**Курсовая работа  
по информационным технологиям  
по теме «Графическое приложение: вычисление дискретного преобразования Фурье для синусоиды с линейно изменяющейся частотой»**

|  |  |
| --- | --- |
| Выполнил студент ФРТ гр. 1183 | Чаминов Д. А. |
| Преподаватель | Ситников И. Ю. |

# Содержание

[Содержание 2](#_Toc104906102)

[Спецификация задания 3](#_Toc104906103)

[Постановка задачи 5](#_Toc104906104)

[Формализованное описание алгоритма решения задачи 7](#_Toc104906105)

[Варианты взаимодействия оператора и программы (Use Case) 10](#_Toc104906106)

[Диаграмма потоков данных (Data flow diagram) 11](#_Toc104906107)

[Выбор и обоснование типов переменных. Разработка структур данных 11](#_Toc104906108)

[Вводимые и выводимые параметры и их типы 15](#_Toc104906109)

[Диаграмма классов 16](#_Toc104906110)

[Структура проекта, перечисление нужных файлов 17](#_Toc104906111)

[Инструкция по использованию 19](#_Toc104906112)

[Текст программы и файлов заголовков с комментариями 23](#_Toc104906113)

[Рисунки с копиями экрана при работе программы 78](#_Toc104906114)

[Контрольный пример, сравнение результата с эталоном 82](#_Toc104906115)

[TODO: Ведомость соответствия программы спецификации 87](#_Toc104906116)

[TODO: Выводы 88](#_Toc104906117)

# Спецификация задания

* 1. Требования к расчёту и программе:
     1. Программа должна выполнять дискретное преобразование Фурье (ДПФ) для сигнала вида в соответствии с введенными пользователем параметрами и визуализировать результат расчёта в виде графика модуля ДПФ;
     2. Программа должна строить график сигнальной функции с возможностью его масштабирования, смещения, использования логарифмического масштаба по оси ординат, изменения этих и параметров сигнальной функции пользователем, а также иметь возможность сохранения графика в файл в формате BMP (bitmap image);
     3. Поддержка универсальной символьной кодировки Unicode;
     4. Наличие графического интерфейса;
     5. Использование пользовательского типа данных и оперирование динамическими массивами;
     6. Программа должна разрабатываться в среде MS Visual Studio на языке программирования C++ с применением библиотек Microsoft Foundation Classes (MFC) как оконное приложение на базе диалогов для операционных систем MS Windows 7 (x32) и выше.
     7. Приложение должно иметь сведения о программе, авторе и авторских правах, название и иконку;
     8. Интерфейс должен использовать кнопки для управления приложением, поля ввода данных для задания сигнальной функции;
     9. В приложении должны содержаться списки выбора;
     10. Интерфейс должен быть русскоязычным, интуитивно понятным, соответствовать понятию «дружественный интерфейс»;
  2. Требования к отчёту:
     1. Отчёт должен соответствовать ГОСТу 19.701–90 единой системы программной документации;
     2. Отчет должен содержать описание программного интерфейса, диаграмму классов и диаграмму потоков данных, выбор и обоснование переменных, пользовательских типов и классов, код программы с комментариями, пример работы программы и контрольный пример;

Срок сдачи отчета:

Срок сдачи курсовой работы:

|  |  |  |
| --- | --- | --- |
| Преподаватель |  | Ситников И. Ю. |
| Студент |  | Чаминов Д. А. |

# Постановка задачи

Задача: для сигнала вида , где – параметры, задаваемые пользователем рассчитать дискретное преобразование Фурье, построить график сигнальной функции и график модуля ее ДПФ.

Дискретное преобразование Фурье переводит вектор в вектор частотных отсчетов . Такое преобразование можно рассматривать умножение матрицы на вектор , где – матрица преобразования (унитарная матрица для некоторого базиса пространства ).

Для компонента преобразования справедлива следующая формула:  
где – число отсчетов, – отсчет сигнальной функции с номером , – число Эйлера, – число пи и – мнимая единица. Применяя к этому выражению формулу Эйлера, получим следующее:

Учитывая четность и нечетность , а также то, что в данной задаче сигнал вещественный, получим:

Модуль

Для расчёта положения точки на экране необходимо перейти к другому масштабу и учесть смещение. Если отрезок длиной 1 должен на экране по оси совпадать с длиной пикселей, а по оси – , левая верхняя точка графика имеет координаты , а высота графика равна пикселей, то для перехода к координатам окна можно использовать следующие преобразования:

Разность координат показывает смещение точки относительно координат окна, но в координатах графика; умножение на коэффициент масштаба показывает длину в пикселях проекции радиус-вектора, проведенного из нового начала координат в эту точку; знак минус перед показывает отражение системы координат, а компенсирует вызванное таким поворотом смещение.

# Формализованное описание алгоритма решения задачи

Для выполнения расчётов в программе определено несколько пользовательских классов. Все классы диалогов являются наследниками CDialogEx.

Первый из них – класс начального окна courseworkDlg. Он содержит один CStatic элемент, в котором дано краткое описание задачи, кнопку открытия диалога с более подробными сведениями о задаче, кнопку открытия диалога с расчётом и кнопку закрытия программы.

Диалог с расчётом Calculator содержит в себе элементы управления для настройки параметров сигнальной функции и графиков: 9 CEdit (3 для параметров функции и 6 для настроек масштаба) , 2 CSliderCtrl для удобной настройки шага графика и выбора числа отсчётов для ДПФ, 3 CButton – check box (флажок) для установки логарифмического масштаба для каждого из графиков и включения анимации, 3 CMFCColorButton – выпадающие списки для выбора цвета, 4 кнопки (обновление графика, его сохранение, сброс к начальным параметрам и закрытие диалога), 20 CStatic (1 для отображения сигнальной функции, 1 для отображения координат курсора, 1 для отображения количества отсчётов и 17 для пояснения приглашений ко вводу), а также 2 объекта пользовательского класса CMyGraph для отрисовки графиков и объекты пользовательских классов SignalFunction и DFTFunction, которые описывают сигнальную функцию и дискретное преобразование Фурье, необходимые для построения графиков.

При нажатии кнопки сохранения должен вызываться метод для вызова диалога сохранения void Calculator::OnBnClickedButtonSaveGr(), в котором из объектов CMyGraph извлекаются картинки, объединяются в одну, и получившееся изображение сохраняется в выбранное место.

При нажатии кнопки сброса вызывается метод void Calculator::OnBnClickedButtonreset(), в котором сбрасываются значения всех элементов управления кроме кнопки включающей анимацию.

При нажатии кнопки обновления вызывается метод void Calculator::OnBnClickedButtonreset(), в которой вызывается метод получения параметров из всех элементов управления void Calculator::UpdateCalculatorParams(), в котором происходит проверка заполнения элементов (с созданием диалогового окна об ошибке или присвоением соответствующих параметров через методы CMyGraph или напрямую объектам сигнальной функций и объекту ДПФ). Помимо обновления параметров, из этого метода через CWnd::RedrawWindow() система отправляет элементу CMyGraph сообщение о необходимости перерисовать окно.

CMyGraph является наследником CStatic. Он содержит в себе автоматический динамический массив объектов типа MathFunction\* – ссылки на функции, графики которых необходимо построить, 2 объекта CBitmap с фоном и готовым графиком, а также 2 структуры из 2-х чисел типа double – границы области построения.

При получении сообщений о перерисовке вызывается метод void CMyGraph::OnPaint(), который в свою очередь вызывает метод void CMyGraph::Draw, специализирующийся на рисовании графика в объекте CBitmap (этот метод оборачивает вызов методов void CMyGraph::DrawBg и void CMyGraph::DrawGraph, отрисовывающих в CBitmap фон и график), а затем копирует готовую картинку на экран. Для анимации используется частичная отрисовка графика (определенное число точек) и отрисовка столбцов меньшей длины для гистограммы по сообщениям таймера с увеличением числа точек и высоты столбцов до действительных значений.

Класс CMyGraph обладает множеством set’еров, которые вызывают соответствующие методы для каждой из функций.

MathFunction – абстрактный класс (интерфейс), объекты наследников (реализаций) которого содержат сведения об области определения, области построения, размерах расчётного промежутка, типе масштаба, цвете линий функции на графике, а также о том, являются ли рассчитанные точки актуальными; методы установки параметров, расчёта математических точек (чисто виртуальная функция), расчёта точек (в координатах окна), пересчёта точек в координаты окна, автоматический динамический массив объектов POINT – точек в координатах окна и метод его передачи для отрисовки, а также автоматический динамический массив double – значений отсчетов для ДПФ.

Метод передачи точек MathFunction::get\_points() проверяет их актуальность и вызывает (в случае если данные устарели) метод пересчёта точек MathFunction::calulate(), который с заданным шагом на промежутке пересечения области определения и расчётной области вызывает метод MathFunction::f() расчёта функции в этой точке и метод перевода координат, полученной таким образом точки в координаты окна POINT MathFunction::to\_the\_new\_coords\_system(double x, double y) const, которая основываясь на формулах пересчета приведенных выше вычисляет координаты точки в окне.

Классы SignalFunction и DFTFunction содержат в себе некоторые дополнительные сведения (коэффициенты и значения функции в отсчетах для сигнальной функции и ссылка на сигнальную функцию для класса функции ДПФ) и методы установки этих параметров.

Расчёт в классе SignalFunction отличается тем, что значения функции в точках с шагом дискретизации 10 МГц сохраняются в vector<double> samples для дальнейшего применения ДПФ.

В классе DFTFunction расчёт отличается тем, что в качестве параметра функции f передается номер отсчета, для которого производятся вычисления. В самой функции f происходит расчёт мнимой и действительной частей при помощи суммирования данных сигнальной функции с коэффициентами, соответствующими номерам отсчета. Затем находится модуль ДПФ путем извлечения корня из суммы квадратов мнимой и реальной частей, рассчитывается частота, соответствующая этом отсчету.

# Варианты взаимодействия оператора и программы (Use Case)

# Диаграмма потоков данных (Data flow diagram)

# Выбор и обоснование типов переменных. Разработка структур данных

В программе для данных о координатах окна или о координатах объекта в окне используется типы POINT и Cpoint, которые хранят два целых числа int, диапазона которых достаточно, чтобы однозначно указать одну из координат точки экрана.

Для хранения данных о границах элементов используются типы CRect и RECT, которые хранят координаты двух противолежащих точек прямоугольника в виде целых чисел int.

Для представления цвета используется тип COLORREF, который является синонимом беззнакового 32-разрядного целого (unsigned long int) хранит данные о трех компонентах цвета (красной, зеленой и голубой) в виде битовых полей. Для краткой записи преобразования 3-х восьмибитных компонент используется макрос RGB.

В качестве блока расчёта выступают несколько объектов типа MathFunction. Этот тип содержит базовые данные о «функции» (здесь под функцией MathFunction подразумевается некоторый объект, который численными методами генерирует последовательность точек в данной области с заданным шагом для отрисовки графика, причем точки определяются в координатах области построения): область определения некоторой функции (в виде 2х чисел с плавающей точкой двойной точности double), данные об области вычислений (в виде анонимной структуры из 4х чисел double, содержащей начальные и конечные значения по каждой из осей), данные об области отрисовки (CRect), шаг расчёта (double), данные о цвете графика функции (COLORREF), индикатор того, произведен ли уже расчёт (bool) и индикатор логарифмического масштаба (bool). Все вышеперечисленные параметры кроме цвета являются защищенными (недоступны вне дочерних классов), поскольку их изменение напрямую может привести к недостоверности сведений об актуальности рассчитанных координат (несоответствие индикатора is\_calculated действительности, которое в свою очередь приводит к сохранению неактуальных данных). Для этих параметров определены соответствующие set’теры; get’теры определены только для массива точек и индикатора готовности.

Наследником класса MathFunction является SignalFunction, который дополнительно хранит параметры сигнальной функции (3 числа double) и автоматический динамический массив std::vector<double> - массив значений функции для дальнейшего расчёта ДПФ и get’тер для этого массива, возвращающий константную ссылку.

Наследником SignalFunction является DFTFunction, который производит дискретное преобразование Фурье. Этот класс обладает 1 дополнительным атрибутом – ссылкой на объект сигнальной функции (SignalFunction\*), через которую осуществляется доступ к данным отсчетов сигнальной функции. Атрибуты SignalFunction (параметры ) в этом классе необходимы для того, чтобы отслеживать изменения сигнальной функции независимо от неё – обращение с обновлением к DFTFunction, а затем только к SignalFunction приведет к обновлению только сигнальной функции. Причем этот порядок может быть любым, поэтому появляется необходимость в этих параметрах.

Построением графиков занимаются объекты класса CMyGraph – наследник класса CStatic из библиотек MFC. Переменные-члены этого класса можно условно разделить на 4 группы: необходимые для анимации, отвечающие за растровые изображения графиков, влияющие только на визуальную составляющую (в их отсутствии график был бы правильным, но выглядел бы хуже), отвечающие за отрисовку самого графика.

К первой группе относятся 3 переменные – текущая фаза анимации деленная на  (т. е. в долях от единицы) (float), скорость анимации – то, на сколько увеличится фаза за 1 кадр (циклическая частота) (float), показывающий идет ли анимация индикатор (bool), который при включенной анимации эквивалентен выражению .

Ко второй группе можно отнести 2 переменные CBitmap (из библиотек MFC), в которых хранятся фоновое изображение с осями и сам график, индикаторы готовности и актуальности этих растровых изображений – 2 переменных типа bool, 2 объекта CDC (из библиотек MFC) – инструменты для отрисовки фона и графика и 2 HGDIOBG – указатели на объекты начальных настроек CDC.

К третьей группе относятся 1 переменная POINT, показывающая смещение графика относительно области построения (вдоль осей графика, но в точках экрана), индикатор, показывающий, является ли график гистограммой – 1 bool, 1 переменная типа int, которая содержит в себе длину засечек на осях, 1 объект, представляющий анонимную структуру из 2х чисел int, показывающий число засечек на каждой из осей, 1 переменная COLORREF – цвет фона.

К последней группе относятся 2 экземпляра анонимной структуры из двух чисел double, определяющие область построения (границы значений по каждой из осей), индикатор логарифмического масштаба bool, автоматический динамический массив указателей на объекты «функций» std::vector<MathFunction\*>, которые необходимо отобразить на графике. Поскольку SignalFunction и DFTFunction являются реализациями интерфейса MathFunction, указатели на них можно использовать в качестве указателей на MathFunction. Иными словами – объекты типов SignalFunction и DFTFunction являются и объектами типа MathFunction.

Следующие пользовательские типы – классы окон приложения. Они являются наследниками класса CDialogEx (из библиотек MFC). Всего в программе определено 4 класса окна: главное окно с кратким описанием задания, окно с более подробными теоретическими сведениями, окно «О программе» и окно для расчёта. Окна с описанием программы и теоретическими сведениями не содержат атрибутов. Начальное окно содержит текст Cstatic – краткий текст задания и иконку HICON. Окно расчёта содержит в себе множество элементов контроля: 3 CMFCColorButton для выбора цвета, 2 CSliderCtrl для выбора шага и числа отсчётов, 9 CEdit для ввода значений, 3 CButton для установки логарифмического масштаба и включения анимации. Кроме элементов контроля это окно содержит в себе 2 CMyGraph для отображения графиков сигнальной функции и гистограммы модуля её ДПФ.

Само оконное приложение – единственный экземпляр класса CcourseworkApp, являющегося наследником CWinApp и не имеющего атрибутов.

# Вводимые и выводимые параметры и их типы

В программу через элементы контроля поступают следующие данные:

1. Через CEdit поступают строки CString, содержащие основную информацию о сигнальной функции и областях построения графиков. Эти данные преобразуются к типу double, поэтому вводимые строки могут содержать только знак (+ или -), цифры, точку и символы “e” или “E” для записи числа в экспоненциальной форме.
2. Через кнопки CButton поступает информация о режимах масштаба и анимации в виде логических переменных типа bool.
3. Через ползунок CSliderCtrl в программу поступает целое число – шаг построения графика (в пикселях).
4. Через такой же ползунок поступает целое число – количество отсчётов.
5. Через элементы CMFCColorButton поступают переменные типа COLORREF, содержащие информацию о цветах графика, выбранных пользователем.

Также в программу поступают данные о состоянии курсора (для трассировки графиков) в виде переменной CPoint, содержащей его координаты, и переменной UINT\_PTR – набора флагов состояния курсора и основных управляющих клавиш.

Вывод текстовых данных, хранящихся в строках CString, осуществляется с помощью диалоговых окон AfxMessageBox и элементов CStatic.

Вывод графических данных осуществляется 2 способами: через элементы CMyGraph и через сохранение на жестком диске в форматах bmp, gif, png, jpg.

# Диаграмма классов



# Структура проекта, перечисление нужных файлов

Основные файлы программы находятся в директории coursework.

Файлы Calculator.h, courseworkDlg.h содержат объявления классов диалоговых окон (расчёта, основного окна и окна «О программе»), а соответствующие .cpp файлы содержат реализации этих классов. В файлах CMyGraph.h и CMyGraph.cpp содержится объявление и определение класса CMyGraph – элемента окна, в котором происходит отрисовка графиков. В файлах MathFunction.h и MathFunction.cpp содержатся объявления и определения классов, которые определяют вид функций для отрисовки в элементах CMyGraph. В файлах BeautyLib.cpp и BeautyLib.h содержатся объявления и определения функций, помогающих преобразовать число к удобному формату для вывода. Файл logo.ico содержит иконку окна приложения, а coursework.ico – иконку исполняемого файла. Файлы coursework.rc и coursework.rc2 содержат разметку диалоговых окон и данные о программе. Остальные файлы являются служебными, созданы средой разработки MS Visual Studio и являются необходимыми для сборки приложения. Дерево структуры директорий и файлов приведено ниже.

**│ coursework.sln**

**│ Курсовая работа.docx**

**│**

**└───coursework**

**│ BeautyLib.cpp**

**│ BeautyLib.h**

**│ Calculator.cpp**

**│ Calculator.h**

**│ ClassDiagram.cd**

**│ CMyGraph.cpp**

**│ CMyGraph.h**

**│ coursework.aps**

**│ coursework.cpp**

**│ coursework.h**

**│ coursework.rc**

**│ coursework.vcxproj**

**│ coursework.vcxproj.filters**

**│ coursework.vcxproj.user**

**│ courseworkDlg.cpp**

**│ courseworkDlg.h**

**│ framework.h**

**│ logo.ico**

**│ MathFunction.cpp**

**│ MathFunction.h**

**│ pch.cpp**

**│ pch.h**

**│ resource.h**

**│ targetver.h**

**│**

**└───res**

**coursework.ico**

**coursework.rc2**

# Инструкция по использованию

1. Запустите приложение (двойной щелчок левой кнопки мыши (далее ЛКМ) по файлу coursework.exe).
2. Для просмотра дополнительных сведение о преобразовании щелкните левой кнопкой мыши по кнопке с надписью «Подробнее».
3. Для закрытия окна с подробными сведениями щелкните ЛКМ по кнопке «Закрыть» или кнопке с изображением креста в правом верхнем углу окна.
4. Для перехода к вычислениям из главного окна (окно с кратким описанием задания и кнопками «Закрыть», «Подробнее» и «Вычислить») щелкните ЛКМ по кнопке «Вычислить».
5. В поля ввода в соответствии с параметрами, указанными возле них, введите значения соответствующих переменных.
6. Вводимые в эти поля данные могут содержать только число, состоящее из арабских цифр, знака («+» или «-»), точки, или число записанное в экспоненциальном виде (см п. 2.7)
7. При записи числа в экспоненциальном формате выглядят так: “-1.23e45”, где символ “e” или “E” (оба символа относятся к латинскому алфавиту) разделяет число на 2 части, которые будут восприняты программой как число .
8. Вместо числа в программу может быть передано (как и выведено на экран) значение “+inf” или “-inf”. При вводе они будут соответствовать самым большим числам, помещающимся в памяти (границам области допустимых значений данных). Такие числа не рекомендуются к использованию по понятным причинам. В случае, если программа вместо чисел выводит эти значения следует уменьшить диапазон, в котором производятся вычисления.
9. Для навигации между полями ввода и другими элементами окна можно использовать ЛКМ или клавишу «Tab».
10. Поля, между которыми стоит многоточие («…») обозначают диапазоны расчётов и границы области построение. В левое поле следует вводить меньшую из границ, а в правую – большую.
11. При использовании логарифмического масштаба графика следует указывать положительные значения границ построения по оси, для которой включен логарифмический масштаб.
12. Для построения гистограммы отображающей модуль ДПФ в целях удобного отображения диаграммы рекомендуется в качестве нижней границы использовать неотрицательные числа.
13. Для настройки шага расчёта используйте «ползунок» с подписью «Шаг расчёта». Крайнее левое положение соответствует состоянию, при котором в результате масштабирования 1 отсчёт будет иметь ширину в 1 точку экрана вдоль оси абсцисс, а при крайнем правом – 10 точек экрана. Таким образом «действительный» шаг расчёта настраивается еще и шириной диапазона расчёта, т. е. для получения меньшего шага следует передвинуть ползунок в крайнее левое положение и (при необходимости) взять более узкий промежуток по оси абсцисс.
14. Для изменения состояния настройки логарифмического масштаба щелкните ЛКМ по квадрату рябом с надписью «Логарифмический масштаб» или по самой надписи.
15. Состояние, при котором в квадрате отображается «галочка», соответствует состоянию «включен», а при отсутствии «галочки» – «выключен».
16. При включении масштаба границы области построения меняются по правилу , где – новое значение, а – старое, что позволяет удобно задать границы до включения логарифмического масштаба путем ввода показателя степени 10. При выключении происходит обратное преобразование .
17. Включение и выключение анимации происходит аналогично настройке логарифмического масштаба, но с полем «Анимация».
18. Для настройки цветов фона, графика сигнальной функции, столбцов гистограммы модуля ДПФ щелкните ЛКМ по цветному квадрату, расположенному рядом с соответствующей надписью («Фон», «Сигнал», «ДПФ»); в открывшейся палитре выберите цвет, щелкнув по квадрату соответствующего цвета, расположенном в этой палитре.
19. Для отображения графика и диаграммы используйте кнопку «Обновить». Если в квадрате рядом с надписью «Анимация» стоит «галочка», то при нажатии на кнопку «Обновить» автоматически запустится анимация.
20. Для сброса параметров и настроек к начальным используйте кнопку «Сбросить». Все параметры и настройки кроме настроек анимации будут сброшены.
21. Для сохранения картинки с графиком и гистограммой воспользуйтесь кнопкой «Сохранить» или пунктом системного меню (вызывается щелчком правой кнопки мыши по заголовку окна) «Сохранить как», в открывшемся диалоге выберите имя и тип (расширение) файла, нажмите «Сохранить». В случае ошибки во время сохранения вам будет показано сообщение об этом.
22. Для закрытия окна с вычислениями воспользуйтесь кнопкой «Закрыть» или кнопкой с крестиком в заголовке окна.
23. Для просмотра сведений о программе из начального диалогового окна выберите пункт системного меню (см п 2.21) «О программе»
24. В открывшемся окне предоставлена информация о приложении. Чтобы закрыть его воспользуйтесь кнопкой «Закрыть» или кнопкой с крестиком в заголовке окна.
25. Для завершения работы с приложением необходимо закрыть все его окна (при помощи кнопок «Закрыть» или кнопок с крестиком в заголовках окон).

# Текст программы и файлов заголовков с комментариями

//

// pch.h: это предварительно скомпилированный заголовочный файл.

//

// Перечисленные ниже файлы компилируются только один раз, что ускоряет последующие сборки.

#ifndef PCH\_H

#define PCH\_H

// for M\_PI constant

#define \_USE\_MATH\_DEFINES

#include "framework.h"

#endif //PCH\_H

//

// targetver.h

//

#pragma once

// Включение SDKDDKVer.h обеспечивает определение самой последней доступной платформы Windows.

// Если требуется выполнить сборку приложения для предыдущей версии Windows, включите WinSDKVer.h и

// задайте для макроса \_WIN32\_WINNT значение поддерживаемой платформы перед включением SDKDDKVer.h.

#include <SDKDDKVer.h>

//

// pch.cpp: файл исходного кода, соответствующий предварительно скомпилированному заголовочному файлу

//

#include "pch.h"

//

// framework.h

//

#pragma once

#ifndef VC\_EXTRALEAN

#define VC\_EXTRALEAN // Исключите редко используемые компоненты из заголовков Windows

#endif

#include "targetver.h"

#define \_ATL\_CSTRING\_EXPLICIT\_CONSTRUCTORS // конструкторы CString будут явными

// отключает функцию скрытия предупреждений MFC

#define \_AFX\_ALL\_WARNINGS

#include <afxwin.h> // основные и стандартные компоненты MFC

#include <afxext.h> // Расширения MFC

#ifndef \_AFX\_NO\_OLE\_SUPPORT

#include <afxdtctl.h> // поддержка MFC для типовых элементов управления Internet Explorer 4

#endif

#ifndef \_AFX\_NO\_AFXCMN\_SUPPORT

#include <afxcmn.h> // поддержка MFC для типовых элементов управления Windows

#endif // \_AFX\_NO\_AFXCMN\_SUPPORT

#include <afxcontrolbars.h> // поддержка MFC для лент и панелей управления

#ifdef \_UNICODE

#if defined \_M\_IX86

#pragma comment(linker,"/manifestdependency:\"type='win32' name='Microsoft.Windows.Common-Controls' version='6.0.0.0' processorArchitecture='x86' publicKeyToken='6595b64144ccf1df' language='\*'\"")

#elif defined \_M\_X64

#pragma comment(linker,"/manifestdependency:\"type='win32' name='Microsoft.Windows.Common-Controls' version='6.0.0.0' processorArchitecture='amd64' publicKeyToken='6595b64144ccf1df' language='\*'\"")

#else

#pragma comment(linker,"/manifestdependency:\"type='win32' name='Microsoft.Windows.Common-Controls' version='6.0.0.0' processorArchitecture='\*' publicKeyToken='6595b64144ccf1df' language='\*'\"")

#endif

#endif

//{{NO\_DEPENDENCIES}}

//

// Resource.h

//

// Включаемый файл, созданный в Microsoft Visual C++.

// Используется coursework.rc

//

#define IDM\_ABOUTBOX 0x0010

#define IDD\_ABOUTBOX 100

#define IDS\_ABOUTBOX 101

#define IDD\_COURSEWORK\_DIALOG 102

#define IDS\_STRING102 102

#define IDS\_VERSION 102

#define IDS\_COMMIT 103

#define IDD\_Calculator 104

#define IDS\_SHORT\_TASK\_INFO 104

#define IDS\_STRING105 105

#define IDS\_MORE\_TASK\_INFO 105

#define IDR\_MAINFRAME 128

#define ID\_LOGO\_ICO 133

#define IDC\_STATIC\_TASK 1000

#define IDC\_Task\_About\_Bn 1001

#define IDC\_CALCULATE\_BUTTON 1015

#define IDC\_STATIC\_graph 1020

#define IDC\_STATIC\_graph2 1021

#define IDC\_EDIT\_param\_a 1028

#define IDC\_CHECK\_is\_log\_scale 1029

#define IDC\_BUTTON\_UPDATE\_graph 1030

#define IDC\_STATIC\_signal 1031

#define IDC\_CHECK\_is\_log\_scale2 1032

#define IDC\_EDIT\_param\_m 1033

#define IDC\_EDIT\_xscale\_from 1034

#define IDC\_EDIT\_param\_f 1035

#define IDC\_EDIT\_yscale\_from 1036

#define IDC\_EDIT\_xscale\_to 1037

#define IDC\_EDIT\_yscale\_to 1038

#define IDC\_SLIDER\_STEP 1039

#define IDC\_BUTTON1 1040

#define IDC\_BUTTON\_SAVE\_GR 1040

#define IDC\_BUTTON\_SETTINGS\_GR 1041

#define IDC\_BUTTON\_reset 1041

#define IDC\_EDIT\_yscale\_from2 1042

#define IDC\_MFCCOLORBUTTON\_BG 1043

#define IDC\_MFCCOLORBUTTON\_AXES 1044

#define IDC\_EDIT\_yscale\_to2 1044

#define IDC\_MFCCOLORBUTTON\_SIGNAL 1045

#define IDC\_MFCCOLORBUTTON\_DCF 1046

#define IDC\_MFCCOLORBUTTON\_dft 1046

#define IDC\_STATIC\_CPOS 1047

#define IDC\_CHECK\_ANIM 1048

#define IDC\_SLIDER\_SAMPLES 1055

#define IDC\_STATIC\_SAMPLES 1056

// Next default values for new objects

//

#ifdef APSTUDIO\_INVOKED

#ifndef APSTUDIO\_READONLY\_SYMBOLS

#define \_APS\_NEXT\_RESOURCE\_VALUE 138

#define \_APS\_NEXT\_COMMAND\_VALUE 32771

#define \_APS\_NEXT\_CONTROL\_VALUE 1057

#define \_APS\_NEXT\_SYMED\_VALUE 107

#endif

#endif

//

// BeautyLib.h

//

#pragma once

#include <atlstr.h> // for CString

#include <utility> // for std::pair

/// <summary>

/// Returns formatted string (if it needed) like '123.1234\*10^3'

/// </summary>

/// <param name="number">the number</param>

/// <param name="exponent">sets the hight bound of

/// the regular number mode. When the exponent of the number

/// will be greater than it (or less then the negative exp)

/// there will be used exponential notation</param>

/// <param name="after\_dot">amount of digits after the dot</param>

/// <returns>formatted string</returns>

CString beautifulRepresentation(double number, int exponent = 3, int after\_dot = 4);

/// <summary>

/// Returns the significant and

/// the exponent (decimal) of the number

/// </summary>

/// <param name="number">the number</param>

/// <returns>pair of significant and exponent</returns>

std::pair<double, int> getME(double number);

//

// BeautyLib.cpp

//

#include "pch.h"

#include "BeautyLib.h"

#include <cmath>

CString beautifulRepresentation(double number, int exponent /\* = 3 \*/, int after\_dot /\* = 4 \*/) {

// place for result

CString res;

// getting the significant and exponent of a number

auto me = getME(number);

// string for a number format

CString format;

// setting the format

format.Format(L"%s%d%c", L"%.", after\_dot, L'f');

// if hight bound reached

if (abs(me.second) > exponent) {

// setting brackets for negative

if (me.second < 0) {

format += L"\u221910^(%d)";

} else {

format += L"\u221910^%d";

}

// formatting result in exp mode

res.Format(format, me.first, me.second);

} else {

// formatting result in regular mode

res.Format(format, number);

}

return res;

}

std::pair<double, int> getME(double number) {

/\* n = m \* 10^(e) \*/

// if zero return zero

if (number == 0) {

return std::make\_pair(0., 0);

}

// place for result

double m;

int e;

// lg(n) = lg(m) + e, lg(m) < 1

e = floor(log10(abs(number)));

// m = n / 10^(e)

m = number / pow(10, e);

// return the result

return std::make\_pair(m, e);

}

//

// MathFunction.h

//

#pragma once // include this file once

#include <vector> // vector of dots/points

/// <summary>

/// class defines type of object that will be

/// drawed in CMyGraph element

/// </summary>

class MathFunction

{

protected:

/// <summary>

/// deafult dot calculation function

/// is used in default points calculation function

/// </summary>

/// <param name="x">argument</param>

/// <returns>function value in the x</returns>

virtual double f(double x) = 0;

// pure virtual function (must be defined in subclasses)

// definition scope

double from, to;

// calculating step (in math coords)

double step = 1;

// indicates if axis is log

bool is\_log;

// calculating scope

struct

{

double x\_from, x\_to;

double y\_from, y\_to;

}scale;

// indicates if vector is ready

bool is\_calculated = false;

// drawing area

CRect rect;

// vector of the points

std::vector<POINT> points;

public:

// color on the graph

COLORREF color;

public:

// setters

bool setScale(double x\_from, double x\_to, double y\_from, double y\_to);

bool setDefinitionScope(double from, double to);

bool setStep(double s);

bool setRect(CRect r);

bool setLog(bool b);

void setColor(COLORREF rgb);

void setNotCalculated();

// getters

bool getCalculated();

virtual const std::vector<POINT>& get\_points();

protected:

/// <summary>

/// calculates vector of the points

/// </summary>

virtual void calculate();

/// <summary>

/// math coords to the element coords conversion

/// </summary>

/// <param name="x">x coord</param>

/// <param name="y">y coord</param>

/// <returns>POINT in element coords</returns>

POINT coordsToDot(double x, double y) const;

};

/// <summary>

/// Subclass that defines signal function like x(t)=a\*sin(x\*(m\*x+f))

/// </summary>

class SignalFunction : public MathFunction

{

protected:

// parameters of the signal function

double a, m, f\_;

// samples amount

size\_t samples\_amount = 100;

static const double samples\_step;

// vector of the dots for DFT

std::vector<double> samples;

protected:

/// <summary>

/// Calculates x(t)=a\*sin(x\*(m\*x+f))

/// </summary>

/// <param name="t"></param>

/// <returns>x(t)</returns>

virtual double f(double t) override;

public:

// setters

bool set\_a(double a);

bool set\_m(double m);

bool set\_f(double f);

bool set\_samples\_amount(size\_t N);

// dots vector getter

const std::vector<double>& getData();

// overriding of the parent calculation function

virtual void calculate() override;

};

/// <summary>

/// Subclass that defines DFT for signal function

/// </summary>

class DFTFunction : public SignalFunction

{

public:

/// <summary>

/// Constructs the class

/// </summary>

/// <param name="s">a pinter to the SignalFunction - signal function</param>

DFTFunction(SignalFunction\* s = nullptr);

protected:

/// <summary>

/// calculates one sample

/// </summary>

/// <param name="x">number of the sample (is double cos of overriding)</param>

/// <returns>|Xk|</returns>

double f(int m);

// a pointer to a signal function

SignalFunction\* signal;

public:

// calculation overriding

void calculate() override;

// signal function setter

void set\_signal(SignalFunction\* s);

};

//

// MathFunction.cpp

//

#include <math.h> // for math like sin, log, etc

#include <cassert>// for asserts

// including project files

#include "pch.h"

#include "MathFunction.h"

void MathFunction::setNotCalculated() {

// turn is calculated indicator to false

is\_calculated = false;

}

bool MathFunction::setScale(double x\_from, double x\_to, double y\_from, double y\_to) {

// if scale is not changed return false

if (x\_from == scale.x\_from && y\_from == scale.y\_from &&

x\_to == scale.x\_to && y\_to == scale.y\_to) {

return false;

}

// else

setNotCalculated();

// construct and assign scale

scale = { x\_from, x\_to, y\_from, y\_to };

return true;

}

bool MathFunction::setDefinitionScope(double from, double to) {

// if scale is not changed return false

if (from == this->from && to == this->to) { return false; }

// else

setNotCalculated();

// assign the scope

this->from = from;

this->to = to;

return true;

}

bool MathFunction::setStep(double s) {

// if step is not changed return false

if (step == s) { return false; }

// else

setNotCalculated();

// assign step

step = s;

return true;

}

bool MathFunction::setRect(CRect r) {

// if drawing area isn't changed return false

if (r.left == rect.left && r.right == rect.right &&

r.top == rect.top && r.bottom == rect.bottom) {

return false;

}

//else

setNotCalculated();

// assign drawing rect

rect = r;

return true;

}

bool MathFunction::setLog(bool b) {

// if log indicator is not changed return false

if (b == is\_log) { return false; }

// else

setNotCalculated();

// assign indicator

is\_log = b;

return true;

}

void MathFunction::setColor(COLORREF rgb) {

// set the color

color = rgb;

}

bool MathFunction::getCalculated() {

return is\_calculated;

}

const std::vector<POINT>& MathFunction::get\_points() {

// if vector is not ready

if (!is\_calculated) {

calculate(); // calculate

}

return points; // return the vector

}

POINT MathFunction::coordsToDot(double x, double y) const {

// coord shifting to the coords with zero in (x\_from,y\_from)

double shifted\_x = x - scale.x\_from;

double shifted\_y = y - scale.y\_from;

// multiplication with scale coefs +-.../(... - ...)

// makes the distance between 2 points correct

// and shifting makes (x\_from, y\_from) be in the left bottom corner

x = rect.left + shifted\_x \* rect.Width() / (scale.x\_to - scale.x\_from);

y = rect.bottom - shifted\_y \* rect.Height() / (scale.y\_to - scale.y\_from);

// create point and return it

return POINT({ (long)round(x), (long)round(y) });

}

void MathFunction::calculate() {

// calculating range calculating

double start = max(scale.x\_from, from);

double stop = min(scale.x\_to, to);

// reserving memory for the points

points.resize(ceil((stop - start) / step));

// for points with the step

for (size\_t i = 0; i < points.size(); i++) {

double x = start + step \* i; // calculating x of the point (progression)

double y = f(x); // calculating y of the point

if (is\_log) { // if log scale

y = log10(abs(y)); // calculate log of y

}

points[i] = coordsToDot(x, y); // coord conversion

}

is\_calculated = true; // set calculated

}

const double SignalFunction::samples\_step = 1e-7; // const initialization

double SignalFunction::f(double t) {

// signal function ↓

return a \* sin(2 \* M\_PI \* (f\_ + m \* t) \* t);

}

bool SignalFunction::set\_a(double a) {

// if 'a' is not changed return false

if (a == this->a) { return false; }

// else

setNotCalculated();

// change the 'a'

this->a = a;

return true;

}

bool SignalFunction::set\_m(double m) {

// like with 'a' param

if (m == this->m) { return false; }

setNotCalculated();

this->m = m;

return true;

}

bool SignalFunction::set\_f(double f) {

// like with 'a' param

if (f == this->f\_) { return false; }

setNotCalculated();

this->f\_ = f;

return true;

}

bool SignalFunction::set\_samples\_amount(size\_t N) {

// check if not changed

if (samples\_amount == N) { return false; }

// mark as irrelevant

setNotCalculated();

// set

samples\_amount = N;

// return that there was difference

return true;

}

const std::vector<double>& SignalFunction::getData() {

// if data is not relevant

if (!is\_calculated) {

calculate(); // update data

}

return samples; // return it

}

void SignalFunction::calculate() {

// calculating scope of the calculating

double start = max(scale.x\_from, from);

double stop = min(scale.x\_to, to);

// memory reserving for the samples data

samples.resize(samples\_amount);

// repeat for each i = 0, 1, ..., samples\_amount-1

for (int i = 0; i < samples\_amount; i++) {

// calculate x (like an arithmetic progression)

// calculate f(x)

// write f(x) to the samples vector

samples[i] = f(i \* samples\_step + start);

}

// amount of dots on the graph calculation

int N = ceil((stop - start) / step);

// memory reserving for the points vector

points.resize(N);

// repeat for each i = 0, 1, ..., N

for (int i = 0; i < N; i++) {

// calculate x (like an arithmetic progression)

double x = i \* step + start;

// calculate f(x)

double y = f(x);

// use logarithm if needed

if (is\_log) { y = log10(y); }

// converse coords and write

// the point to the vector

points[i] = coordsToDot(x, y);

}

// mark as calculated

is\_calculated = true;

}

DFTFunction::DFTFunction(SignalFunction\* s) {

// assign the pointer to signal function

signal = s;

}

double DFTFunction::f(int m) {

#ifdef \_\_DEBUG

assert(signal); // must exist

#endif

// getting signal function samples

std::vector<double> samples = signal->getData();

// amount of samples

size\_t N = samples.size();

// real and imaginary parts

double re = 0, im = 0;

// for each sample

for (size\_t n = 0; n < N; n++) {

// summing

re += samples[n] \* cos(2 \* M\_PI \* m \* n / N);

im += samples[n] \* sin(-2 \* M\_PI \* m \* n / N);

}

// return the abs(X)

return sqrt(re \* re + im \* im);

}

void DFTFunction::calculate() {

// calculating scope calculating

double start = max(scale.x\_from, from);

double stop = min(scale.x\_to, to);

// memory reserving for points

points.resize(samples\_amount);

// for each point with the step

for (size\_t i = 0; i < samples\_amount; i++) {

// calculating f=Fд\*k/N

double x = 1. / samples\_step \* i / samples\_amount;

double y = f(i); // calculating y of the point abs(DFT)

if (is\_log) { // if log scale enabled

y = log10(abs(y)); // calculate log

}

points[i] = coordsToDot(x, y); // coord conversion

}

is\_calculated = true; // mark points as relevant

}

void DFTFunction::set\_signal(SignalFunction\* s) {

signal = s; // set signal

setNotCalculated(); // set not relevant

}

//

// coursework.h

//

#pragma once

#ifndef \_\_AFXWIN\_H\_\_

#error "включить pch.h до включения этого файла в PCH"

#endif

#include "resource.h" // including interface markup

/// <summary>

/// Class that defines the application - child of CWinApp

/// </summary>

class CcourseworkApp : public CWinApp

{

public:

/// <summary>

/// Constructor

/// </summary>

CcourseworkApp();

public:

/// <summary>

/// Initialization

/// </summary>

virtual BOOL InitInstance();

/// <summary>

/// message processing loop

/// </summary>

DECLARE\_MESSAGE\_MAP()

};

/// <summary>

/// the single instance of the class,

/// it represents the app

/// </summary>

extern CcourseworkApp theApp;

//

// coursework.cpp

//

// including project files

#include "pch.h"

#include "framework.h"

#include "coursework.h"

#include "courseworkDlg.h"

// definitions for the debug mode

#ifdef \_DEBUG

#define new DEBUG\_NEW

#endif

// start message processing loop for the app

BEGIN\_MESSAGE\_MAP(CcourseworkApp, CWinApp)

ON\_COMMAND(ID\_HELP, &CWinApp::OnHelp)

END\_MESSAGE\_MAP()

// end message processing loop

CcourseworkApp::CcourseworkApp() {

// restart manager supporting

m\_dwRestartManagerSupportFlags = AFX\_RESTART\_MANAGER\_SUPPORT\_RESTART;

}

// the single instance of the application class

// it represents the app

CcourseworkApp theApp;

BOOL CcourseworkApp::InitInstance() {

// data for dynamic lib of control elements connection

INITCOMMONCONTROLSEX InitCtrls;

// initialization of this data

InitCtrls.dwSize = sizeof(InitCtrls);

InitCtrls.dwICC = ICC\_WIN95\_CLASSES;

// ensure that the lib is connected

InitCommonControlsEx(&InitCtrls);

// calling InitInstance of the base class

CWinApp::InitInstance();

// creating shell manager

CShellManager\* pShellManager = new CShellManager;

// Активация визуального диспетчера "Классический Windows" для включения элементов управления MFC

CMFCVisualManager::SetDefaultManager(RUNTIME\_CLASS(CMFCVisualManagerWindows));

// setting the win registry key for application data

SetRegistryKey(\_T("SPbETU Chaminov Dmeatry 1183"));

// creating main dlg

CcourseworkDlg dlg;

// setting dlg as main dlg

m\_pMainWnd = &dlg;

// waiting till the dlg is closed

INT\_PTR nResponse = dlg.DoModal();

// response checking

if (nResponse == -1) {

TRACE(traceAppMsg, 0, "Предупреждение. Не удалось создать диалоговое окно, поэтому работа приложения неожиданно завершена.\n");

TRACE(traceAppMsg, 0, "Предупреждение. При использовании элементов управления MFC для диалогового окна невозможно #define \_AFX\_NO\_MFC\_CONTROLS\_IN\_DIALOGS.\n");

}

// safe deleting of the shell manager

if (pShellManager != nullptr) {

delete pShellManager;

}

#if !defined(\_AFXDLL) && !defined(\_AFX\_NO\_MFC\_CONTROLS\_IN\_DIALOGS)

ControlBarCleanUp();

#endif

// close the app (as main dlg is closed)

return FALSE;

}

//

// courseworkDlg.h

//

#pragma once

/// <summary>

/// Class that defines exterior and behavior

/// of the main dialog - child of CDialogEx

/// </summary>

class CcourseworkDlg : public CDialogEx

{

public:

/// <summary>

/// Standard constructor

/// </summary>

/// <param name="pParent">a pointer to the parent window</param>

CcourseworkDlg(CWnd\* pParent = nullptr);

// dialog id for building

#ifdef AFX\_DESIGN\_TIME

enum { IDD = IDD\_COURSEWORK\_DIALOG };

#endif

protected:

virtual void DoDataExchange(CDataExchange\* pDX); // поддержка DDX/DDV

protected:

/// <summary>

/// App Icon

/// </summary>

HICON m\_hIcon;

/// <summary>

/// Short task info

/// </summary>

CString theTask;

// functions of the message processing

virtual BOOL OnInitDialog();

afx\_msg void OnSysCommand(UINT nID, LPARAM lParam);

afx\_msg HCURSOR OnQueryDragIcon();

// message processing loop

DECLARE\_MESSAGE\_MAP()

public:

// functions of the message processing

afx\_msg void OnBnClickedOk();

afx\_msg void OnBnClickedTaskAboutBn();

afx\_msg void OnBnClickedCalculateButton();

};

//

// courseworkDlg.cpp

//

// including project files

#include "pch.h"

#include "framework.h"

#include "coursework.h"

#include "courseworkDlg.h"

#include "afxdialogex.h"

#include "Calculator.h"

// definitions for the debug mode

#ifdef \_DEBUG

#define new DEBUG\_NEW

#endif

//=====================CAboutDlg class===========================

/// <summary>

/// Class that defines dlg "about app"

/// </summary>

class CAboutDlg : public CDialogEx

{

public:

/// <summary>

/// Constructor

/// </summary>

CAboutDlg();

/// <summary>

/// Defines on dlg initialization behavior

/// </summary>

BOOL OnInitDialog();

// dialog id for building

#ifdef AFX\_DESIGN\_TIME

enum { IDD = IDD\_ABOUTBOX };

#endif

protected:

virtual void DoDataExchange(CDataExchange\* pDX); // поддержка DDX/DDV

protected:

/// message processing loop

DECLARE\_MESSAGE\_MAP()

};

CAboutDlg::CAboutDlg() : CDialogEx(IDD\_ABOUTBOX) { // call constructor of the base class

}

BOOL CAboutDlg::OnInitDialog() {

// set the title, commit index - the index of

// the last git commit (defined in string table coursework.rc')

CString

app\_v, // app version

app\_ci; // last commit index

app\_v.LoadStringW(IDS\_VERSION);

app\_ci.LoadStringW(IDS\_COMMIT);

CString title;

title.Format(L"О программе. Версия: %s-%s", app\_v, app\_ci);

SetWindowText(title);

return 0;

}

void CAboutDlg::DoDataExchange(CDataExchange\* pDX) {

CDialogEx::DoDataExchange(pDX); // use base data exchange

}

// message loop

BEGIN\_MESSAGE\_MAP(CAboutDlg, CDialogEx)

END\_MESSAGE\_MAP()

//===============================================================

CcourseworkDlg::CcourseworkDlg(CWnd\* pParent /\*=nullptr\*/)

: CDialogEx(IDD\_COURSEWORK\_DIALOG, pParent) //base constructor

{

// load task text from the string table

theTask.LoadStringW(IDS\_SHORT\_TASK\_INFO);

// set icon

m\_hIcon = AfxGetApp()->LoadIcon(ID\_LOGO\_ICO);

}

void CcourseworkDlg::DoDataExchange(CDataExchange\* pDX) {

CDialogEx::DoDataExchange(pDX); // base data exchange

DDX\_Text(pDX, IDC\_STATIC\_TASK, theTask); // data exchange between CStatic and CString

}

// message processing loop

BEGIN\_MESSAGE\_MAP(CcourseworkDlg, CDialogEx)

ON\_WM\_SYSCOMMAND()

ON\_WM\_PAINT()

ON\_WM\_QUERYDRAGICON()

ON\_BN\_CLICKED(IDOK, &CcourseworkDlg::OnBnClickedOk)

ON\_BN\_CLICKED(IDC\_Task\_About\_Bn, &CcourseworkDlg::OnBnClickedTaskAboutBn)

ON\_BN\_CLICKED(IDC\_CALCULATE\_BUTTON, &CcourseworkDlg::OnBnClickedCalculateButton)

END\_MESSAGE\_MAP()

// Обработчики сообщений CcourseworkDlg

BOOL CcourseworkDlg::OnInitDialog() {

CDialogEx::OnInitDialog(); // base OnInit

// set title

CString title;

title.LoadStringW(IDS\_VERSION);

title.Format(L"Курсовая - Чаминов Д. А. гр. 1183 v.%s", title);

SetWindowText(title);

// IDM\_ABOUTBOX должен быть в пределах системной команды.

ASSERT((IDM\_ABOUTBOX & 0xFFF0) == IDM\_ABOUTBOX);

ASSERT(IDM\_ABOUTBOX < 0xF000);

// adding "About" to the system menu

CMenu\* pSysMenu = GetSystemMenu(FALSE);

if (pSysMenu != nullptr) {

BOOL bNameValid;

CString strAboutMenu;

bNameValid = strAboutMenu.LoadString(IDS\_ABOUTBOX);

ASSERT(bNameValid);

if (!strAboutMenu.IsEmpty()) {

pSysMenu->AppendMenu(MF\_SEPARATOR);

pSysMenu->AppendMenu(MF\_STRING, IDM\_ABOUTBOX, strAboutMenu);

}

}

// set icon (32x32 - big icon)

SetIcon(m\_hIcon, TRUE);

return TRUE;

}

// catching system menu commands

void CcourseworkDlg::OnSysCommand(UINT nID, LPARAM lParam) {

if ((nID & 0xFFF0) == IDM\_ABOUTBOX) {

// create and show about dlg

CAboutDlg dlgAbout;

dlgAbout.DoModal();

} else {

// base action

CDialogEx::OnSysCommand(nID, lParam);

}

}

// Система вызывает эту функцию для получения отображения курсора при перемещении

// свернутого окна.

HCURSOR CcourseworkDlg::OnQueryDragIcon() {

return static\_cast<HCURSOR>(m\_hIcon);

}

// on button 'ok' clicked message

void CcourseworkDlg::OnBnClickedOk() {

CDialogEx::OnOK(); // base action

}

// on button 'about the task' clicked message

void CcourseworkDlg::OnBnClickedTaskAboutBn() {

// create string

CString str;

// load info string

str.LoadStringW(IDS\_MORE\_TASK\_INFO);

// create and show messagebox with theoretical information

AfxMessageBox(str, MB\_OK | MB\_ICONINFORMATION);

}

// on button 'calculate' has been clicked message

void CcourseworkDlg::OnBnClickedCalculateButton() {

// create and show calculator dlg

Calculator dlg;

dlg.DoModal();

}

//

// CMyGraph.h

//

#pragma once

#include <vector> // for the vector

#include <utility> // for std::pair

#include "MathFunction.h" // functions to draw

/// <summary>

/// Class represents CStatic element,

/// that will draw graphics of

/// MathFunction like objects

/// </summary>

class CMyGraph : public CStatic

{

// enables access to run-time class information

DECLARE\_DYNAMIC(CMyGraph)

protected:

/// <summary>

/// Draws background with axes, axes serifs and numbers on them

/// </summary>

/// <param name="dc">dc to use</param>

void drawBg(CDC& dc);

/// <summary>

/// indicates if background is done

/// </summary>

bool background\_calculated = false;

/// <summary>

/// background color

/// </summary>

COLORREF bg\_color = RGB(255, 255, 255);

/// <summary>

/// Bitmap for the background

/// </summary>

CBitmap bg\_bmp;

/// <summary>

/// Old object for bgdc

/// </summary>

HGDIOBJ old\_bmp = NULL;

/// <summary>

/// dc for drawing background

/// </summary>

CDC bgdc;

protected:

/// <summary>

/// Draws graph ON THE PAINTED BACKGROUND

/// </summary>

/// <param name="dc">dc to draw</param>

void drawGraph(CDC& dc);

/// <summary>

/// indicates if graph is done

/// </summary>

bool graph\_is\_done = false;

/// <summary>

/// Graph bitmap

/// </summary>

CBitmap graph;

/// <summary>

/// old object for the graph\_dc

/// </summary>

HGDIOBJ old\_g\_bmp = NULL;

/// <summary>

/// dc gor drawing graph

/// </summary>

CDC graph\_dc;

/// <summary>

/// 'smart' function that checkes is graph and bg drawed and paint it

/// recommended to use against of drawGraph and drawBg

/// </summary>

/// <param name="dc">dc to paint</param>

void draw(CDC& dc);

protected:

/// <summary>

/// indicates if animation in progress

/// </summary>

bool animation\_in\_process = false;

/// <summary>

/// animation phase divided by 2pi

/// </summary>

float current\_animation\_phase = 0;

/// <summary>

/// animation cyclic frequency divided by 2pi

/// </summary>

float animation\_speed = 5e-2;

/// <summary>

/// Function that returns dot coordinates in animation phase

/// </summary>

/// <param name="point">A point on the graph to recalculate</param>

/// <param name="r">rectangular border of the calculating region</param>

/// <returns>New point as CPoint</returns>

CPoint recalcDotForAnimation(CPoint point, const CRect& r);

public:

/// <summary>

/// shifting of the graph from the left bottom corner of the element

/// </summary>

POINT shift = { 80,30 };

/// <summary>

/// represents amount of serifs on axes

/// </summary>

struct { int x, y; } serifs = { 5,3 };

/// <summary>

/// length of serifs in the pixels

/// </summary>

int serifsize = 15;

/// <summary>

/// indicates if it is histogram

/// </summary>

bool is\_hist = false;

/// <summary>

/// widths of hist columns

/// </summary>

int hist\_width = 1;

/// <summary>

/// calculating step in pixels

/// </summary>

int step = 0;

/// <summary>

/// bounds of the calculating area (in math coords)

/// </summary>

struct

{

double from, to;

}scale\_x, scale\_y;

/// <summary>

/// idicates if there is logarithmic scale on the ordinate axis

/// </summary>

bool is\_log = false;

/// <summary>

/// Vector of the MathFunction like objects to draw

/// </summary>

std::vector<MathFunction\*> functions;

/// <summary>

/// default constructor

/// </summary>

CMyGraph();

/// <summary>

/// destructor

/// </summary>

virtual ~CMyGraph();

protected:

// for the message processing loop

DECLARE\_MESSAGE\_MAP()

public:

/// <summary>

/// defines on message WM\_PAINT behavior

/// </summary>

afx\_msg void OnPaint();

// setters

void setScale(double x\_from, double x\_to, double y\_from, double y\_to);

void setStep(int step);

void setRect(RECT r);

void setLog(bool b);

void setNotCalculated();

void setBgColor(COLORREF col);

void setAnimState(bool state);

void setColumnsCount(int N);

// getter

COLORREF getBgColor();

/// <summary>

/// Back conversion elemnts to real coords

/// </summary>

/// <param name="wx">x coord in the elements coords</param>

/// <param name="wy">y coord in the elements coords</param>

/// <param name="rect">rectangular border of the drawing region (of this element)</param>

/// <returns>pair of double - real (math) coords</returns>

std::pair<double, double> dotToCoords(int wx, int wy, CRect rect);

/// <summary>

/// Direct conversion real (math) to element (in pixels) coords

/// </summary>

/// <param name="x">x math coord</param>

/// <param name="y">y math coord</param>

/// <returns>element coords as POINT</returns>

POINT coordsToDot(double x, double y, CRect rect);

/// <summary>

/// Defines behavior on timer tick (it is not connected to the timer)

/// you should call it if you think that it's time to draw next frame

/// </summary>

/// <returns>returns if animation ended</returns>

bool timerTick();

};

//

// CMyGraph.cpp

//

// including project files

#include "pch.h"

#include "coursework.h"

#include "CMyGraph.h"

#include "BeautyLib.h"

// for run-time data

IMPLEMENT\_DYNAMIC(CMyGraph, CStatic)

std::pair<double, double> CMyGraph::dotToCoords(int wx, int wy, CRect r) {

// rect shifting to get the graph area

r.bottom -= shift.y;

r.left += shift.x;

double x, y;

// moving axis to ↑→ and shifting to the zero (in pixels)

x = wx - r.left;

y = r.bottom - wy;

// scaling

x \*= (scale\_x.to - scale\_x.from) / r.Width();

y \*= (scale\_y.to - scale\_y.from) / r.Height();

// shifting in math coords

x += scale\_x.from;

y += scale\_y.from;

// make pair and return it

return std::make\_pair(x, y);

}

POINT CMyGraph::coordsToDot(double x, double y, CRect rect) {

// shifting bounds to get calculating area

rect.left += shift.x;

rect.bottom -= shift.y;

// shifting in the math coords

x = x - scale\_x.from;

y = y - scale\_y.from;

// scaling

x = x \* rect.Width() / (scale\_x.to - scale\_x.from);

y = -(y \* rect.Height() / (scale\_y.to - scale\_y.from));

// shifting in win coords

x += rect.left;

y += rect.bottom;

// create point from x and y and return it

return POINT({ (long)round(x), (long)round(y) });

}

bool CMyGraph::timerTick() {

// increase current phase

current\_animation\_phase += animation\_speed;

// limit the phase

if (current\_animation\_phase > 1) { current\_animation\_phase = 1; }

// draw the frame with increased phase

RedrawWindow();

// return if the animation is stopped

return current\_animation\_phase == 1;

}

void CMyGraph::drawBg(CDC& dc) {

// getting client rect

CRect r;

GetClientRect(r);

// filling with bg color

dc.FillSolidRect(&r, bg\_color);

// set the pen and remember old object

CPen pen(PS\_SOLID, 2, RGB(0, 0, 0));

HGDIOBJ oldpen = dc.SelectObject(pen);

// draw axes

dc.MoveTo(shift.x, 0);

dc.LineTo(shift.x, r.bottom - shift.y);

dc.LineTo(r.Width(), r.bottom - shift.y);

// find font for the numbers and powers of 10

CFont font;

font.CreateFontW(18, 0, 0, 0, FW\_NORMAL, 0, 0, 0,

DEFAULT\_CHARSET, OUT\_RASTER\_PRECIS,

CLIP\_DEFAULT\_PRECIS, DEFAULT\_QUALITY,

DEFAULT\_PITCH || FF\_ROMAN, \_T("Times"));

CFont pfont;

pfont.CreateFontW(14, 0, 0, 0, FW\_NORMAL, 0, 0, 0,

DEFAULT\_CHARSET, OUT\_RASTER\_PRECIS,

CLIP\_DEFAULT\_PRECIS, DEFAULT\_QUALITY,

DEFAULT\_PITCH || FF\_ROMAN, \_T("Times"));

// select font and remember old obj

HGDIOBJ oldfont = dc.SelectObject(font);

// calculate step in real (math) coords

// (because back conv can give uncomfortable numbers)

double step = (scale\_x.to - scale\_x.from) / (serifs.x);

// for the serif

for (int i = 0; i <= serifs.x; i++) {

// calculate position (in math coords)

// (arithmetic progression)

double the\_x = i \* step + scale\_x.from;

// calculate position (in element coords)

CPoint sp = coordsToDot(the\_x, scale\_y.from, r);

// some moves to draw the serif with length=serifsize

sp.Offset(0, -serifsize / 2);

dc.MoveTo(sp);

sp.Offset(0, serifsize);

dc.LineTo(sp);

sp.Offset(0, 4);

// get exp notation of the number

auto number = getME(the\_x);

// default text

CString st = L"Что-то пошло не так!";

// if number ain't too long

if (abs(number.second) <= 3) {

// write the number to the string

st.Format(L"%.4f", the\_x);

// set text align

dc.SetTextAlign(TA\_CENTER);

//show the text

dc.TextOutW(sp.x, sp.y, st);

} else {// if number is large

// offset the point for text

sp.Offset(30, 0);

// write the number to the string

st.Format(L"%.4f\u221910", number.first);

// set text align

dc.SetTextAlign(TA\_RIGHT);

// show the text

dc.TextOutW(sp.x, sp.y, st);

// offset point for exponent label

sp.Offset(1, -5);

// set text align

dc.SetTextAlign(TA\_LEFT);

// select font for exponent

dc.SelectObject(pfont);

// write exp to the string

st.Format(L"%d", number.second);

// show the text

dc.TextOutW(sp.x, sp.y, st);

// select usual font

dc.SelectObject(font);

}

}

// if isn't log scale

if (!is\_log) { // same as with x axis

step = (scale\_y.to - scale\_y.from) / (serifs.y);

for (int i = 0; i <= serifs.y; i++) {

double the\_y = i \* step + scale\_y.from;

CPoint sp = coordsToDot(scale\_x.from, the\_y, r);

sp.Offset(serifsize / 2, 0);

dc.MoveTo(sp);

sp.Offset(-serifsize, 0);

dc.LineTo(sp);

sp.Offset(0, -3);

CString st = L"Что-то пошло не так!";

auto number = getME(the\_y);

// if number ain't too long

if (abs(number.second) <= 3) {

st.Format(L"%.4f", the\_y);

dc.SetTextAlign(TA\_RIGHT);

dc.TextOutW(sp.x, sp.y, st);

} else {// if number is large

sp.Offset(-10, 0);

st.Format(L"%.4f\u221910", number.first);

dc.SetTextAlign(TA\_RIGHT);

dc.TextOutW(sp.x, sp.y, st);

sp.Offset(1, -5);

dc.SetTextAlign(TA\_LEFT);

dc.SelectObject(pfont);

st.Format(L"%d", number.second);

dc.TextOutW(sp.x, sp.y, st);

dc.SelectObject(font);

}

}

} else { // log scale

// progression with powers of 10

int start\_power = floor(scale\_y.from),

stop\_power = ceil(scale\_y.to),

step\_power = ceil(((double)stop\_power - start\_power) / (serifs.y));

for (int i = 0; i <= serifs.y; i++) { // for power of 10

// calculate the power

int current\_power = i \* step\_power + start\_power;

// calculate the y of the serif (in math coords)

double the\_y = pow(10, (double)current\_power);

// calculate in element coords

CPoint sp = coordsToDot(scale\_x.from, log10(the\_y), r);

// moves to draw serif

sp.Offset(serifsize / 2, 0);

dc.MoveTo(sp);

sp.Offset(-serifsize, 0);

dc.LineTo(sp);

sp.Offset(-40, -5);

// write '10'

CString st = L"10";

dc.SetTextAlign(TA\_TOP);

dc.TextOutW(sp.x, sp.y, st);

// offset, change the font

sp.Offset(16, -5);

dc.SelectObject(pfont);

// write the power

st.Format(L"%d", current\_power);

dc.SetTextAlign(TA\_LEFT);

dc.TextOutW(sp.x, sp.y, st);

// select normal font

dc.SelectObject(font);

}

}

// select old objects (get to the start state)

dc.SelectObject(oldfont);

dc.SelectObject(oldpen);

// remember background as painted

background\_calculated = true;

}

void CMyGraph::drawGraph(CDC& dc) {

// create pen, select it and remember old obj

CPen gr(BS\_SOLID, 1, RGB(0, 0, 0));

HGDIOBJ oldpen = dc.SelectObject(gr);

// get the client area borders

CRect r;

GetWindowRect(&r);

r = { 0,0,r.right - r.left,r.bottom - r.top };

// copy bg from the bitmap

dc.BitBlt(0, 0, r.Width(), r.Height(), &bgdc, 0, 0, SRCCOPY);

// calculating area

CRect rforf(r);

rforf.left += shift.x;

rforf.bottom -= shift.y;

// rgn for the clipping graph

CRgn rgnf;

rgnf.CreateRectRgn(rforf.left, rforf.top, rforf.right, rforf.bottom);

dc.SelectClipRgn(&rgnf);

// for each MathFunction like object in this graph

for (MathFunction\* f : functions) {

// set that it's the first point

bool is\_first = true;

// create pen for this function, and choose it

CPen gr(BS\_SOLID, 1, f->color);

dc.SelectObject(gr);

// get dots vector

auto dots = f->get\_points();

//amount of dots

size\_t dots\_count = dots.size();

for (size\_t i = 0; i < dots\_count; i++) {

// current dot from the dots

CPoint dot(dots[i]);

if (is\_hist) {

if (animation\_in\_process) {

// recalc coords according to phase

dot = recalcDotForAnimation(dot, rforf);

}

// hist column calculating and drawing

RECT rect = { dot.x,dot.y,dot.x + hist\_width,rforf.bottom };

dc.FillSolidRect(&rect, f->color);

} else {// not hist

if (is\_first) {

dc.MoveTo(dot); //just move here not draw the line

is\_first = false; // set not first

} else {

dc.LineTo(dot); // draw the line

}

// if current phase means to stop drawing

if ( animation\_in\_process &&

i > dots\_count \* pow(dots\_count, current\_animation\_phase - 1)) {

break; // stop drawing

}

}

}

}

// restore dc state

dc.SelectObject(oldpen);

CRgn rgn;

rgn.CreateRectRgn(r.left, r.top, r.right, r.bottom);

dc.SelectClipRgn(&rgn);

}

void CMyGraph::draw(CDC& dc) {

// getting client area

CRect r;

GetClientRect(r);

// if graph isn't done

if (!graph\_is\_done) {

// if bgdc doesn't exist

if (!bgdc) {

// create dc and prepare it for drawing in bitmap

bgdc.CreateCompatibleDC(&dc);

bg\_bmp.CreateCompatibleBitmap(&dc, r.Width(), r.Height());

old\_bmp = bgdc.SelectObject(bg\_bmp);

}

// if bg isn't done

if (!background\_calculated) {

// draw bg

drawBg(bgdc);

}

// if graph\_dc doesn't exist

if (!graph\_dc) {

// create dc and prepare it for drawing in bitmap

graph\_dc.CreateCompatibleDC(&dc);

graph.CreateCompatibleBitmap(&dc, r.Width(), r.Height());

old\_g\_bmp = graph\_dc.SelectObject(graph);

}

//draw the graph

drawGraph(graph\_dc);

}

// copy to passed as param dc

dc.BitBlt(0, 0, r.Width(), r.Height(), &graph\_dc, 0, 0, SRCCOPY);

}

CPoint CMyGraph::recalcDotForAnimation(CPoint p, const CRect& r) {

// shift and reverse

double y = r.bottom - p.y;

// protection from small numbers

if (y < 1) { p.y = r.bottom - y \* current\_animation\_phase; return p; }

//recalcing

p.y = r.bottom - y \* pow(y, current\_animation\_phase - 1);

return p;

}

CMyGraph::CMyGraph() // default constructors call

:CStatic(),

scale\_x(), scale\_y() {

}

CMyGraph::~CMyGraph() {

// dc's state restoring

bgdc.SelectObject(old\_bmp);

graph\_dc.SelectObject(old\_g\_bmp);

}

// message processing loop

BEGIN\_MESSAGE\_MAP(CMyGraph, CStatic)

ON\_WM\_PAINT()

END\_MESSAGE\_MAP()

void CMyGraph::OnPaint() {

// just draw it

CPaintDC dc(this);

draw(dc);

}

// setters,

// they set states (graph\_is done, background calculated)

// according to changes

void CMyGraph::setScale(double x\_from, double x\_to, double y\_from, double y\_to) {

if (x\_from != scale\_x.from || x\_to != scale\_x.to || y\_from != scale\_y.from || y\_to != scale\_y.to) {

background\_calculated = false;

graph\_is\_done = false;

}

this->scale\_x = { x\_from,x\_to };

this->scale\_y = { y\_from,y\_to };

for (MathFunction\* f : functions) {

f->setScale(x\_from, x\_to, y\_from, y\_to);

}

}

void CMyGraph::setStep(int step\_) {

if (this->step != step) {

graph\_is\_done = false;

}

RECT r;

GetClientRect(&r);

double step = step\_ \* (scale\_x.to - scale\_x.from) / (r.right - r.left);

this->step = step\_;

for (MathFunction\* f : functions) {

f->setStep(step);

}

}

void CMyGraph::setRect(RECT r) {

r.bottom -= shift.y;

r.left += shift.x;

for (MathFunction\* f : functions) {

f->setRect(r);

}

}

void CMyGraph::setLog(bool b) {

if (b != is\_log) {

graph\_is\_done = false;

background\_calculated = false;

}

this->is\_log = b;

for (MathFunction\* f : functions) {

f->setLog(b);

}

}

void CMyGraph::setNotCalculated() {

graph\_is\_done = false;

background\_calculated = false;

for (MathFunction\* f : functions) {

f->setNotCalculated();

}

}

void CMyGraph::setBgColor(COLORREF col) {

if (col != bg\_color) {

graph\_is\_done = false;

background\_calculated = false;

bg\_color = col;

}

}

void CMyGraph::setAnimState(bool state) {

current\_animation\_phase = 0;

animation\_in\_process = state;

}

void CMyGraph::setColumnsCount(int N) {

CRect r;

GetClientRect(r);

int w = ceil((float)(r.Width() - shift.x) / N);

if (w == hist\_width) { return; }

graph\_is\_done = false;

hist\_width = w;

}

// just a getter

COLORREF CMyGraph::getBgColor() {

return bg\_color;

}

//

// Calculator.h

//

#pragma once

#include "CMyGraph.h" // for graphs

/// <summary>

/// Defines dlg with calculating

/// </summary>

class Calculator : public CDialogEx

{

// timer settings for animation

const static uint64\_t timer\_id = 12345654;

const static uint64\_t timer\_delay = 40;

// enables access to run-time class information

DECLARE\_DYNAMIC(Calculator)

public:

/// <summary>

/// Standard constructor

/// </summary>

/// <param name="pParent">a pointer to the parent window</param>

Calculator(CWnd\* pParent = nullptr);

/// <summary>

/// std destructor

/// </summary>

virtual ~Calculator();

/// <summary>

/// defines init behavior

/// </summary>

BOOL OnInitDialog();

/// <summary>

/// Updates params in the functions and graphs

/// with info from controls

/// </summary>

void UpdateCalculatorParams();

protected:

// MathFunction objects for graphs:

/// <summary>

/// the object represents signal function

/// </summary>

SignalFunction signal;

/// <summary>

/// the object represents DFT

/// </summary>

DFTFunction dft;

// Graphs:

/// <summary>

/// signal graph painter

/// </summary>

CMyGraph graph\_signal;

/// <summary>

/// DFT hist painter

/// </summary>

CMyGraph graph\_DFT;

// Controls:

CSliderCtrl slider\_samples;

CSliderCtrl slider\_step;

CEdit edit\_a;

CEdit edit\_m;

CEdit edit\_f;

CEdit edit\_x\_f;

CEdit edit\_x\_t;

CEdit edit\_y\_f;

CEdit edit\_y\_t;

CEdit edit\_y\_dft\_f;

CEdit edit\_y\_dft\_t;

CButton cb\_is\_dft\_log;

CButton cb\_is\_log;

CButton cb\_anim;

CMFCColorButton bg\_cp;

CMFCColorButton signal\_cp;

CMFCColorButton dft\_cp;

// static text with samples amount

CStatic text\_slider\_samples;

// dialog id for building

#ifdef AFX\_DESIGN\_TIME

enum { IDD = IDD\_Calculator };

#endif

protected:

virtual void DoDataExchange(CDataExchange\* pDX); // поддержка DDX/DDV

// message processing loop

DECLARE\_MESSAGE\_MAP()

protected:

/// <summary>

/// resets color pickers

/// </summary>

void ResetColorPickers();

/// <summary>

/// resets all data

/// </summary>

void ResetInputData();

public:

// messages processing functions

afx\_msg void OnBnClickedButtonUpdate();

afx\_msg void OnBnClickedButtonreset();

afx\_msg void OnBnClickedButtonSaveGr();

afx\_msg void OnBnClickedCheckislogscale();

afx\_msg void OnBnClickedCheckislogscale2();

afx\_msg void OnMouseMove(UINT nFlags, CPoint point);

afx\_msg void OnTimer(UINT\_PTR nIDEvent);

afx\_msg void OnSysCommand(UINT nID, LPARAM lParam);

afx\_msg void OnHScroll(UINT nSBCode, UINT nPos, CScrollBar\* pScrollBar);

};

//

// Calculator.cpp

//

// including project files

#include "pch.h"

#include "coursework.h"

#include "BeautyLib.h"

#include "Calculator.h"

#include "afxdialogex.h"

#include <initguid.h> // for guids

#include <cassert> // for asserts

#include <cmath> // for math functions as log, pow, etc

// guids for image codecs

DEFINE\_GUID(ImageFormatBMP, 0xb96b3cab, 0x0728, 0x11d3, 0x9d,

0x7b, 0x00, 0x00, 0xf8, 0x1e, 0xf3, 0x2e);

DEFINE\_GUID(ImageFormatJPEG, 0xb96b3cae, 0x0728, 0x11d3, 0x9d,

0x7b, 0x00, 0x00, 0xf8, 0x1e, 0xf3, 0x2e);

DEFINE\_GUID(ImageFormatPNG, 0xb96b3caf, 0x0728, 0x11d3, 0x9d,

0x7b, 0x00, 0x00, 0xf8, 0x1e, 0xf3, 0x2e);

DEFINE\_GUID(ImageFormatGIF, 0xb96b3cb0, 0x0728, 0x11d3, 0x9d,

0x7b, 0x00, 0x00, 0xf8, 0x1e, 0xf3, 0x2e);

// for runtime data

IMPLEMENT\_DYNAMIC(Calculator, CDialogEx)

Calculator::Calculator(CWnd\* pParent /\*=nullptr\*/)

: CDialogEx(IDD\_Calculator, pParent), // base constructor

signal(), // SignalFunction default constructor

dft() { // DFTFunction default constructor

dft.set\_signal(&signal); // Set signal function for dft object

}

Calculator::~Calculator() { // default destructor

}

BOOL Calculator::OnInitDialog() {

// control elements connection

graph\_signal.SubclassDlgItem(IDC\_STATIC\_graph, this);

graph\_DFT.SubclassDlgItem(IDC\_STATIC\_graph2, this);

bg\_cp.SubclassDlgItem(IDC\_MFCCOLORBUTTON\_BG, this);

signal\_cp.SubclassDlgItem(IDC\_MFCCOLORBUTTON\_SIGNAL, this);

dft\_cp.SubclassDlgItem(IDC\_MFCCOLORBUTTON\_DCF, this);

slider\_step.SubclassDlgItem(IDC\_SLIDER\_STEP, this);

slider\_samples.SubclassDlgItem(IDC\_SLIDER\_SAMPLES, this);

text\_slider\_samples.SubclassDlgItem(IDC\_STATIC\_SAMPLES, this);

cb\_is\_log.SubclassDlgItem(IDC\_CHECK\_is\_log\_scale, this);

cb\_is\_dft\_log.SubclassDlgItem(IDC\_CHECK\_is\_log\_scale2, this);

cb\_anim.SubclassDlgItem(IDC\_CHECK\_ANIM, this);

edit\_a.SubclassDlgItem(IDC\_EDIT\_param\_a, this);

edit\_m.SubclassDlgItem(IDC\_EDIT\_param\_m, this);

edit\_f.SubclassDlgItem(IDC\_EDIT\_param\_f, this);

edit\_x\_f.SubclassDlgItem(IDC\_EDIT\_xscale\_from, this);

edit\_x\_t.SubclassDlgItem(IDC\_EDIT\_xscale\_to, this);

edit\_y\_f.SubclassDlgItem(IDC\_EDIT\_yscale\_from, this);

edit\_y\_t.SubclassDlgItem(IDC\_EDIT\_yscale\_to, this);

edit\_y\_dft\_t.SubclassDlgItem(IDC\_EDIT\_yscale\_to2, this);

edit\_y\_dft\_f.SubclassDlgItem(IDC\_EDIT\_yscale\_from2, this);

// setting range of the slider

slider\_step.SetRangeMin(1);

slider\_step.SetRangeMax(10);

slider\_samples.SetRangeMin(100);

slider\_samples.SetRangeMax(1000);

// turn animation on

cb\_anim.SetCheck(1);

// find element

CWnd\* p = GetDlgItem(IDC\_STATIC\_signal);

//set default text

if (p) { p->SetWindowTextW(\_T("x(t) = a\u2219sin(2\u03c0(f + mt)\u2219t")); }

// set focus to the first input element

edit\_a.SetFocus();

// set default data for controls

ResetInputData();

ResetColorPickers();

// set default for the MathFunction s

// and add it to graph drawer

signal.setDefinitionScope( -std::numeric\_limits<double>::infinity(),

std::numeric\_limits<double>::infinity()

);

graph\_signal.functions.push\_back(&signal);

graph\_signal.is\_hist = false;

dft.setDefinitionScope( -std::numeric\_limits<double>::infinity(),

std::numeric\_limits<double>::infinity()

);

graph\_DFT.functions.push\_back(&dft);

graph\_DFT.is\_hist = true;

// adding 'save' to the sys menu

CMenu\* pSysMenu = GetSystemMenu(FALSE);

if (pSysMenu != nullptr) {

CString strSaveMenu = L"Сохранить как";

if (!strSaveMenu.IsEmpty()) {

pSysMenu->AppendMenu(MF\_SEPARATOR);

pSysMenu->AppendMenu(MF\_STRING, IDD\_Calculator, strSaveMenu);

}

}

return 0;

}

void Calculator::UpdateCalculatorParams() {

// getting data from input as strings

// and converting string to doubles

CString a\_s;

edit\_a.GetWindowTextW(a\_s);

double a = \_wtof(a\_s);

CString m\_s;

edit\_m.GetWindowTextW(m\_s);

double m = \_wtof(m\_s);

CString f\_s;

edit\_f.GetWindowTextW(f\_s);

double f = \_wtof(f\_s);

CString x\_from\_s;

edit\_x\_f.GetWindowTextW(x\_from\_s);

double x\_from = \_wtof(x\_from\_s);

CString x\_to\_s;

edit\_x\_t.GetWindowTextW(x\_to\_s);

double x\_to = \_wtof(x\_to\_s);

CString y\_from\_s;

edit\_y\_f.GetWindowTextW(y\_from\_s);

double y\_from = \_wtof(y\_from\_s);

CString y\_to\_s;

edit\_y\_t.GetWindowTextW(y\_to\_s);

double y\_to = \_wtof(y\_to\_s);

CString y\_from2\_s;

edit\_y\_dft\_f.GetWindowTextW(y\_from2\_s);

double y\_from2 = \_wtof(y\_from2\_s);

CString y\_to2\_s;

edit\_y\_dft\_t.GetWindowTextW(y\_to2\_s);

double y\_to2 = \_wtof(y\_to2\_s);

// if lower bound is more then high one

if (x\_from >= x\_to || y\_from > y\_to) {

// send message to the user

AfxMessageBox(\_T("Невозможный масштаб"), MB\_OK | MB\_ICONERROR);

// end (do not update info)

return;

}

// if there is no empty fields

if (a\_s != "" && m\_s != "" && f\_s != "" && x\_from\_s != "" && x\_to\_s != "" && y\_from\_s != "" && y\_to\_s != "") {

// coefs

f \*= 1e7;

m \*= 1e7;

x\_from \*= 1e-4;

x\_to \*= 1e-4;

// set signal params

signal.set\_a(a); signal.set\_f(f); signal.set\_m(m);

dft.set\_a(a); dft.set\_f(f); dft.set\_m(m);

// if log (signal function)

if (cb\_is\_log.GetCheck() == 1) {

if (y\_from <= 0) {

// send message to the user

AfxMessageBox(\_T("Неверные границы логарифмического масштаба"),

MB\_OK | MB\_ICONERROR);

return;

}

// set scale

graph\_signal.setScale(x\_from, x\_to, log10(y\_from), log10(y\_to));

} else {

// set scale

graph\_signal.setScale(x\_from, x\_to, y\_from, y\_to);

}

double

x\_from\_dft = 0,

x\_to\_dft = 1e7;

// same but with DFT

if (cb\_is\_dft\_log.GetCheck() == 1) {

if (y\_from2 < 0) {

AfxMessageBox(\_T("Отрицательные границы логарифмического масштаба"), MB\_OK | MB\_ICONERROR);

return;

}

graph\_DFT.setScale(x\_from\_dft, x\_to\_dft, log10(y\_from2), log10(y\_to2));

} else {

graph\_DFT.setScale(x\_from\_dft, x\_to\_dft, y\_from2, y\_to2);

}

// declare rectangular

CRect r;

// write client region to it

graph\_signal.GetClientRect(r);

// get graph step (amount of pixels between two nearest

// graph points along the abscissa axis)

int step = slider\_step.GetPos();

// setting the step and region

graph\_signal.setStep(step);

graph\_DFT.setStep(step);

graph\_signal.setRect(r);

graph\_DFT.GetClientRect(r);

graph\_DFT.setRect(r);

// setting samples amount

int N = slider\_samples.GetPos();

dft.set\_samples\_amount(N);

signal.set\_samples\_amount(N);

graph\_DFT.setColumnsCount(N);

// set log scale

graph\_signal.setLog(cb\_is\_log.GetCheck() == 1);

graph\_DFT.setLog(cb\_is\_dft\_log.GetCheck() == 1);

// set colors

graph\_signal.setBgColor(bg\_cp.GetColor());

graph\_DFT.setBgColor(bg\_cp.GetColor());

graph\_signal.functions[0]->setColor(signal\_cp.GetColor());

graph\_DFT.functions[0]->setColor(dft\_cp.GetColor());

// find the static element and put formatted string to it

CWnd\* p;

p = GetDlgItem(IDC\_STATIC\_signal);

if (p) {

CString signal;

signal.Format(L"x(t) = %s\u2219sin(2π(%s + %s\u2219t)\u2219t)",

beautifulRepresentation(a, 2, 2),

beautifulRepresentation(f, 2, 2),

beautifulRepresentation(m, 2, 2)

);

p->SetWindowTextW(signal);

}

} else { // empty elements were found

// find the static element and put formetted string to it

CWnd\* p = GetDlgItem(IDC\_STATIC\_signal);

if (p) { p->SetWindowTextW(\_T("x(t) = a\u2219sin(2\u03c0(f + mt)\u2219t")); }

// send message to the user

AfxMessageBox(\_T("Недостаточно параметров"), MB\_OK | MB\_ICONERROR);

}

}

void Calculator::DoDataExchange(CDataExchange\* pDX) {

CDialogEx::DoDataExchange(pDX); // base data exchange

// controls data exchange

DDX\_Control(pDX, IDC\_STATIC\_graph, graph\_signal);

DDX\_Control(pDX, IDC\_SLIDER\_STEP, slider\_step);

DDX\_Control(pDX, IDC\_CHECK\_is\_log\_scale, cb\_is\_log);

DDX\_Control(pDX, IDC\_EDIT\_param\_a, edit\_a);

DDX\_Control(pDX, IDC\_EDIT\_param\_m, edit\_m);

DDX\_Control(pDX, IDC\_EDIT\_param\_f, edit\_f);

DDX\_Control(pDX, IDC\_EDIT\_xscale\_from, edit\_x\_f);

DDX\_Control(pDX, IDC\_EDIT\_xscale\_to, edit\_x\_t);

DDX\_Control(pDX, IDC\_EDIT\_yscale\_from, edit\_y\_f);

DDX\_Control(pDX, IDC\_EDIT\_yscale\_to, edit\_y\_t);

DDX\_Control(pDX, IDC\_STATIC\_graph2, graph\_DFT);

DDX\_Control(pDX, IDC\_CHECK\_is\_log\_scale2, cb\_is\_dft\_log);

DDX\_Control(pDX, IDC\_EDIT\_yscale\_from2, edit\_y\_dft\_f);

DDX\_Control(pDX, IDC\_EDIT\_yscale\_to2, edit\_y\_dft\_t);

DDX\_Control(pDX, IDC\_CHECK\_ANIM, cb\_anim);

DDX\_Control(pDX, IDC\_SLIDER\_SAMPLES, slider\_samples);

DDX\_Control(pDX, IDC\_STATIC\_SAMPLES, text\_slider\_samples);

}

// messages processing loop

BEGIN\_MESSAGE\_MAP(Calculator, CDialogEx)

ON\_BN\_CLICKED(IDC\_BUTTON\_UPDATE\_graph, &Calculator::OnBnClickedButtonUpdate)

ON\_BN\_CLICKED(IDC\_BUTTON\_reset, &Calculator::OnBnClickedButtonreset)

ON\_BN\_CLICKED(IDC\_BUTTON\_SAVE\_GR, &Calculator::OnBnClickedButtonSaveGr)

ON\_BN\_CLICKED(IDC\_CHECK\_is\_log\_scale, &Calculator::OnBnClickedCheckislogscale)

ON\_BN\_CLICKED(IDC\_CHECK\_is\_log\_scale2, &Calculator::OnBnClickedCheckislogscale2)

ON\_WM\_MOUSEMOVE()

ON\_WM\_TIMER()

ON\_WM\_SYSCOMMAND()

ON\_WM\_HSCROLL()

END\_MESSAGE\_MAP()

void Calculator::OnBnClickedButtonUpdate() {

UpdateCalculatorParams(); // update data

if (cb\_anim.GetCheck() == 1) { // if do animate

graph\_signal.setAnimState(true); // set anim state

graph\_DFT.setAnimState(true); // same

// set timer

SetTimer(timer\_id, timer\_delay, nullptr);

}

// draw graphs

graph\_signal.RedrawWindow();

graph\_DFT.RedrawWindow();

}

void Calculator::OnBnClickedButtonreset() {

ResetInputData(); // reset input data

ResetColorPickers();// reset color pickers

}

void Calculator::ResetColorPickers() {

// sets default colors

bg\_cp.SetColor(RGB(0xff, 0xfb, 0xf0));

signal\_cp.SetColor(RGB(0x80, 0, 0));

dft\_cp.SetColor(RGB(0, 0, 0x80));

}

void Calculator::ResetInputData() {

// sets default input data

edit\_a.SetWindowTextW(\_T("1"));

edit\_m.SetWindowTextW(\_T("1"));

edit\_f.SetWindowTextW(\_T("0.2"));

edit\_x\_f.SetWindowTextW(\_T("0"));

edit\_x\_t.SetWindowTextW(\_T("1"));

edit\_y\_f.SetWindowTextW(\_T("-1.5"));

edit\_y\_t.SetWindowTextW(\_T("1.5"));

edit\_y\_dft\_f.SetWindowTextW(\_T("0.01"));

edit\_y\_dft\_t.SetWindowTextW(\_T("10000"));

text\_slider\_samples.SetWindowTextW(L"Число отсчётов: 500");

cb\_is\_dft\_log.SetCheck(1);

cb\_is\_log.SetCheck(0);

slider\_step.SetPos(1);

slider\_samples.SetPos(500);

}

void Calculator::OnBnClickedButtonSaveGr() {

// Areas of the 1st pic, 2nd pic and result pic

CRect r1, r2, rr;

// offset between pics

const int offset = 20;

// getting areas

graph\_signal.GetClientRect(r1);

graph\_DFT.GetClientRect(r2);

//calculating the area

rr = { 0,0,r1.Width(),r1.Height() + r2.Height() + offset };

CBitmap bmp; // bitmap for the picture

CDC bmdc; // dc for drawing

// existing dc to create own bmdc

CWindowDC wdc(&graph\_signal);

// creating dc and bitmap

bmdc.CreateCompatibleDC(&wdc);

bmp.CreateCompatibleBitmap(&wdc, rr.Width(), rr.Height());

// selecting bitmap

HGDIOBJ olddc = bmdc.SelectObject(&bmp);

// copying pic from CWindowDC(&graph\_signal)

bmdc.BitBlt(0, 0, r1.Width(), r1.Height(), &wdc, 0, 0, SRCCOPY);

// filling space between pics

bmdc.FillSolidRect(

CRect({ 0,r1.Height(),r1.Width(),r1.Height() + offset }),

graph\_signal.getBgColor()

);

// copying pic from CWindowDC(&graph\_DFT)

bmdc.BitBlt( 0, r1.Height() + offset, r2.Width(), r2.Height(),

&CWindowDC(&graph\_DFT), 0, 0, SRCCOPY);

// dc restoring

bmdc.SelectObject(olddc);

// creating file dialog

CFileDialog dlg(FALSE, \_T(".bmp"), L"График ДПФ.bmp", OFN\_OVERWRITEPROMPT,

L"BMP Files (\*.bmp)|\*.bmp|PNG Files (\*.png)|\*.png| GIF Files(\*.gif)"

L"|\*.gif| JPG Files (\*.jpg)|\*.jpg|All Files (\*.\*)|\*.\*||");

dlg.DoModal(); // show the dlg

// getting file path

CString path = dlg.GetOFN().lpstrFile;

// find the last dot in the path

short i = path.ReverseFind(L'.');

// file extension

CString extension;

if (i == -1) {

// there is no dot

path += ".bmp"; // default

extension = \_T(".bmp"); // default

} else {

// calculating length of the

// path after last dot

i = path.GetLength() - i;

// slicing the path string to get extension

extension = path.Right(i);

// lowering the extension (from ".BMP" to ".bmp", etc)

extension.MakeLower();

}

// Creating CImage according to the bitmap 'bmp'

CImage img;

img.Attach(HBITMAP(bmp));

HRESULT saving; // place for the error code

// switch-case like block (choosing the file format)

if (extension == \_T(".jpg")) {

saving = img.Save(path, ImageFormatJPEG);

} else if (extension == \_T(".png")) {

saving = img.Save(path, ImageFormatPNG);

} else if (extension == \_T(".gif")) {

saving = img.Save(path, ImageFormatGIF);

} else if (extension == \_T(".bmp")) {

saving = img.Save(path, ImageFormatBMP);

} else { // default

// if not found correct img formats

// add ".bmp" to the end of path

// file.myexten -> file.myexten.bmp

// to avoid mistakes with replacing extension

path += ".bmp";

saving = img.Save(path, ImageFormatBMP);

}

if (FAILED(saving)) { // errors detected

// tell to user

AfxMessageBox(\_T("При сохранении файла что-то пошло не так"), MB\_OK | MB\_ICONERROR);

}

}

void Calculator::OnBnClickedCheckislogscale() {

CString fromStr, toStr; // strings for scale data

edit\_y\_f.GetWindowTextW(fromStr); // getting data

edit\_y\_t.GetWindowTextW(toStr); // same

double

from = \_wtof(fromStr), // converting to the double

to = \_wtof(toStr);

if (cb\_is\_log.GetCheck() == 1) {

//turned to log

// exponentiation

fromStr.Format(L"%f", pow(10, from));

edit\_y\_f.SetWindowTextW(fromStr);

toStr.Format(L"%f", pow(10, to));

edit\_y\_t.SetWindowTextW(toStr);

} else {

//turned to the normal

// logarithm

if (from <= 0 || to <= 0) { return; }

fromStr.Format(L"%f", log10(from));

edit\_y\_f.SetWindowTextW(fromStr);

toStr.Format(L"%f", log10(to));

edit\_y\_t.SetWindowTextW(toStr);

}

}

void Calculator::OnBnClickedCheckislogscale2() {

// same but with another controls

CString fromStr, toStr;

edit\_y\_dft\_f.GetWindowTextW(fromStr);

edit\_y\_dft\_t.GetWindowTextW(toStr);

double

from = \_wtof(fromStr),

to = \_wtof(toStr);

if (cb\_is\_dft\_log.GetCheck() == 1) { // turned to log

// exponentiating

fromStr.Format(L"%f", pow(10, from));

edit\_y\_dft\_f.SetWindowTextW(fromStr);

toStr.Format(L"%f", pow(10, to));

edit\_y\_dft\_t.SetWindowTextW(toStr);

} else { // turned to the normal

if (from <= 0 || to <= 0) { return; }

fromStr.Format(L"%f", log10(from));

edit\_y\_dft\_f.SetWindowTextW(fromStr);

toStr.Format(L"%f", log10(to));

edit\_y\_dft\_t.SetWindowTextW(toStr);

}

}

void Calculator::OnMouseMove(UINT nFlags, CPoint point) {

CPoint p(point); // copy coords

ClientToScreen(&p); // coord conversion

// find the static text object

CWnd\* pWnd = nullptr;

pWnd = GetDlgItem(IDC\_STATIC\_CPOS);

// default text

CString s = L"Для трассировки наведите курсор на график";

assert(pWnd);

// get client rects of the graphs

CRect rs;

graph\_signal.GetWindowRect(rs);

CRect rd;

graph\_DFT.GetWindowRect(rd);

// if point in the 1st graph

if (rs.PtInRect(p)) {

// move the point

p.Offset(-rs.left, -rs.top);

// rect shifting

rs = { 0,0,rs.Width(),rs.Height() };

// coord back conversion

auto dot = graph\_signal.dotToCoords(p.x, p.y, rs);

// format string according to scale type

if (!graph\_signal.is\_log) {

s.Format(L"t:%s; x:%s",

beautifulRepresentation(dot.first, 2, 4),

beautifulRepresentation(dot.second, 2, 4));

} else {

s.Format(L"t:%s; x:%s",

beautifulRepresentation(dot.first, 2, 4),

beautifulRepresentation(pow(10, dot.second), 2, 4));

}

} else if (rd.PtInRect(p)) { //same but with 2nd graph

p.Offset(-rd.left, -rd.top);

rd = { 0,0,rd.Width(),rd.Height() };

auto dot = graph\_DFT.dotToCoords(p.x, p.y, rd);

if (!graph\_DFT.is\_log) {

s.Format(L"\u03c9:%s; X:%s",

beautifulRepresentation(dot.first, 2, 4),

beautifulRepresentation(dot.second, 2, 4));

} else {

s.Format(L"\u03c9:%s; X:%s",

beautifulRepresentation(dot.first, 2, 4),

beautifulRepresentation(pow(10, dot.second), 2, 4));

}

}//else{;}

// set formatted text

pWnd->SetWindowTextW(s);

// base message processing

CDialogEx::OnMouseMove(nFlags, point);

}

void Calculator::OnTimer(UINT\_PTR nIDEvent) {

if (nIDEvent == timer\_id) { // if it is animation timer

// tick and check is it the last tick

// in the animation for both graphs

bool end = graph\_signal.timerTick();

end = graph\_DFT.timerTick() && end;

// P.S. move operands if you want to draw one by one

if (end) { // if it is the last tick in both graphs

// stop the timer

KillTimer(timer\_id);

// set animation stopped

graph\_DFT.setAnimState(false);

graph\_signal.setAnimState(false);

}

}

CDialogEx::OnTimer(nIDEvent); // base timer processing

}

// on system menu called

void Calculator::OnSysCommand(UINT nID, LPARAM lParam) {

// if action is 'save'

if (nID == IDD\_Calculator) {

OnBnClickedButtonSaveGr();

} else {// base case

CDialogEx::OnSysCommand(nID, lParam);

}

}

// updates CStatic with number of samples

void Calculator::OnHScroll(UINT nSBCode, UINT nPos, CScrollBar\* pScrollBar) {

// check if it's slider for number of samples

if (pScrollBar == reinterpret\_cast<CScrollBar\*>(&slider\_samples)) {

// create string

CString str;

// format the string

str.Format(L"Число отсчётов: %d", slider\_samples.GetPos());

// update CStatic text

text\_slider\_samples.SetWindowTextW(str);

}

// base action

CDialogEx::OnHScroll(nSBCode, nPos, pScrollBar);

}

// Microsoft Visual C++ generated resource script.

// coursework.rc

// there is dlg windows markup in this file

#include "resource.h"

#define APSTUDIO\_READONLY\_SYMBOLS

/////////////////////////////////////////////////////////////////////////////

//

// Generated from the TEXTINCLUDE 2 resource.

//

#ifndef APSTUDIO\_INVOKED

#include "targetver.h"

#endif

#include "afxres.h"

#include "verrsrc.h"

/////////////////////////////////////////////////////////////////////////////

#undef APSTUDIO\_READONLY\_SYMBOLS

/////////////////////////////////////////////////////////////////////////////

// Русский (Россия) resources

#if !defined(AFX\_RESOURCE\_DLL) || defined(AFX\_TARG\_RUS)

LANGUAGE LANG\_RUSSIAN, SUBLANG\_DEFAULT

#ifdef APSTUDIO\_INVOKED

/////////////////////////////////////////////////////////////////////////////

//

// TEXTINCLUDE

//

1 TEXTINCLUDE

BEGIN

"resource.h\0"

END

2 TEXTINCLUDE

BEGIN

"#ifndef APSTUDIO\_INVOKED\r\n"

"#include ""targetver.h""\r\n"

"#endif\r\n"

"#include ""afxres.h""\r\n"

"#include ""verrsrc.h""\r\n"

"\0"

END

3 TEXTINCLUDE

BEGIN

"#define \_AFX\_NO\_SPLITTER\_RESOURCES\r\n"

"#define \_AFX\_NO\_OLE\_RESOURCES\r\n"

"#define \_AFX\_NO\_TRACKER\_RESOURCES\r\n"

"#define \_AFX\_NO\_PROPERTY\_RESOURCES\r\n"

"\r\n"

"#if !defined(AFX\_RESOURCE\_DLL) || defined(AFX\_TARG\_RUS)\r\n"

"LANGUAGE 25, 1\r\n"

"#include ""res\\coursework.rc2"" // ресурсы, не редактируемые в Microsoft Visual C++\r\n"

"#include ""l.RUS\\afxres.rc"" // Стандартные компоненты\r\n"

"#if !defined(\_AFXDLL)\r\n"

"#include ""l.RUS\\afxribbon.rc"" // Лента MFC и ресурсы панели управления\r\n"

"#endif\r\n"

"#endif\r\n"

"\0"

END

#endif // APSTUDIO\_INVOKED

/////////////////////////////////////////////////////////////////////////////

//

// Icon

//

// Icon with lowest ID value placed first to ensure application icon

// remains consistent on all systems.

IDR\_MAINFRAME ICON "res\\coursework.ico"

ID\_LOGO\_ICO ICON "logo.ico"

/////////////////////////////////////////////////////////////////////////////

//

// Dialog

//

IDD\_ABOUTBOX DIALOGEX 0, 0, 241, 160

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

FONT 14, "Times New Roman", 400, 1, 0xCC

BEGIN

CTEXT "МИНОБРНАУКИ РОССИИ\nСАНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ\nЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ""ЛЭТИ""\nИМ. В.И.УЛЬЯНОВА (ЛЕНИНА)",IDC\_STATIC,7,7,227,40,SS\_NOPREFIX

CTEXT "Санкт-Петербург\n2022 г.",IDC\_STATIC,7,117,227,21

DEFPUSHBUTTON "Закрыть",IDOK,184,139,50,14,WS\_GROUP

CTEXT "Курсовая работа по информатике\nстудента ФРТ гр.1183\nЧаминова Дмитрия Анатолевича\n\n\n© СПБГЭТУ ""ЛЭТИ"" Чаминов Дмитрий Анатольевич",IDC\_STATIC,37,46,167,47,SS\_NOPREFIX

END

IDD\_COURSEWORK\_DIALOG DIALOGEX 0, 0, 227, 134

STYLE DS\_SETFONT | DS\_MODALFRAME | DS\_CENTER | WS\_POPUP | WS\_VISIBLE | WS\_CAPTION | WS\_SYSMENU

EXSTYLE WS\_EX\_APPWINDOW

FONT 14, "Times New Roman", 400, 0, 0xCC

BEGIN

DEFPUSHBUTTON "Закрыть",IDOK,5,108,40,20

CTEXT "Курсовая",IDC\_STATIC,5,7,215,20

LTEXT "краткое описание задания",IDC\_STATIC\_TASK,5,27,215,70

PUSHBUTTON "Подробнее",IDC\_Task\_About\_Bn,93,108,40,20

PUSHBUTTON "Вычислить",IDC\_CALCULATE\_BUTTON,180,108,40,20

END

IDD\_Calculator DIALOGEX 0, 0, 353, 246

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

CAPTION "Расчёт"

FONT 14, "Times New Roman", 400, 0, 0xCC

BEGIN

PUSHBUTTON "Закрыть",IDCANCEL,295,220,50,12

LTEXT "x(t)=a\*sin(2π(f+m\*t)\*t)",IDC\_STATIC\_signal,130,5,216,8

RTEXT "a",IDC\_STATIC,9,8,31,8

EDITTEXT IDC\_EDIT\_param\_a,45,6,26,12,ES\_AUTOHSCROLL

RTEXT "m\*10^(-7)",IDC\_STATIC,9,28,31,8

EDITTEXT IDC\_EDIT\_param\_m,45,26,26,12,ES\_AUTOHSCROLL

RTEXT "f\*10^(-7)",IDC\_STATIC,9,48,31,8

EDITTEXT IDC\_EDIT\_param\_f,45,46,26,12,ES\_AUTOHSCROLL

CTEXT "Шаг графика",IDC\_STATIC,10,68,95,8

CONTROL "",IDC\_SLIDER\_STEP,"msctls\_trackbar32",TBS\_BOTH | TBS\_NOTICKS | WS\_TABSTOP,10,78,95,15

LTEXT "ниже",IDC\_STATIC,10,93,19,8

LTEXT "выше",IDC\_STATIC,90,93,17,8

LTEXT "x\*10^4",IDC\_STATIC,8,110,44,8

EDITTEXT IDC\_EDIT\_xscale\_from,38,110,26,12,ES\_AUTOHSCROLL

CTEXT "...",IDC\_STATIC,68,115,8,8

EDITTEXT IDC\_EDIT\_xscale\_to,81,110,26,12,ES\_AUTOHSCROLL

LTEXT "Сигнальная функция",IDC\_STATIC,10,124,107,8

LTEXT "y",IDC\_STATIC,20,134,8,8

EDITTEXT IDC\_EDIT\_yscale\_from,30,134,26,12,ES\_AUTOHSCROLL

CTEXT "...",IDC\_STATIC,60,134,8,8

EDITTEXT IDC\_EDIT\_yscale\_to,73,134,26,12,ES\_AUTOHSCROLL

CONTROL "Логарифмический масштаб",IDC\_CHECK\_is\_log\_scale,"Button",BS\_AUTOCHECKBOX | WS\_TABSTOP,20,149,99,10

LTEXT "Дискретное преобразование Фурье",IDC\_STATIC,10,164,107,8

LTEXT "y",IDC\_STATIC,20,205,8,8

EDITTEXT IDC\_EDIT\_yscale\_from2,30,205,26,12,ES\_AUTOHSCROLL

CTEXT "...",IDC\_STATIC,60,209,8,8

EDITTEXT IDC\_EDIT\_yscale\_to2,73,205,26,12,ES\_AUTOHSCROLL

CONTROL "Логарифмический масштаб",IDC\_CHECK\_is\_log\_scale2,"Button",BS\_AUTOCHECKBOX | WS\_TABSTOP,20,220,99,10

CONTROL "Анимация",IDC\_CHECK\_ANIM,"Button",BS\_AUTOCHECKBOX | WS\_TABSTOP,77,5,47,10

LTEXT "Фон",IDC\_STATIC,77,16,18,12

CONTROL "",IDC\_MFCCOLORBUTTON\_BG,"MfcColorButton",WS\_TABSTOP,102,16,18,12

LTEXT "Сигнал",IDC\_STATIC,77,30,22,12

CONTROL "",IDC\_MFCCOLORBUTTON\_SIGNAL,"MfcColorButton",WS\_TABSTOP,102,30,18,12

LTEXT "ДПФ",IDC\_STATIC,77,45,18,12

CONTROL "",IDC\_MFCCOLORBUTTON\_DCF,"MfcColorButton",WS\_TABSTOP,102,45,18,12

PUSHBUTTON "Обновить",IDC\_BUTTON\_UPDATE\_graph,130,220,50,12

PUSHBUTTON "Сбросить",IDC\_BUTTON\_reset,185,220,50,12

PUSHBUTTON "Сохранить",IDC\_BUTTON\_SAVE\_GR,240,220,50,12

LTEXT "МЕСТО ДЛЯ ГРАФИКА",IDC\_STATIC\_graph,131,15,215,86,SS\_SUNKEN,WS\_EX\_TRANSPARENT

LTEXT "МЕСТО ДЛЯ ГРАФИКА",IDC\_STATIC\_graph2,130,106,215,99,SS\_SUNKEN,WS\_EX\_TRANSPARENT

LTEXT "координаты курсора",IDC\_STATIC\_CPOS,130,205,190,8

LTEXT "Число отсчётов",IDC\_STATIC\_SAMPLES,20,177,85,8

CONTROL "",IDC\_SLIDER\_SAMPLES,"msctls\_trackbar32",TBS\_BOTH | TBS\_NOTICKS | WS\_TABSTOP,20,190,85,11

END

/////////////////////////////////////////////////////////////////////////////

//

// Version

//

VS\_VERSION\_INFO VERSIONINFO

FILEVERSION 0,5,0,3

PRODUCTVERSION 1,0,0,1

FILEFLAGSMASK 0x3fL

#ifdef \_DEBUG

FILEFLAGS 0x1L

#else

FILEFLAGS 0x0L

#endif

FILEOS 0x40004L

FILETYPE 0x1L

FILESUBTYPE 0x0L

BEGIN

BLOCK "StringFileInfo"

BEGIN

BLOCK "041904b0"

BEGIN

VALUE "CompanyName", "СПБГЭТУ «ЛЭТИ»"

VALUE "FileDescription", "Курсовая Чаминова Д. А. СБГЭТУ ""ЛЭТИ"" ФРТ гр. 1183"

VALUE "FileVersion", "0.5.0.3"

VALUE "InternalName", "coursework-Chaminov.exe"

VALUE "LegalCopyright", "(c) СПБГЭТУ «ЛЭТИ».Все права защищены."

VALUE "OriginalFilename", "coursework-Chaminov.exe"

VALUE "ProductName", "Курсовая работа студента ФРТ гр. 1183 по дисциплине «Информационные технологии» Чаминова Дмитрия Анатольевича"

VALUE "ProductVersion", "1.0.0.1"

END

END

BLOCK "VarFileInfo"

BEGIN

VALUE "Translation", 0x419, 1200

END

END

/////////////////////////////////////////////////////////////////////////////

//

// DESIGNINFO

//

#ifdef APSTUDIO\_INVOKED

GUIDELINES DESIGNINFO

BEGIN

IDD\_ABOUTBOX, DIALOG

BEGIN

LEFTMARGIN, 7

RIGHTMARGIN, 234

VERTGUIDE, 37

VERTGUIDE, 204

TOPMARGIN, 7

BOTTOMMARGIN, 153

END

IDD\_COURSEWORK\_DIALOG, DIALOG

BEGIN

LEFTMARGIN, 5

RIGHTMARGIN, 220

TOPMARGIN, 7

BOTTOMMARGIN, 128

HORZGUIDE, 108

END

IDD\_Calculator, DIALOG

BEGIN

RIGHTMARGIN, 346

VERTGUIDE, 8

VERTGUIDE, 120

TOPMARGIN, 5

BOTTOMMARGIN, 239

HORZGUIDE, 2

HORZGUIDE, 122

END

END

#endif // APSTUDIO\_INVOKED

/////////////////////////////////////////////////////////////////////////////

//

// AFX\_DIALOG\_LAYOUT

//

IDD\_COURSEWORK\_DIALOG AFX\_DIALOG\_LAYOUT

BEGIN

0

END

IDD\_ABOUTBOX AFX\_DIALOG\_LAYOUT

BEGIN

0

END

IDD\_Calculator AFX\_DIALOG\_LAYOUT

BEGIN

0

END

/////////////////////////////////////////////////////////////////////////////

//

// Dialog Info

//

IDD\_Calculator DLGINIT

BEGIN

IDC\_MFCCOLORBUTTON\_BG, 0x37c, 215, 0

0x4d3c, 0x4346, 0x6f43, 0x6f6c, 0x4272, 0x7475, 0x6f74, 0x5f6e, 0x6e45,

0x6261, 0x656c, 0x744f, 0x6568, 0x4272, 0x7475, 0x6f74, 0x3e6e, 0x4146,

0x534c, 0x3c45, 0x4d2f, 0x4346, 0x6f43, 0x6f6c, 0x4272, 0x7475, 0x6f74,

0x5f6e, 0x6e45, 0x6261, 0x656c, 0x744f, 0x6568, 0x4272, 0x7475, 0x6f74,

0x3e6e, 0x4d3c, 0x4346, 0x6f43, 0x6f6c, 0x4272, 0x7475, 0x6f74, 0x5f6e,

0x6e45, 0x6261, 0x656c, 0x7541, 0x6f74, 0x616d, 0x6974, 0x4263, 0x7475,

0x6f74, 0x3e6e, 0x5254, 0x4555, 0x2f3c, 0x464d, 0x4343, 0x6c6f, 0x726f,

0x7542, 0x7474, 0x6e6f, 0x455f, 0x616e, 0x6c62, 0x4165, 0x7475, 0x6d6f,

0x7461, 0x6369, 0x7542, 0x7474, 0x6e6f, 0x3c3e, 0x464d, 0x4343, 0x6c6f,

0x726f, 0x7542, 0x7474, 0x6e6f, 0x435f, 0x6c6f, 0x6d75, 0x736e, 0x6f43,

0x6e75, 0x3e74, 0x3c35, 0x4d2f, 0x4346, 0x6f43, 0x6f6c, 0x4272, 0x7475,

0x6f74, 0x5f6e, 0x6f43, 0x756c, 0x6e6d, 0x4373, 0x756f, 0x746e, "\076"

IDC\_MFCCOLORBUTTON\_SIGNAL, 0x37c, 215, 0

0x4d3c, 0x4346, 0x6f43, 0x6f6c, 0x4272, 0x7475, 0x6f74, 0x5f6e, 0x6e45,

0x6261, 0x656c, 0x744f, 0x6568, 0x4272, 0x7475, 0x6f74, 0x3e6e, 0x4146,

0x534c, 0x3c45, 0x4d2f, 0x4346, 0x6f43, 0x6f6c, 0x4272, 0x7475, 0x6f74,

0x5f6e, 0x6e45, 0x6261, 0x656c, 0x744f, 0x6568, 0x4272, 0x7475, 0x6f74,

0x3e6e, 0x4d3c, 0x4346, 0x6f43, 0x6f6c, 0x4272, 0x7475, 0x6f74, 0x5f6e,

0x6e45, 0x6261, 0x656c, 0x7541, 0x6f74, 0x616d, 0x6974, 0x4263, 0x7475,

0x6f74, 0x3e6e, 0x5254, 0x4555, 0x2f3c, 0x464d, 0x4343, 0x6c6f, 0x726f,

0x7542, 0x7474, 0x6e6f, 0x455f, 0x616e, 0x6c62, 0x4165, 0x7475, 0x6d6f,

0x7461, 0x6369, 0x7542, 0x7474, 0x6e6f, 0x3c3e, 0x464d, 0x4343, 0x6c6f,

0x726f, 0x7542, 0x7474, 0x6e6f, 0x435f, 0x6c6f, 0x6d75, 0x736e, 0x6f43,

0x6e75, 0x3e74, 0x3c35, 0x4d2f, 0x4346, 0x6f43, 0x6f6c, 0x4272, 0x7475,

0x6f74, 0x5f6e, 0x6f43, 0x756c, 0x6e6d, 0x4373, 0x756f, 0x746e, "\076"

IDC\_MFCCOLORBUTTON\_DCF, 0x37c, 215, 0

0x4d3c, 0x4346, 0x6f43, 0x6f6c, 0x4272, 0x7475, 0x6f74, 0x5f6e, 0x6e45,

0x6261, 0x656c, 0x744f, 0x6568, 0x4272, 0x7475, 0x6f74, 0x3e6e, 0x4146,

0x534c, 0x3c45, 0x4d2f, 0x4346, 0x6f43, 0x6f6c, 0x4272, 0x7475, 0x6f74,

0x5f6e, 0x6e45, 0x6261, 0x656c, 0x744f, 0x6568, 0x4272, 0x7475, 0x6f74,

0x3e6e, 0x4d3c, 0x4346, 0x6f43, 0x6f6c, 0x4272, 0x7475, 0x6f74, 0x5f6e,

0x6e45, 0x6261, 0x656c, 0x7541, 0x6f74, 0x616d, 0x6974, 0x4263, 0x7475,

0x6f74, 0x3e6e, 0x5254, 0x4555, 0x2f3c, 0x464d, 0x4343, 0x6c6f, 0x726f,

0x7542, 0x7474, 0x6e6f, 0x455f, 0x616e, 0x6c62, 0x4165, 0x7475, 0x6d6f,

0x7461, 0x6369, 0x7542, 0x7474, 0x6e6f, 0x3c3e, 0x464d, 0x4343, 0x6c6f,

0x726f, 0x7542, 0x7474, 0x6e6f, 0x435f, 0x6c6f, 0x6d75, 0x736e, 0x6f43,

0x6e75, 0x3e74, 0x3c35, 0x4d2f, 0x4346, 0x6f43, 0x6f6c, 0x4272, 0x7475,

0x6f74, 0x5f6e, 0x6f43, 0x756c, 0x6e6d, 0x4373, 0x756f, 0x746e, "\076"

0

END

/////////////////////////////////////////////////////////////////////////////

//

// String Table

//

STRINGTABLE

BEGIN

IDS\_ABOUTBOX "&О программе"

IDS\_VERSION "0.5.03"

IDS\_COMMIT "7fff7af5b2cd7ac1d028f38406607bd008ae2150"

IDS\_SHORT\_TASK\_INFO "Для сигнала заданного вида вычислить дискретное преобразование Фурье(ДПФ), нарисовать график сигнальной функции и график модуля ее ДПФ, предоставить возможность изменять масштаб графика по Х и по У, выбор линейного или логарифмического масштаба по У для графика ДПФ, интерактивный режим изменения параметров сигнала, запись в файл графиков сигнала и ДПФ в формате BMP."

IDS\_MORE\_TASK\_INFO "Дискретное преобразование Фурье является линейным преобразованием, которое переводит 𝖭-мерный комплексный вектор временных отсчётов 𝘅 в 𝖭-мерный комплексный вектор спектральных (частотных) отсчётов 𝗫 той же длины. Таким образом преобразование может быть реализовано как умножение квадратной матрицы на вектор 𝗫=𝖠∙𝘅 , где матрица 𝖠 – некая унитарная матрица, которую можно рассматривать как матрицу преобразования (поворота) системы координат некоторого 𝖭-мерного пространства.\nФормула преобразования для 𝗄-того компонента ДПФ:\n𝖷𝗄=Σ𝗑𝗇2219𝖾𝗑𝗉(-2π𝗂𝗄𝗇/𝖭)\nЭту формулу можно преобразовать с помощью формулы Эйлера.\nКвадрат модуля отсчета равен |𝖷𝗄|²=𝖱𝖾²(𝖷𝗄)+𝖨𝗆²(𝖷𝗄)"

END

#endif // Русский (Россия) resources

/////////////////////////////////////////////////////////////////////////////

#ifndef APSTUDIO\_INVOKED

/////////////////////////////////////////////////////////////////////////////

//

// Generated from the TEXTINCLUDE 3 resource.

//

#define \_AFX\_NO\_SPLITTER\_RESOURCES

#define \_AFX\_NO\_OLE\_RESOURCES

#define \_AFX\_NO\_TRACKER\_RESOURCES

#define \_AFX\_NO\_PROPERTY\_RESOURCES

#if !defined(AFX\_RESOURCE\_DLL) || defined(AFX\_TARG\_RUS)

LANGUAGE 25, 1

#include "res\coursework.rc2" // ресурсы, не редактируемые в Microsoft Visual C++

#include "l.RUS\afxres.rc" // Стандартные компоненты

#if !defined(\_AFXDLL)

#include "l.RUS\afxribbon.rc" // Лента MFC и ресурсы панели управления

#endif

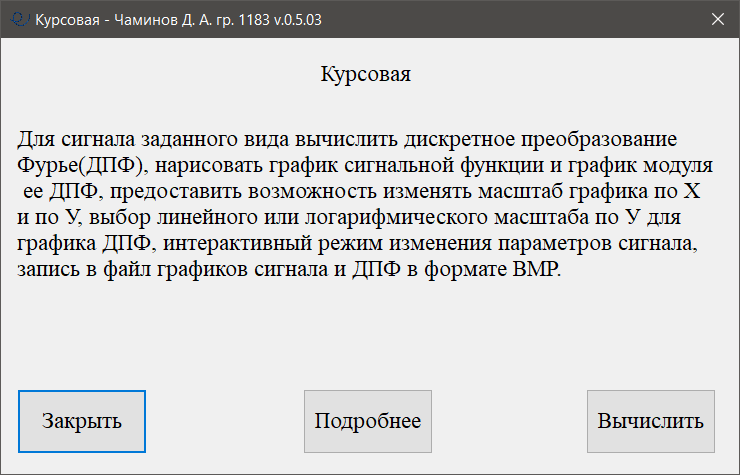
#endif

/////////////////////////////////////////////////////////////////////////////

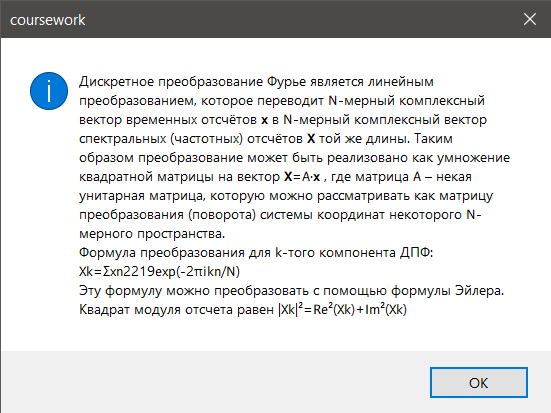
#endif // not APSTUDIO\_INVOKED

# Рисунки с копиями экрана при работе программы

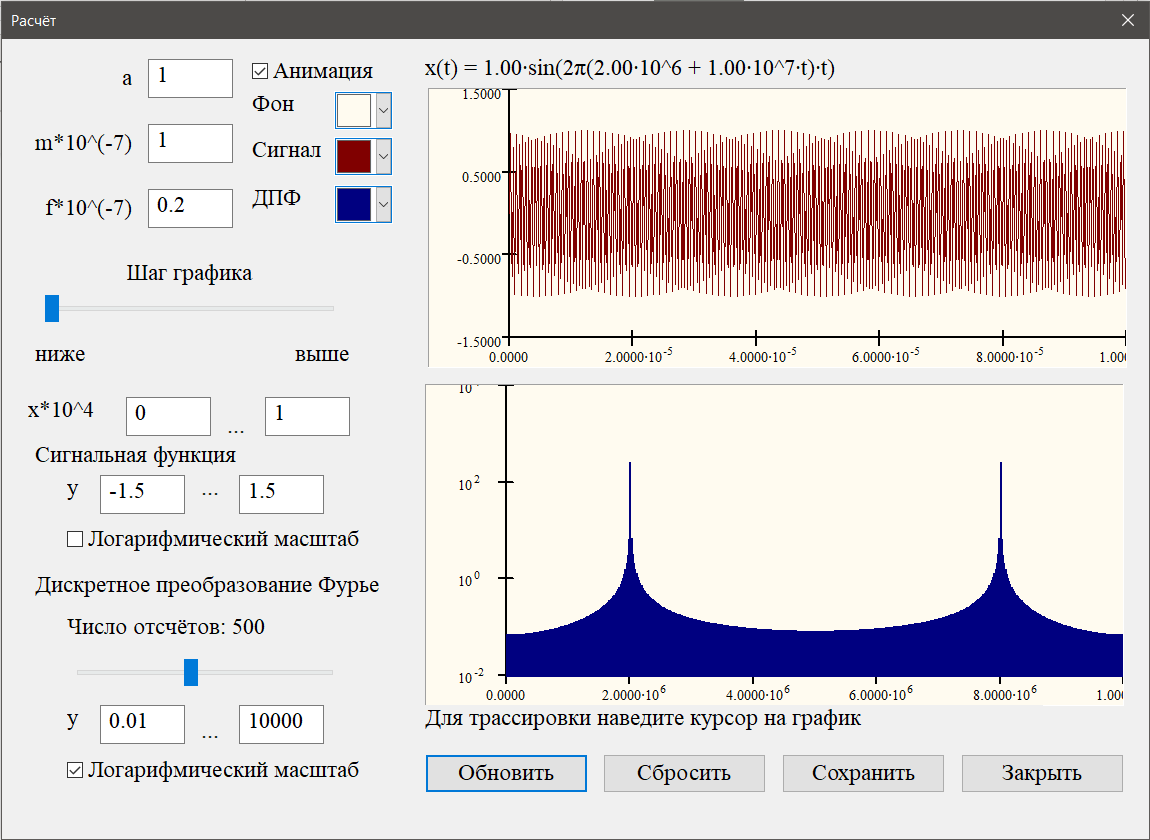
Главное окно:



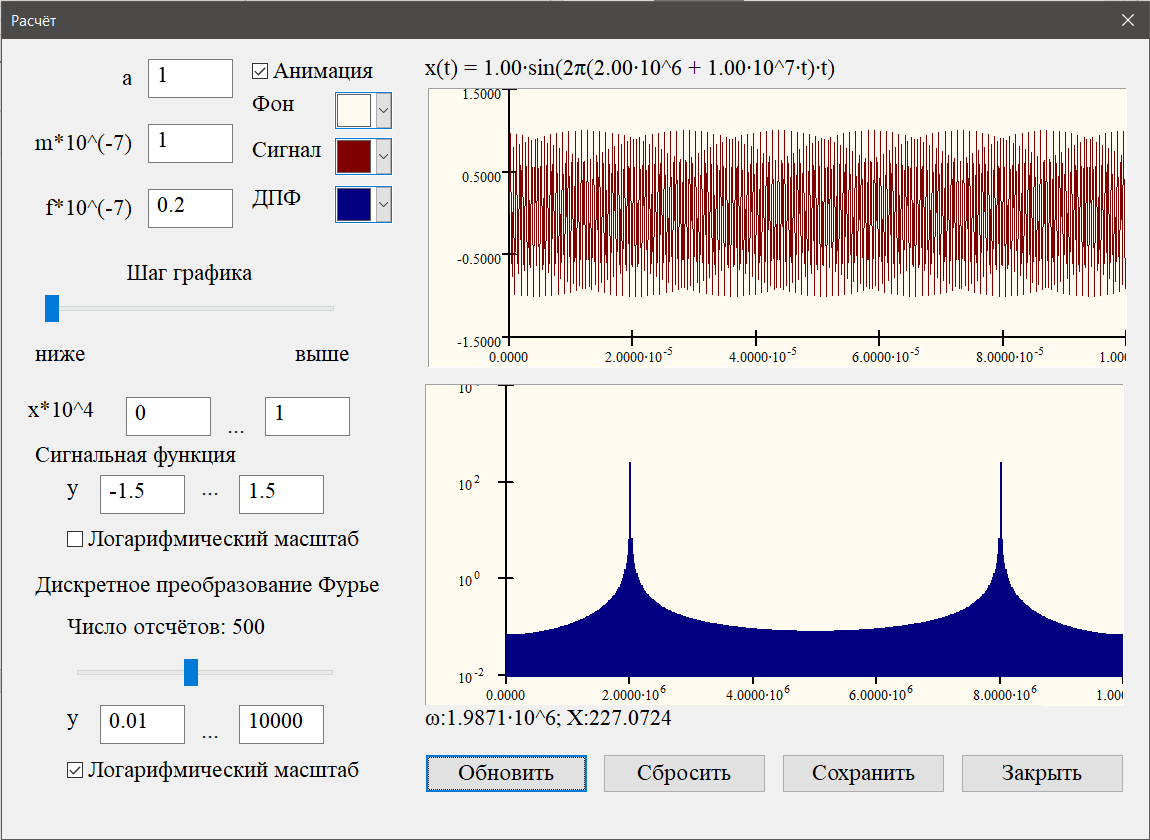
Дополнительная информация о ДПФ:

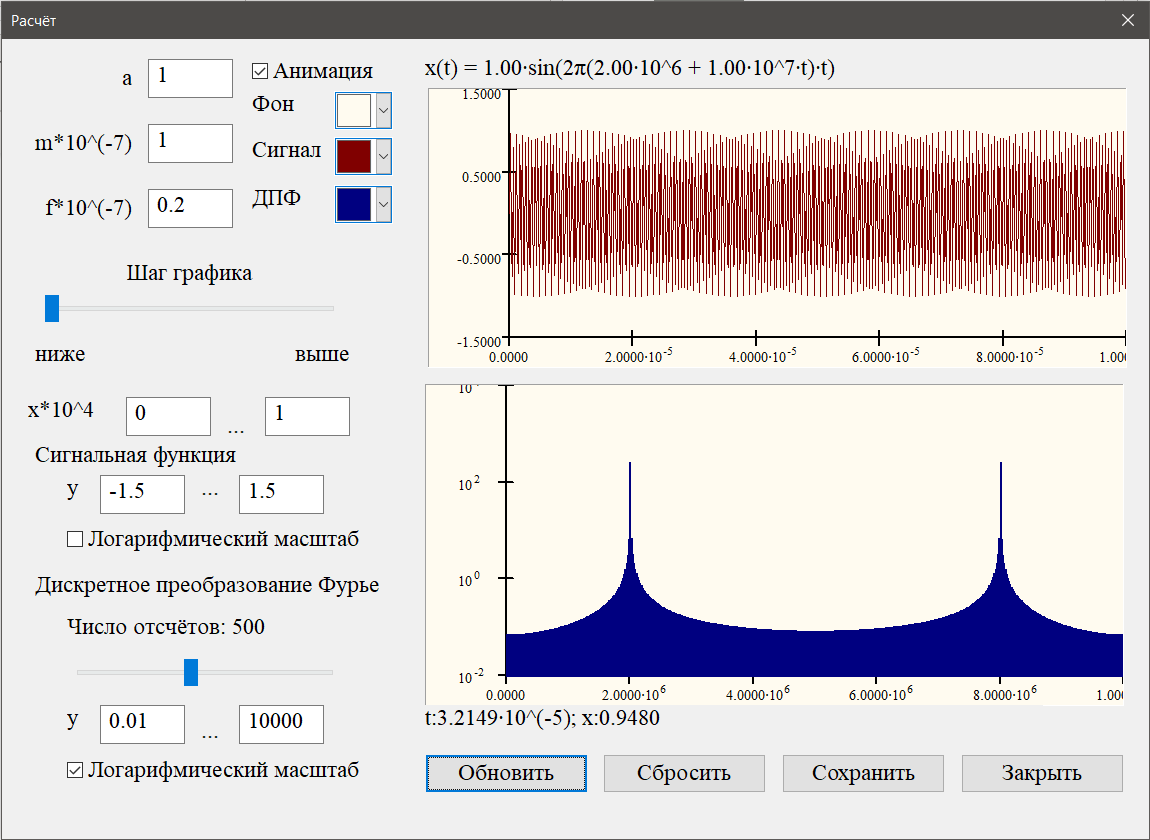


Выполненный расчёт:

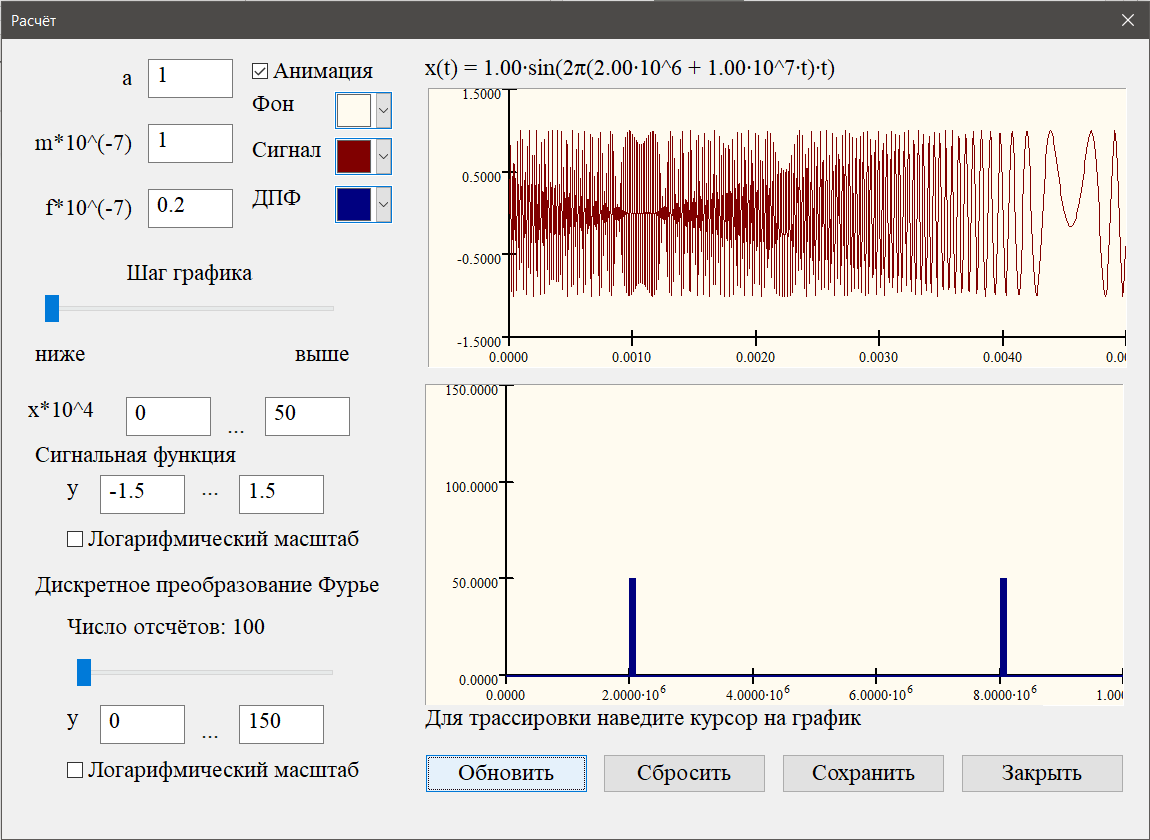


Трассировка значений на графике и гистограмме:

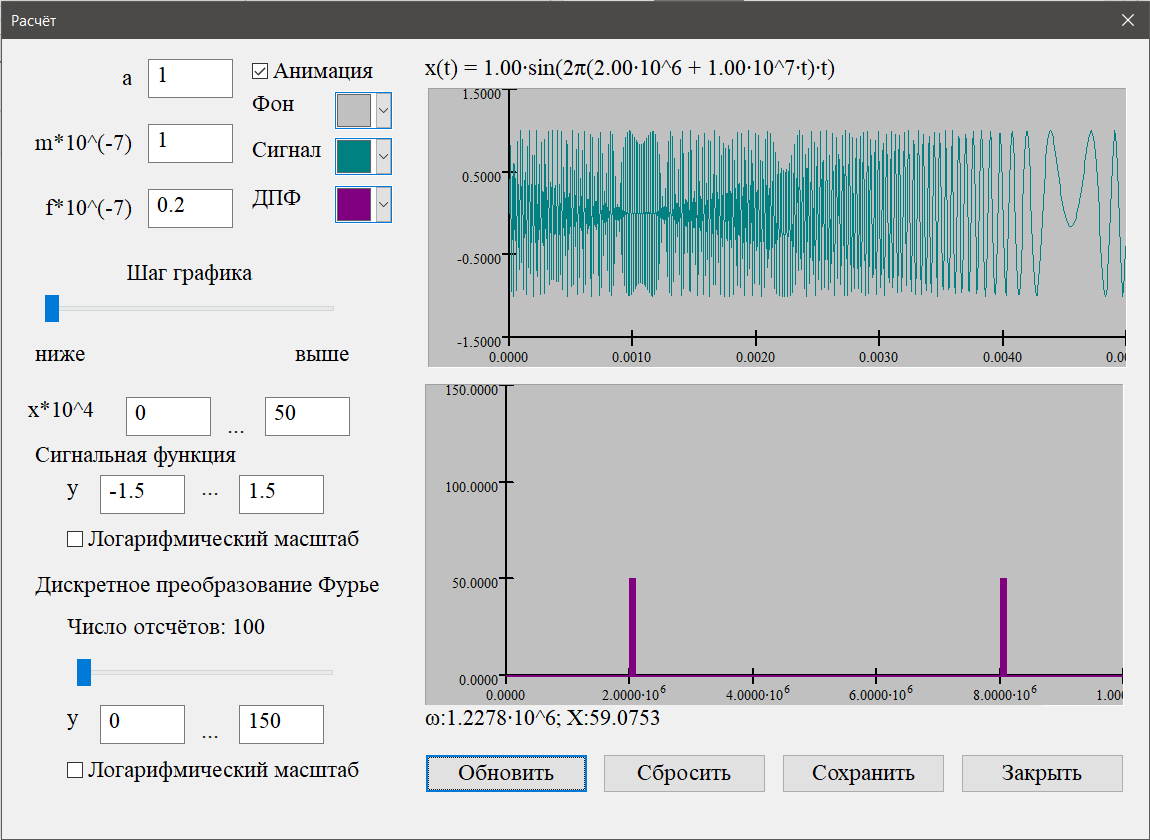




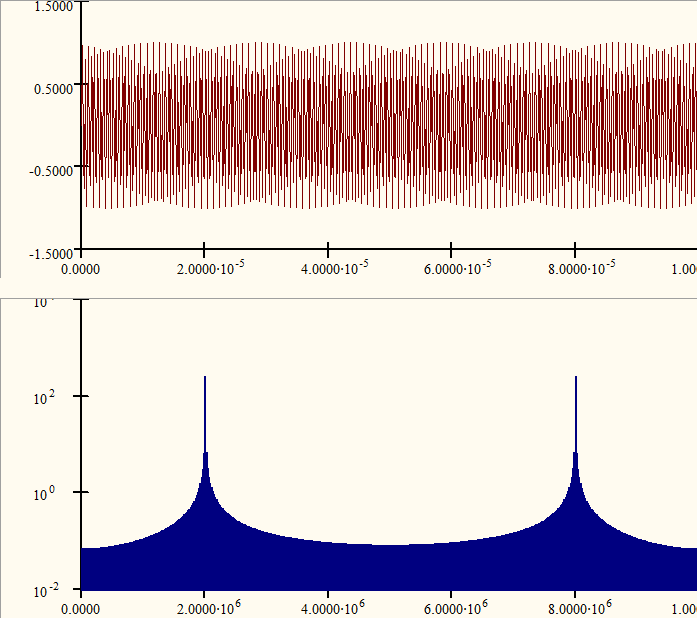
Изменённый масштаб:



Изменение цветов:



Сохраненное изображение:



# Контрольный пример, сравнение результата с эталоном

Для проверки ДПФ были использованы тесты GoogleTest (gtest), а в качестве эталона взята функция, реализующая быстрое преобразование Фурье из библиотеки numpy для языка программирования python. Объект теста (NumpyComparisonTest – наследник ::testing::Test) при инициализации запускал генератор тестов (файл “the standard.py”, написанный на языке python 3.6) и открывал файл “test.txt”, в который производились записи при исполнении “the standard.py”. Затем для каждой из 100 пар строк (исходные данные и само преобразование, рассчитанное с помощью библиотечной функции numpy.fft.fft) производилось ДПФ и сравнение с эталоном. Исходный код тестов приведен ниже.

//

// pch.h

//

#define \_USE\_MATH\_DEFINES // for pi constant

#pragma once

// including gtest lib

#include "gtest/gtest.h"

//

// pch.cpp

//

#include "pch.h"

//

// test.cpp

//

#include "pch.h" //for gtest

#include <process.h> // for 'system'

#include <fstream> // for file reading

// for pi constant and

// trigonometry

#include <cmath>

/// <summary>

/// Class of the tests

/// </summary>

class NumpyComparisonTest : public ::testing::Test

{

/// <summary>

/// On initialization

/// </summary>

void SetUp() override {

genTest();

fin.open("test.txt");

}

/// <summary>

/// On destruction

/// </summary>

void TearDown() override {

fin.close();

}

public:

/// <summary>

/// vector of calculated values

/// </summary>

std::vector<double> calculated\_dft;

/// <summary>

/// vector of read dft values

/// </summary>

std::vector<double> true\_dft;

/// <summary>

/// Samples size

/// </summary>

size\_t N = 1000;

protected:

/// <summary>

/// input file stream

/// </summary>

std::ifstream fin;

/// <summary>

/// vector of read samples

/// </summary>

std::vector<double> samples;

/// <summary>

/// asks python file to generate the tests

/// </summary>

void genTest() {

system("python \"./the standard.py\"");

}

/// <summary>

/// DFT function that copied from the project

/// </summary>

/// <param name="samples">a vector of samples</param>

/// <param name="m">number of dft sample to calculate</param>

/// <returns>absolute value of the DFT m sample</returns>

double dft(int m) {

// real and imaginary parts

double re = 0, im = 0;

// for each sample

for (size\_t n = 0; n < N; n++) {

// summing

re += samples[n] \* cos(2 \* M\_PI \* m \* n / N);

im += samples[n] \* sin(-2 \* M\_PI \* m \* n / N);

}

// return the abs(X)

return sqrt(re \* re + im \* im);

}

/// <summary>

/// Reads 'n' numbers from the stream

/// </summary>

/// <param name="in">stream to read</param>

/// <param name="out">output vector</param>

/// <param name="n">amount of numbers</param>

void readTheTestLine(std::vector<double>& out) {

// reserving memory

out.resize(N);

// for each i = 0, 1, ..., n-1

for (int i = 0; i < N; i++) {

// read a number from the stream to the vector[i]

fin >> out[i];

}

}

/// <summary>

/// Calculates dft absolute values

/// </summary>

/// <param name="signal">a vector of source data</param>

/// <param name="out">a place for result</param>

void genDft() {

// reserve memory

calculated\_dft.resize(N);

// for each i = 0, 1, ..., N-1

for (int i = 0; i < N; i++) {

// calculate DFT sample and

// write it to the vector

calculated\_dft[i] = dft(i);

}

}

/// <summary>

/// Calculates one test

/// </summary>

void calculate\_one() {

readTheTestLine(samples);

readTheTestLine(true\_dft);

genDft();

}

};

// Test it

TEST\_F(NumpyComparisonTest, RandomTests) {

// for each of 100 tests

for (int i = 0; i < 100; i++) {

// calculate one

calculate\_one();

// comparison loop

for (int i = 0; i < N; i++) {

ASSERT\_NEAR(true\_dft[i], calculated\_dft[i], 1e-10);

}

}

}

#

# the standart.py

#

# including lib for dft

import numpy as np

# including lib for generating pseudo random numbers

from random import random as rnd

# define the function of 1 argument

def generate\_sequence(n):

"""

generates sequence of numbers in [0.0, 1.0)

with length 'n' and returns 'np.array' of them

"""

# result array

result = np.array([],dtype=np.double)

# reserving memory

result.resize(n)

# repeat for each i = 0, 1, 2, ..., n-1

for i in range(n):

# assign array item to random number

result[i] = rnd()

# return result array

return result

# define the function of 3 arguments

def generate\_test(test\_amount, n, filename):

"""

creates file 'filename' and writes to

it 'testamount'\*2 strings of 'n' elements

alternating source samples and resulting DFT

"""

# open the file

with open(filename, 'w') as f:

# for each i = 0, 1, 2, ..., 'test\_amount'-1

for i in range(test\_amount):

# get sequence of random numbers

s = generate\_sequence(n)

# convert numbers to the strings

# and write it to string with space separtion

string = ' '.join(map(str,s))

# write a line

f.write(string)

f.write('\n')

# calculate DFT using library Fast

# Fourier transform function

dft = np.absolute(np.fft.fft(s))

# convert numbers to the strings

# and write it to string with space separtion

string = ' '.join(map(str,dft))

# write a line

f.write(string)

f.write('\n')

# call the function

"""

create file 'text.txt' in current

path and write 100 numbers sequences

DFT's with length 1000 to it

"""

generate\_test(100,1000,'./test.txt')

Изображение выглядит как текст

Автоматически созданное описаниеТаким образом проверка преобразования была проведена на 100 случайных выборках, каждую из которых составляло 1000 чисел в диапазоне от 0 до 1. Снимок экрана после запуска тестов, подтверждающий их успешное прохождение приведен ниже.

# Ведомость соответствия программы спецификации

|  |  |  |
| --- | --- | --- |
| № | Краткое содержание пункта спецификации | Соответствие спецификации (+/-) |
| 1.1.1 | Выполнение ДПФ программой в соответствии с данными, введёнными пользователем, визуализация результата расчётов и сигнальной функции. | + |
| 1.1.2 | Построение графика, менять параметры масштаба; возможность изменить сигнальную функцию; сохранение графика в формате BMP. | + |
| 1.1.3 | Поддержка кодировки Unicode. | + |
| 1.1.4 | Наличие графического интерфейса. | + |
| 1.1.5 | Наличие пользовательского типа данных, оперирование динамическими массивами. | + |
| 1.1.6 | Разработка в среде MS Visual Studio с использованием библиотек MFC, работоспособность приложения в операционных системах MS Windows 7 (x32) и выше. | + |
| 1.1.7 | Наличие сведений о приложении, авторе, авторских правах, иконки. | + |
| 1.1.8 | Наличие кнопок и полей для ввода, для управления приложением и задания параметров функции. | + |
| 1.1.9 | Наличие списков выбора. | + |
| 1.1.10 | Наличие русскоязычного, интуитивно понятного, «дружественного» интерфейса. | + |
| 1.2.1 | Соответствие отчета ГОСТу 19.701–90 единой системы программной документации. | + |
| 1.2.2 | Наличие необходимых разделов отчета. | + |

# TODO: Выводы