.top = 120;

r.left = 220;

r.bottom = 200;

r.right = 820;

SelectObject(hDC, font20);

Buf = " Здесь вы можете узнать значение интерполяционного полинома Ньютона в конкретной точке. Для начала введите количество точек, определяющих функцию, затем введите координаты и нажмите на кнопку \"Многочлен\". В появившееся поле введите значение Х и нажмите кнопку \"Вычислить\".";

DrawText(hDC, Buf.c\_str(), Buf.length(), &r, DT\_WORDBREAK);

if (result)

{

TextOut(hDC, 220, 500, "Полином Ньютона в данной точке равен:", strlen("Полином Ньютона в данной точке равен:"));

string p = to\_string(Newton(X, Y, pointX));

TextOut(hDC, 600, 500, p.c\_str(), strlen(p.c\_str()));

}

}

return true;

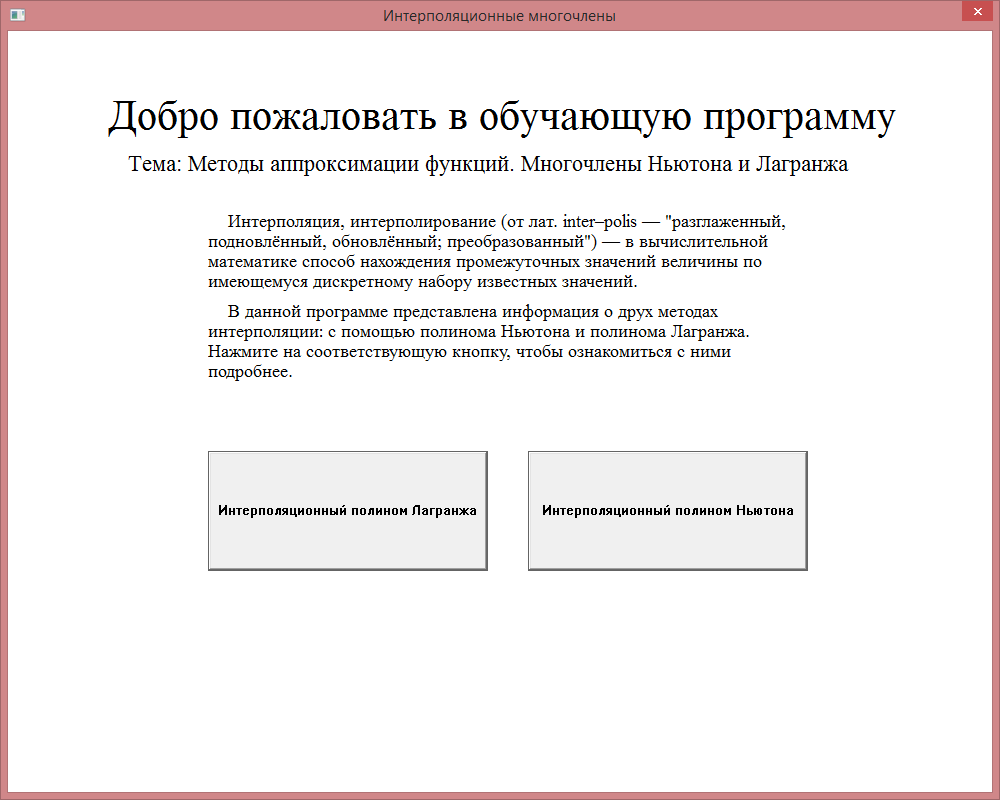
}

Алгоритм работы оставшихся функций схож с рассмотренными. Он использует те же архитектурные и логические принципы. Обработка событий нажатия кнопок представлена в приложении 1.

# 3. ЭКСПЕРИМЕНТАЛЬНАЯ ЧАСТЬ

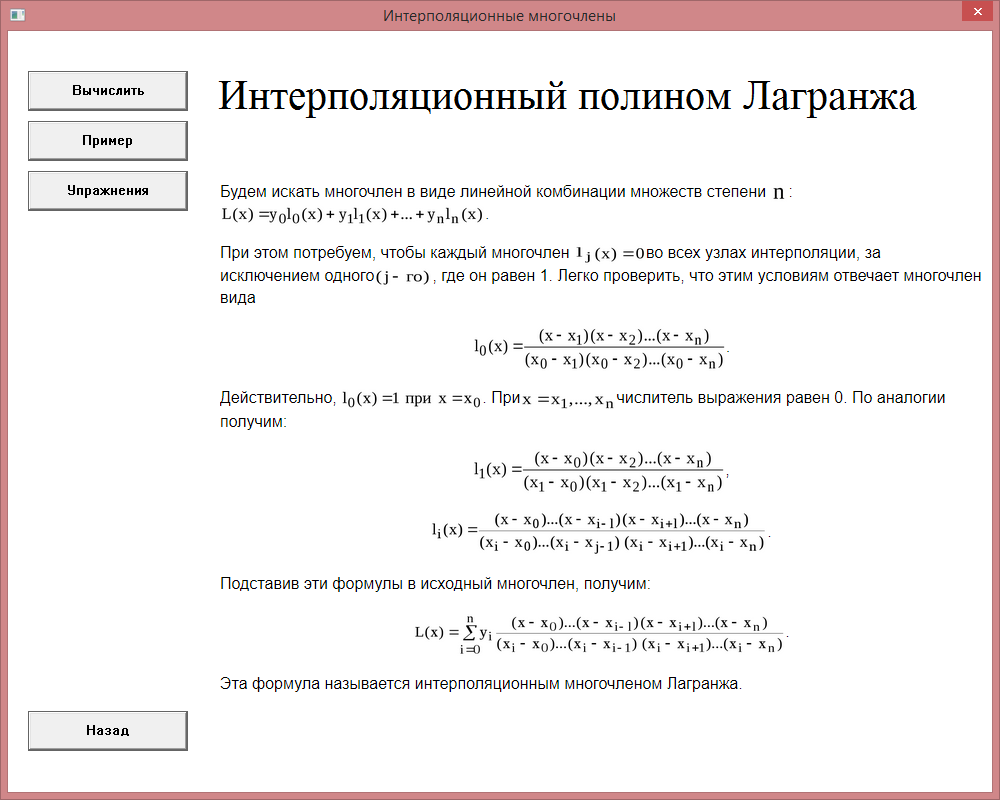
## 3.1. Руководство пользователя

После запуска программы открывается главное окно приложения (рис. 3.1.), на котором представлена домашняя страница.



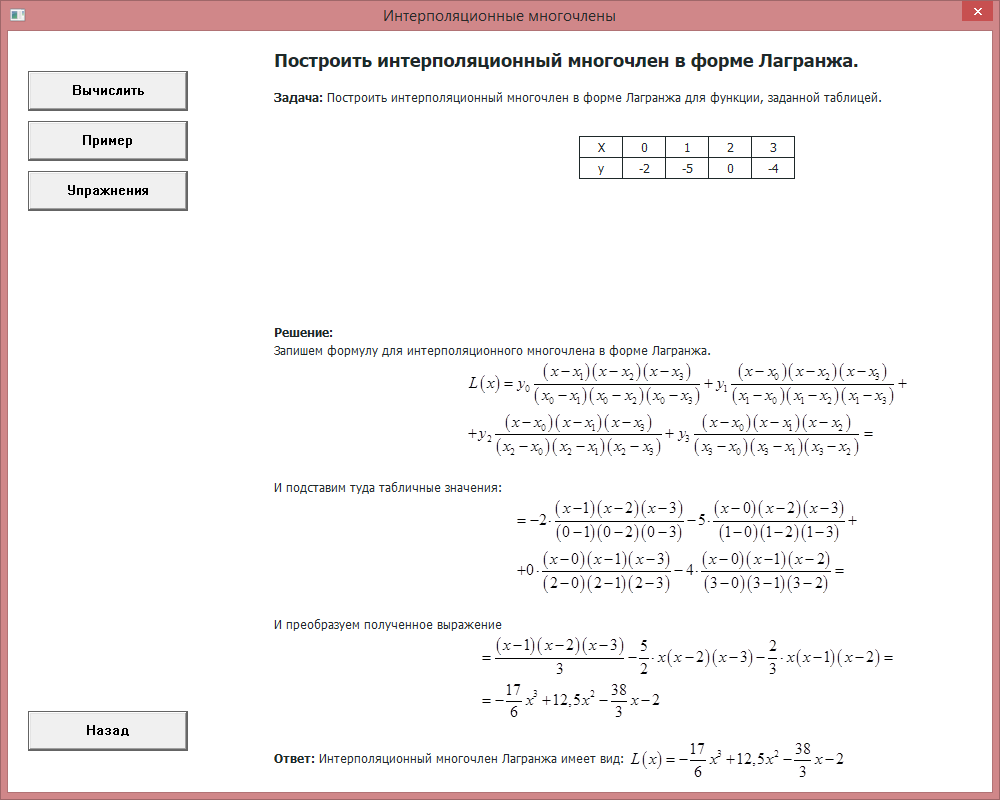
**Рис. 3.1. Главное окно**

Нажатие на одну из кнопок приведёт к переходу к обучающему разделу (рис. 3.3.), в котором рассматривается выбранный интерполяционный многочлен.



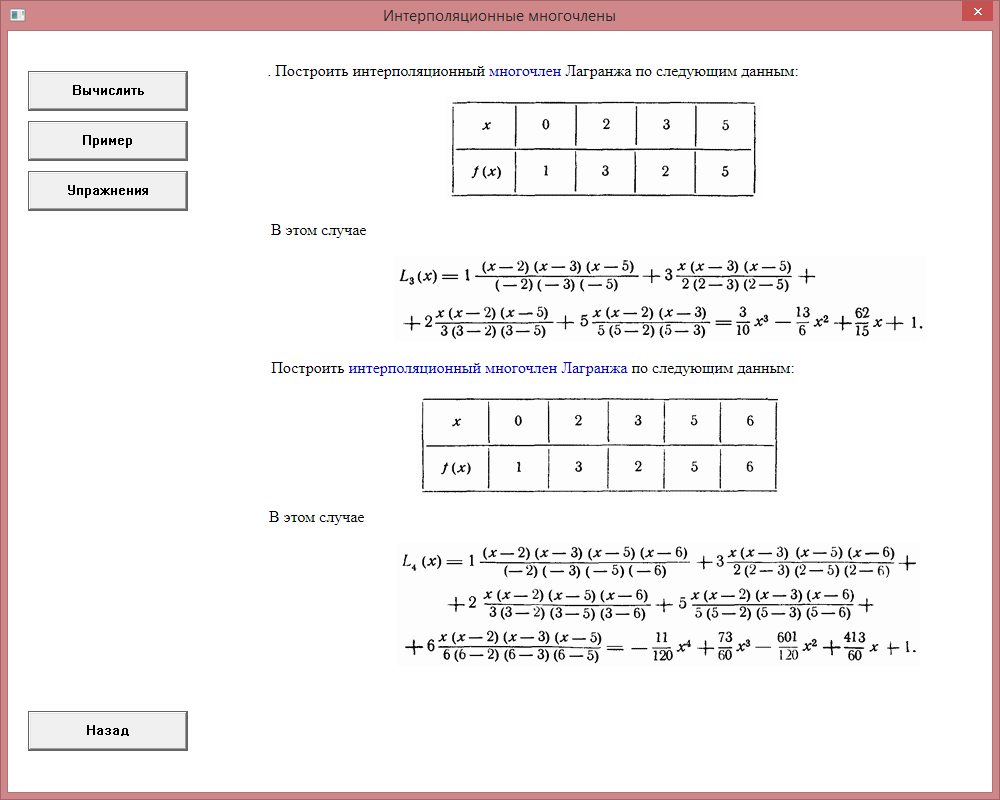
**Рис. 3.2.** **Информационная страница**

Нажатие на кнопку «Назад» вернёт пользователя к стартовому экрану. Кнопка «Пример» открывает раздел, в котором подробно разобрано построение выбранного интерполяционного полинома (рис. 3.3.).



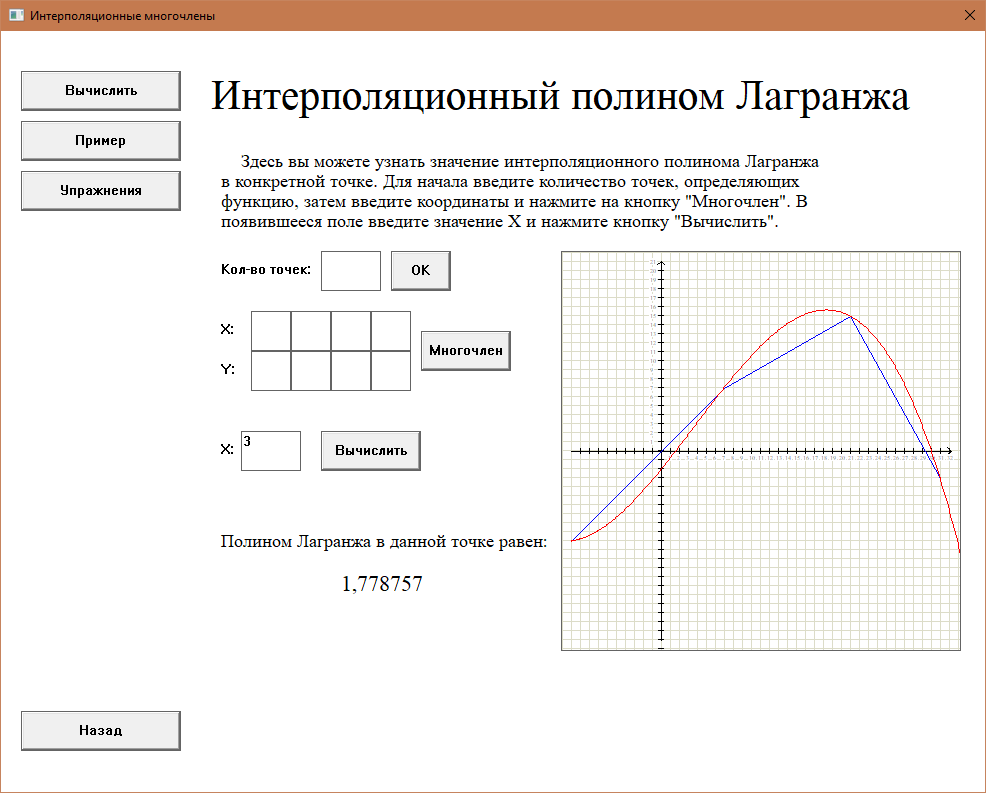
**Рис. 3.3. Страница с примером**

На странице с упражнениями (рис. 3.4.) для самостоятельной подготовки размещены задания и соответствующие им ответы для проверки.



**Рис. 3.4. Упражнения**

В разделе вычисления значения полинома расположены необходимые элементы управления и поля ввода, которые появляются по мере ввода информации. Результат вычисления значения полинома Лагранжа для тестовых данных представлен на рис. 3.5.



**Рис. 3.5. Вычисление значения полинома в точке**

## 3.2. Тестирование

После разработки приложения был проведён ряд различных тестов, направленных на проверку работоспособности приложения.

При вводе недопустимых значений в поля ввода появляется соответствующее предупреждение.

В целях безошибочной работоспособности приложения количество точек для определения полинома ограничено сверху числом 15 и нулём снизу.

Проведенное тестирование программы не выявило ошибок, т.к. подобные действия пользователя учитывались при разработке приложения.

В целом можно сделать вывод, что тестирование прошло успешно.

ЗАКЛЮЧЕНИЕ

При выполнении данной работы, потребовались знание языка программирования С++ и Windows API. В ходе написания программы требовалось обращаться к дополнительным источникам, чтобы более подробно разобраться с построением графического интерфейса Windows, обработкой событий системы и созданием различных элементов окон.

Разработана демонстрационно-обучающая программа по методам интерполяции функций, которая может служить наглядным примером использования интерполяционных многочленов Ньютона и Лагранжа

Тестирование прошло отлично, явных сбоев в работе приложения не было выявлено.

В целом данная работа помогла лучше освоить язык С++ и улучшить свои знания в WinApi.

В дальнейшем возможно усовершенствование данного приложения, путём добавления новых методов интерполяции.

СПИСОК ЛИТЕРАТУРЫ

1. Азарченков, А.А. Оформление текстовых документов: методические указания для студентов всех форм обучения всех специальностей/ А.А. Азарченков. Брянск: БГТУ, 2012. – 45с.
2. Булатицкий, Д.И. Методические указания к выполнению курсового проекта/ Д.И. Булатицкий. Брянск: БГТУ, 2006. – 7с.
3. Аммерал, Л. Принципы программирования в машинной графике. Перевод с английского В.А. Львова. (Москва: Издательство «Сол Систем», 1992. - Серия «Машинная графика на языке СИ»).
4. Румянцев, П.В. Азбука программирования в Win32 Api. – 4 изд. – М.: «Горячая линия – Телком»/ П.В. Румянцев. 2004. – 312 стр.
5. Павловская, Т.А. C/C++. Программирование на языке высокого уровня/ Т. А. Павловская – СПБ.: Питер, 2006. – 246 с.
6. Подбельский, В.В. Программирование на языке Си: учеб. пособие/ Подбельский, С.С. Фоними-финансы и статистика, 2005 – 600с.
7. Шильдт, Г. Самоучитель C++: Пер. с англ. – 3 изд. – СПб.: «BHV – Петербург»/ Г.Шильдт. 2004. – 688 стр.

# ПРИЛОЖЕНИЕ 1

Листинг 1. Обработка кнопок и сценарии отрисовки

case WM\_PAINT:

{

if (runTable)

{

resetTable();

}

reset();

HDC hDC;

PAINTSTRUCT ps;

hDC = BeginPaint(hWnd, &ps);

// Получить контекст окна

if (curMethod == HOME)

{

drawHome(hDC, hWnd);

}

else

{

switch (curScreen)

{

case info:

{

drawInfo(hDC, hWnd, curMethod);

//inWork = true;

break;

};

case graphic:

{

//inWork = true;

drawGraph(hDC, hWnd, curMethod);

break;

};

case examples:

{

drawExample(hDC, hWnd, curMethod);

break;

};

case exercises:

{

drawExercises(hDC, hWnd, curMethod);

break;

};

}

}

EndPaint(hWnd, &ps); // Освободить контекст окна

}; break;

case WM\_COMMAND:

{

switch (wParam)

{

// Пауза

case ID\_BUTTON\_NEWTON:

{

curScreen = info;

curMethod = NEWT;

InvalidateRect(hWnd, NULL, true);

UpdateWindow(hWnd);

}; break;

case ID\_BUTTON\_BACK:

{

curScreen = info;

curMethod = HOME;

inWork = false;

InvalidateRect(hWnd, NULL, true);

UpdateWindow(hWnd);

}; break;

case ID\_BUTTON\_LAGRANZ:

{

curScreen = info;

curMethod = LAGR;

InvalidateRect(hWnd, NULL, true);

UpdateWindow(hWnd);

}; break;

case ID\_BUTTON\_GRAPH:

{

curScreen = graphic;

InvalidateRect(hWnd, NULL, true);

UpdateWindow(hWnd);

}; break;

case ID\_BUTTON\_EXAMP:

{

curScreen = examples;

InvalidateRect(hWnd, NULL, true);

UpdateWindow(hWnd);

}; break;

case ID\_BUTTON\_EXERC:

{

curScreen = exercises;

InvalidateRect(hWnd, NULL, true);

UpdateWindow(hWnd);

}; break;

case ID\_BUTTON\_CHECK:

{

char out\_str[16];

GetWindowText(hEditCheck, out\_str, 16);

if (atoi(out\_str) && (atoi(out\_str) > 0) && (atoi(out\_str) < 15))

{

pointX = atoi(out\_str);

result = true;

}

else

{

MessageBox(NULL,

"Недопустимое значение!", "Ошибка ввода", MB\_OK | MB\_ICONSTOP);

}

InvalidateRect(hWnd, NULL, true);

UpdateWindow(hWnd);

}; break;

case ID\_BUTTON\_SET\_COUNT:

{

char out\_str[16];

GetWindowText(hEditCount, out\_str, 16);

if (atoi(out\_str) && (atoi(out\_str) > 0) && (atoi(out\_str) < 12))

{

oldCount = valCount;

valCount = atoi(out\_str);

inWork = true;

}

else

{

MessageBox(NULL,

"Недопустимое значение!", "Ошибка ввода", MB\_OK | MB\_ICONSTOP);

}

InvalidateRect(hWnd, NULL, true);

UpdateWindow(hWnd);

}; break;

case ID\_BUTTON\_DRAW:

{

char out\_str[16];

bool bad = false;

X.clear();

Y.clear();

for (int i = 0; i < 2; i++)

{

for (int j = 0; j < valCount; j++)

{

char out\_str[16];

GetWindowText(table[i][j], out\_str, 16);

if (atoi(out\_str) && (i == 0))

{

X.push\_back(atoi(out\_str));

}

else if (atoi(out\_str) && (i == 1))

{

Y.push\_back(atoi(out\_str));

}

else

{

MessageBox(NULL,

"Недопустимое значение!", "Ошибка ввода", MB\_OK | MB\_ICONSTOP);

bad = true;

break;

}

}

if (bad)

{

break;

}

}

if (!bad)

{

runGraph = true;

}

InvalidateRect(hWnd, NULL, true);

UpdateWindow(hWnd);

}; break;

}

}; break;

Листинг 2. Функция рисования графика

|  |
| --- |
| void drawGraphic(HDC hDC, HWND hWnd, vector<double> xX, vector<double> yY, COLORREF NewColor)  {  RECT r;  GetClientRect(hWnd, &r);  vector<double> X = xX;  vector<double> Y = yY;    POINTXY \_1stPoint, \_2ndPoint;  \_1stPoint = System.startPoint; // Первой точкой рисования устанавливаем начало координат  HPEN hPen = CreatePen(PS\_SOLID, 1, NewColor);  HPEN hOldPen = (HPEN)SelectObject(hDC, hPen);    System.XStep \*= 1;  for (int i = 0; i < (X.size() - 1); i++)  {  \_1stPoint.x = System.startPoint.x + (System.XStep \* X[i]);  \_1stPoint.y = System.startPoint.y + ((System.YStep \* (-1)) \* Y[i]);  \_2ndPoint.x = System.startPoint.x + (System.XStep \* (X[i + 1]));  \_2ndPoint.y = System.startPoint.y + ((System.YStep \* (-1)) \* (Y[i + 1]));  // Помещаем точку рисования в начало координат  MoveToEx(hDC, \_1stPoint.x, \_1stPoint.y, NULL);  LineTo(hDC, \_2ndPoint.x, \_2ndPoint.y);  }  SelectObject(hDC, hOldPen);  DeleteObject(hPen);  } |