Министерство образования Республики Беларусь

Учреждение образования БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Программирование на языках высокого уровня

К ЗАЩИТЕ ДОПУСТИТЬ

Ассистент кафедры ЭВМ

А. В. Марзалюк

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе на тему

ИНФОРМАЦИОННАЯ СИСТЕМА КОМПЬЮТЕРНОГО МАГАЗИНА С ВОЗМОЖНОСТЬЮ ПОДБОРА КОНФИГУРАЦИИ

БГУИР КР 1-40 02 01 018 ПЗ

Студент А. Г. Слинько

Руководитель А. В. Марзалюк

МИНСК 2023

Учреждение образования

«Белорусский государственный университет информатики   
и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой ЭВМ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

––"–––––––––––––––––20 г.

ЗАДАНИЕ

по курсовому проектированию

Студенту Антоненко Александру Владимировичу

1. Тема проекта Построитель двухмерных графиков функций

2. Срок сдачи студентом законченного проекта 11 *декабря* 2023 *г.*

3. Исходные данные к проекту: нет.

4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке)

1. Лист задания.

2. Введение.

3. Постановка задачи

4. Обзор литературы.

4.1. Обзор методов и алгоритмов решения поставленной задачи.

5. Функциональное проектирование.

5.1. Структура входных и выходных данных.

5.2. Разработка диаграммы классов.

5.3. Описание классов.

6. Разработка программных модулей.

6.1. Разработка блок-схем алгоритмов (два наиболее важных метода).

6.2. Разработка алгоритмов (описание алгоритмов по шагам, для двух

методов).

7. Результаты работы программы.

8. Заключение

9. Литература

10. Приложения

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

6. Консультант по проекту(с обозначением разделов проекта)

*Марзалюк А. В*.

7. Дата выдачи задания 15.09.*20*23 *г.*

8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и трудоемкости отдельных этапов):

1. Выбор задания. Разработка содержания пояснительной

Перечень графического материала - к 01.10.2023 15 %

разделы 2, 3, 4 - к 01.10.2023 10 %:

раздел 5 - к 01.11.2023 20 %:

раздел б - к 01.12.2023 35 %:

разделы 7.89 - к 01.12.2023 5 %:

раздел 10 - к 01.12.2023 5 %:

оформление пояснительной записки и графического материала к 11.12.2023

-10 %:

*Защита курсового проекта с 2*1.12 *по 28*.12.2023 *г.*

РУКОВОДИТЕЛЬ– ––––––––– *Марзалюк А. В.*

Задание принял к исполнению –Х Антоненко *А. В.*

#### ВВЕДЕНИЕ

Эта работа посвящена разработке программного средства для построения двухмерных графиков функций.

Целью работы является приобретение навыков разработки программных средств для визуализации математических функций на компьютере. В данной работе ставится задача создать удобное приложение, которое позволит пользователям строить графики функций и анализировать их.

Программное средство будет разработано в виде графического приложения с использованием библиотеки Qt для создания современного пользовательского интерфейса, при этом используется язык программирования C++. Пользовательский интерфейс будет включать в себя следующие основные элементы:

1. Окно для ввода математической функции.

2. Окно для настройки параметров графика, таких как масштаб и цвета.

3. Окно с самим графиком функции.

В ходе работы применяются различные парадигмы программирования, включая объектно-ориентированное программирование и работу с данными, чтобы обеспечить функциональность приложения.

Таким образом, целью этой работы является создание удобного и мощного программного средства для построения графиков математических функций.

**1 ПОСТАНОВКА ЗАДАЧИ**

Разработать программное средство для построения двухмерных графиков математических функций. Программа должна предоставлять удобный и интерактивный интерфейс для ввода, настройки и визуализации графиков функций.

Задачи работы:

1. Изучить основные принципы построения графиков функций и математические методы для их вычисления.

2. Разработать пользовательский интерфейс программы, включающий в себя:

- Окно для ввода математических функций.

- Окно для настройки параметров графиков (масштаб, цвета и т. д.).

- Окно для отображения графиков функций.

3. Реализовать алгоритмы построения графиков функций на основе введенных пользователем данных.

4. Провести тестирование программы на различных функциях и сценариях использования.

5. Ожидаемые результаты:

**2 ОБЗОР ЛИТЕРАТУРЫ**

В данном разделе проведен обзор существующих методов и алгоритмов, используемых для построения двухмерных графиков функций. Этот обзор поможет определить подходы, которые можно применить при разработке программного средства "Построитель двухмерных графиков функций" на основе библиотеки Qt и языка программирования C++.

Графики функций - мощный инструмент, который помогает наглядно представить зависимость между переменными. Изучение графиков не только делает математику более доступной, но и имеет широкое применение в реальной жизни.

На графике функции мы имеем оси координат, где ось X отображает одну переменную, а ось Y - другую. Точки на графике представляют значения функции для соответствующих входных данных, а кривые соединяют эти точки, образуя график.

Примеры различных функций:

1. Линейные функции: График представляет собой прямую линию. Прямолинейные, их графики не имеют изгибов.

2. Квадратичные функции: Форма графика напоминает параболу. Могут иметь вершину вверх или вниз, в зависимости от знака коэффициента перед квадратичным членом.

3. Тригонометрические функции: Синус и косинус имеют периодические графики. Регулярно повторяющиеся, соответствуя периоду синуса и косинуса.

Графики функций находят широкое применение:

- Физика: Отображение траектории движения объектов.

- Экономика: Моделирование зависимости между переменными в экономических системах.

- Биология: Анализ изменения параметров в биологических процессах.

2.1. Обзор методов и алгоритмов решения поставленной задачи

1. Алгоритмы построения точек графика:

- Метод дискретизации: Один из наиболее распространенных методов. Функция оценивается в дискретных точках на горизонтальной оси, и затем точки соединяются линиями. Этот метод обеспечивает приемлемую скорость построения и хороший результат для большинства функций.

- Алгоритм Брезенхема: Применяется для построения линейных отрезков. Эффективен при рисовании прямых линий и может быть использован для построения графиков.

2. Управление масштабом и интерактивность:

- Масштабирование и перемещение: Для удобства пользователей важно предоставить возможность изменения масштаба. Это может быть реализовано путем изменения диапазона значений на оси X и Y.

- Интерактивные элементы: Добавление элементов управления, таких как ползунки для масштабирования, кнопки для переключения между графиками, анимации и подписи к осям, улучшает интерактивность приложения.

3. Библиотека Qt и C++:

-Использование библиотеки Qt: Библиотека Qt предоставляет богатые средства для создания графических приложений с интерфейсами. Она включает в себя классы для рисования, обработки событий и создания пользовательских виджетов.

-Язык программирования C++: Программирование на C++ обеспечивает высокую производительность и контроль над памятью, что важно при разработке графических приложений.

4. Исследование аналогичных приложений:

-Графические приложения для построения графиков: Исследование существующих графических приложений, таких как WolframAlpha[1] (рисунок 2.1.1), Gnuplot, Matplotlib и Microsoft Excel, может дать представление о стандартах и ожиданиях пользователей.

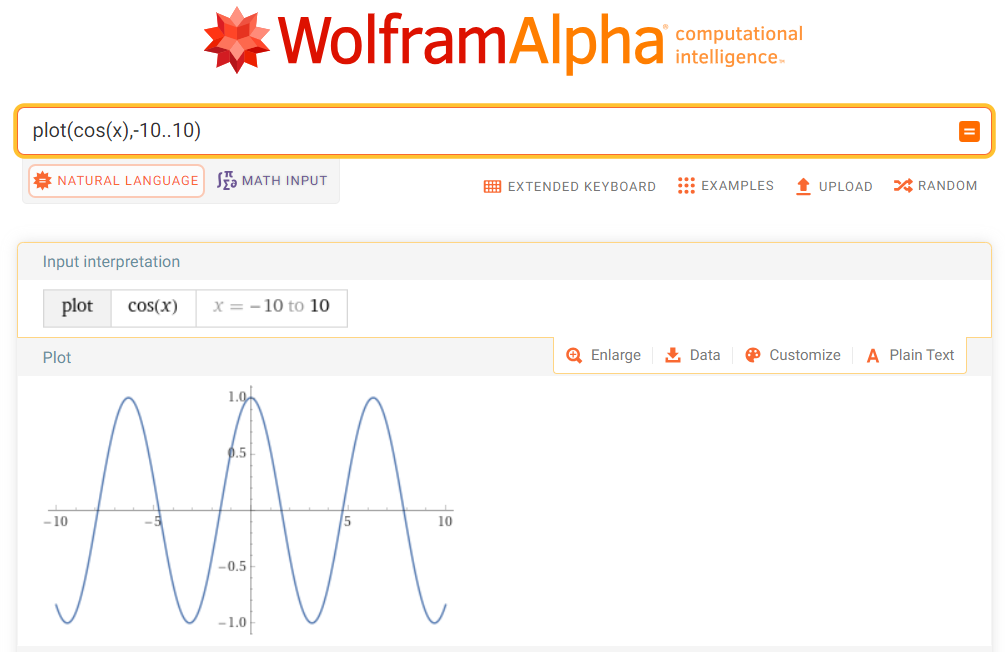


Рисунок 2.1.1 – приложение для построений графиков WolframAlpha

В заключение, обзор литературы показывает, что существуют разнообразные методы и подходы для построения двухмерных графиков функций. Интерактивность и удобство использования играют важную роль в разработке подобных приложений. Применение библиотеки Qt и языка C++ предоставляет мощные инструменты для создания высококачественных графических приложений.

**3. ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ**

В рамках данной главы рассмотренны основные аспекты функционального проектирования для построения 2D графиков функций. Этот этап играет ключевую роль в создании программного продукта, так как определяет структуру входных и выходных данных, а также включает в себя разработку диаграммы классов и описание этих классов.

3.1. Структура входных и выходных данных

На таблице 3.1.1 предстваленна структура входных и выходных данных.

Она служит для такого-то...

Таблица 1 – структура входных и выходных данных

|  |  |  |
| --- | --- | --- |
| Входные данные | Уравнение функции |  |
| Диапазон значений переменной | Минимальное и максимлаьное значение по оси Х |
| Параметры отображения графика | Количестов отрисовываемых точек, шаг отрисовки, цвет графика |
| Выходные данные | Построенный график |  |
| Координаты выбранной точки |  |

3.2. Разработка диаграммы классов

Диаграмма классов представлена в приложении А.

3.3 Описание классов

ploatBuilder:

ploatBuilder - это класс, унаследованный от QMainWindow, который представляет собой основное окно приложения для построения графиков.

Конструктор: Принимает указатель на родительский виджет (по умолчанию nullptr).

Деструктор: Очищает ресурсы, выделенные для объекта.

Слоты: on\_buildButton\_clicked(): Обрабатывает нажатие кнопки "Build" и запускает процесс построения графика.

Приватные члены:ui: Указатель на интерфейс ploatBuilder.

plot: Указатель на объект GraphDrawer для отрисовки графика.

pars: Указатель на объект Parser для обработки математических выражений.

func: Указатель на объект Function, являющийся производным от Parser и предназначенным для работы с функциями.

errorMessage: Строка для хранения сообщения об ошибке.

Parser:

Parser – это класс для обработки математических выражений.

Конструктор: Принимает выражение в виде строки.

Методы:

getExpression(): Возвращает текущее математическое выражение.

validateExpression(): Проверяет корректность выражения.

getFunction(): Возвращает функцию, представленную в выражении.

setExpression(): Устанавливает новое выражение.

toPostfix(): Преобразует инфиксное выражение в постфиксное.

calculateFunction(): Вычисляет значение функции.

Защищенные члены: expression: Текущее математическое выражение.

priority: Карта приоритетов операторов.

GraphDrawer:

GraphDrawer - это класс, унаследованный от QGraphicsView, предназначенный для отображения графиков функций.

Конструктор: Принимает указатель на родительский виджет (по умолчанию 0).

Методы:

setMinBoarder(): Устанавливает минимальную границу.

setMaxBoarder(): Устанавливает максимальную границу.

drawPlot(): Рисует график функции.

numberAxes(): Нумерует деления на осях.

getMinBoarder(): Возвращает минимальную границу.

getMaxBoarder(): Возвращает максимальную границу.

deleteItemsFromGroup(): Удаляет элементы из группы.

resizeEvent(): Обрабатывает изменение размера окна.

Приватные члены: scene: Объект сцены для отображения графика.

group\_1, group\_2, group\_3: Группы элементов на сцене.

timer: Таймер для анимации.

minBoarder, maxBoarder: Минимальная и максимальная границы графика.

height, width: Высота и ширина окна.

Function:

Function - это класс, унаследованный от Parser, предназначенный для работы с математическими функциями.

Конструктор: Принимает выражение в виде строки.

Методы:calculateFunction(): Вычисляет значение функции для заданного аргумента x.

1. Cardioid и Deltoid:

- Это классы, представляющие конкретные типы графиков, унаследованные от `Ploat`.

- Каждый из них имеет конструктор, принимающий различные параметры для инициализации объекта.

- Приватные члены (`radius` для `Cardioid` и `size` для `Deltoid`) используются для хранения характеристик графика.

2. Expression:

- Этот класс представляет собой общее выражение и содержит конструктор для инициализации выражения.

- Методы `getExpression` и `setExpression` используются для получения текущего выражения и установки нового соответственно.

3. Function:

- Этот класс представляет собой функцию, являющуюся производной от `Parser` и наследующую `Expression`.

- Конструктор принимает выражение, а методы `calculateFunction` и `calculateMath` используются для вычисления значения функции.

4. GraphDrawer:

- Этот класс представляет собой виджет для отрисовки графиков и унаследован от `QGraphicsView`.

- Методы и слоты используются для управления параметрами графика, настройки отображения и обработки событий.

- Содержит объекты `QGraphicsScene` и таймер для анимации графиков.

5. Parameters:

- Этот класс представляет диалоговое окно для настройки параметров графика.

- Методы `getColor`, `getPointsNum`, и `getdrawStep` используются для получения выбранных пользователем параметров.

- Методы `setColor`, `setPointsNum`, и `setDrawStep` используются для установки параметров.

6. Parser:

- Этот класс отвечает за парсинг математических выражений.

- Метод `calculateInBracers` вычисляет выражение в скобках, а `toPostfix` преобразует инфиксное выражение в постфиксное.

7. Ploat:

- Этот класс представляет общие параметры графика.

- Конструктор и методы используются для инициализации и получения параметров графика.

8. ploatBuilder:

- Это главный класс приложения, унаследованный от `QMainWindow`.

- Методы и слоты обрабатывают события пользователя и управляют графическими данными.

Класс `Ploat`

- \*\*Конструктор:\*\*

- Принимает функции `fXFunction` и `gXFunction` для оси X.

- Границы `minBoarder` и `maxBoarder` определяют диапазон построения графика.

- `drawStep` - шаг для построения графика.

- `graphicsColor` - цвет графика.

- `pointsNum` - количество точек для построения.

- `multyPloats` - опция для множественных графиков.

- \*\*Методы:\*\*

- Геттеры и сеттеры для всех параметров.

- \*\*Приватные члены:\*\*

- `fXFunction` и `gXFunction` - функции на оси X.

- `minBoarder` и `maxBoarder` - границы.

- `drawStep` - шаг.

- `multyPloats` - опция множественных графиков.

- `graphicsColor` - цвет.

- `pointsNum` - количество точек.

Класс `Cardioid`

- \*\*Конструктор:\*\*

- Принимает параметры от `Ploat`, а также добавляет параметр `radius` - радиус кардиоиды.

- \*\*Приватные члены:\*\*

- `radius` - радиус кардиоиды.

Класс `Deltoid`

- \*\*Конструктор:\*\*

- Принимает параметры от `Ploat`, а также добавляет параметр `size` - размер дельтоиды.

- \*\*Приватные члены:\*\*

- `size` - размер дельтоиды.

Класс `Expression`

- \*\*Конструктор:\*\*

- Принимает строку `expression` и устанавливает ее в член класса.

- \*\*Методы:\*\*

- `getExpression()` - возвращает текущее математическое выражение.

- `setExpression(const QString &expression)` - устанавливает новое математическое выражение.

- \*\*Приватные члены:\*\*

- `expression` - текущее математическое выражение.

Класс `Function`.

- \*\*Конструктор:\*\*

- Принимает строку `expression` и передает ее в конструктор базового класса `Parser`.

- \*\*Методы:\*\*

- `calculateFunction(double x, const QString &expression)` - вычисляет значение функции для заданного `x` и выражения.

- `calculateMath(QString func, double exp)` - вычисляет математическую функцию, используя библиотеку `cmath`.

- \*\*Наследование:\*\*

- Класс `Function` наследуется от `Parser`.

- \*\*Приватные члены:\*\*

- Наследует приватные члены от `Parser` и `Expression`.

Класс `GraphDrawer`

- \*\*Конструктор:\*\*

- Принимает указатель на родительский виджет (по умолчанию `nullptr`).

- Инициализирует графические элементы, сцену, таймер и другие параметры.

- \*\*Методы:\*\*

- Множество методов для установки и получения различных параметров и свойств графика.

- `drawPlot` - рисует график функции с заданными параметрами.

- `numberAxes` - размещает числа на осях.

- `saveImage` - сохраняет изображение графика.

- `drawToAxesLane`, `setToAxesNumbers`, `delToAxesLaneNumbers`, `deleteItemsFromGroup` - методы для работы с элементами графика.

- \*\*Приватные члены:\*\*

- `lines` - вектор линий графика.

- `group\_1`, `group\_2`, `group\_3`, `group\_4` - группы элементов на сцене.

- `scene` - сцена для отрисовки.

- `timer` - таймер для анимации.

- \*\*Сигналы и слоты:\*\*

- `slotAlarmTimer` - слот, который вызывается по таймеру.

- \*\*Приватные переменные:\*\*

- Различные параметры, такие как границы, цвет, смещение, масштабирование и другие.

Класс `Parameters` - это диалоговое окно для установки параметров графика.

- \*\*Конструктор:\*\*

- Принимает указатель на родительский виджет (по умолчанию `nullptr`).

- Инициализирует графический интерфейс и параметры.

- \*\*Методы:\*\*

- Методы для получения и установки цвета, количества точек и шага отрисовки.

- `on\_pushButton\_clicked` - слот, вызываемый при нажатии кнопки в диалоговом окне.

- \*\*Сигналы и слоты:\*\*

- `sendData` - сигнал, отправляющий данные о выбранных параметрах.

- \*\*Приватные переменные:\*\*

- `color`, `pointsNum`, `drawStep` - параметры цвета, количества точек и шага отрисовки.

Класс `Parser`.

- \*\*Конструктор:\*\*

- Принимает строку `expression` и передает ее в конструктор базового класса `Expression`.

- \*\*Методы:\*\*

- `calculateInBracers(QStringList tokens)` - вычисляет выражение в скобках.

- `toPostfix(QString expression)` - преобразует инфиксное выражение в постфиксное.

- `getFXExpression()` и `getGXExpression()` - возвращают указатели на объекты `Expression` для fX и gX соответственно.

- `validateToken(QString token)` - проверяет корректность токена.

- \*\*Приватные переменные:\*\*

- `priority` - карта приоритетов операторов.

- `fXExpression` и `gXExpression` - указатели на объекты `Expression`.

- \*\*Вспомогательная функция:\*\*

- `replaceAllOccurrences` - заменяет все вхождения подстроки в строке.

Класс `ploatBuilder`.

- \*\*Конструктор:\*\*

- Принимает указатель на родительский виджет (по умолчанию `nullptr`).

- Инициализирует графический интерфейс, создает объекты `GraphDrawer`, `Function`, `Cardioid`, `Deltoid`, `Parameters` и устанавливает валидаторы.

- \*\*Методы:\*\*

- `mouseMoveEvent` - обрабатывает событие движения мыши.

- Методы для установки цвета графика, передачи данных, обработки событий кнопок и комбобоксов.

- `receiveData` - слот, принимающий данные о параметрах графика из диалогового окна `Parameters`.

- \*\*Приватные переменные:\*\*

- Различные параметры, такие как цвет, количество точек, шаг отрисовки, флаг множественных графиков.

- Объекты окна параметров, графического рисовальщика, функции и геометрических фигур.

- \*\*Слоты:\*\*

- Множество слотов, обрабатывающих события нажатия кнопок и изменения параметров.

- \*\*Сигналы:\*\*

- `receiveData` - сигнал для передачи данных о параметрах графика.

**4. РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ**

В данном разделе я сосредоточился на процессе разработки программных компонентов, которые были ключевыми для функционирования моего приложения по построению графиков. Более детально были рассмотрены два наиболее значимых метода, а именно drawPlot и calculateFunction, а также представлена шаг за шагом их реализация.

Передо мной стояла задача создать эффективные алгоритмы, способные преобразовывать математические выражения в подходящий для вычислений формат и проводить точные вычисления значений функций. Эти программные модули стали ключевыми строительными блоками моего приложения, определяя его точность и функциональность.

4.1. Разработка блок-схем алгоритмов (два наиболее важных метода).

.drawPlot в классе GraphDrawer:

- Этот метод отвечает за отрисовку графика на виджете. Он является ключевым для визуализации данных и предоставляет пользователю актуальное представление функции на экране.

calculateFunction в классе Function – вычисляет значение функции для заданного значения переменной `x` и математического выражения. Это ключевая операция при построении графика, так как она определяет, каким образом график будет отображаться в различных точках.

Блок-схемы данных алгоритмов представленны в приложении Б.

4.2. Разработка алгоритмов (описание алгоритмов по шагам, для двух

методов).

Метод `drawPlot` класса `GraphDrawer` отвечает за отрисовку графика функции на виджете `QGraphicsView`.

Входные данные: нет.

выходные данные: нет.

1. Начало

2. Проверить состояние чекбокса если установлен то очистить график, если нет, ничего не делать.

3. Инициализация переменных:

- QPen pen(Qt::blue): создание пера для рисования с синим цветом.

- error: строка для хранения сообщения об ошибке.

- yVector: вектор для хранения значений функции.

4. Вычисление параметров масштабирования для осей X и Y.

5. Расчет шага и отступов для осей X и Y:

6. Вычисление значений y.

6. Обработка ошибки когда график по оси Y может отображаться некорректно.

7. Обработка ситуации когда график является линий.

8. Вычисление минимальной и максимальной границы для оси Y.

9. Настройка параметров пера для отрисовки осей c последующей отрисовкой

10. Вычисляем значения ‘x’ и отрисовываем график по точкам.

11. Вызов функции “numberAxes” для отрисовки числовых меток на осях.

12. Конец

Этот метод обеспечивает отображение графика функции на виджете с учетом масштабирования и параметров отрисовки.

Метод calculateFunction в классе Function.

Входные данные: Выражение в виде польской записи, значение неизвестной переменной(x).

выходные данные: Результат вычисления выражения.

1. Начало

2. Инициализация переменных:

- mathExpr: логическая переменная, определяющая наличие математического выражения в строке.

- result: переменная, в которую будет записан результат вычислений.

- error: строка для хранения сообщения об ошибке.

- num: строка для временного хранения числа.

- `variablesStack`: стек для переменных и промежуточных результатов.

3. Разделение выражения на токены используя регулярное выражение с параметром “\\s\* \\s\*”:

4. Проход по токенам.

5. Если токен является математической функцией то производится расчет выражения с помощью функции “calcualteMath” и записывается в “variablesStack”.

6. Если токен является опертором то происходит вычисление выражения в зависимости от оператора с последующей записью в “variablesStack”.

7. Иначе если токен является ‘x’ записываем в “variablesStack” значение ‘x’ переданное пользователем, в люблм друом случае записываем данный токен в “variablesStack”.

8. Если токены кончились проверяем не находится ли на вершине “variablesStack” математических функций и если находятся вычисляем иначе переходим к шагу 4.

9. Возвращаем результат

10. Конец

Этот алгоритм эффективно обрабатывает математические выражения, включая функции и переменные, в постфиксной форме, обеспечивая точные результаты.

**5. РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ.**

В рамках раюоты бный на рисунке был разработанн итерфейс программы представленный на рисунке 5.1.

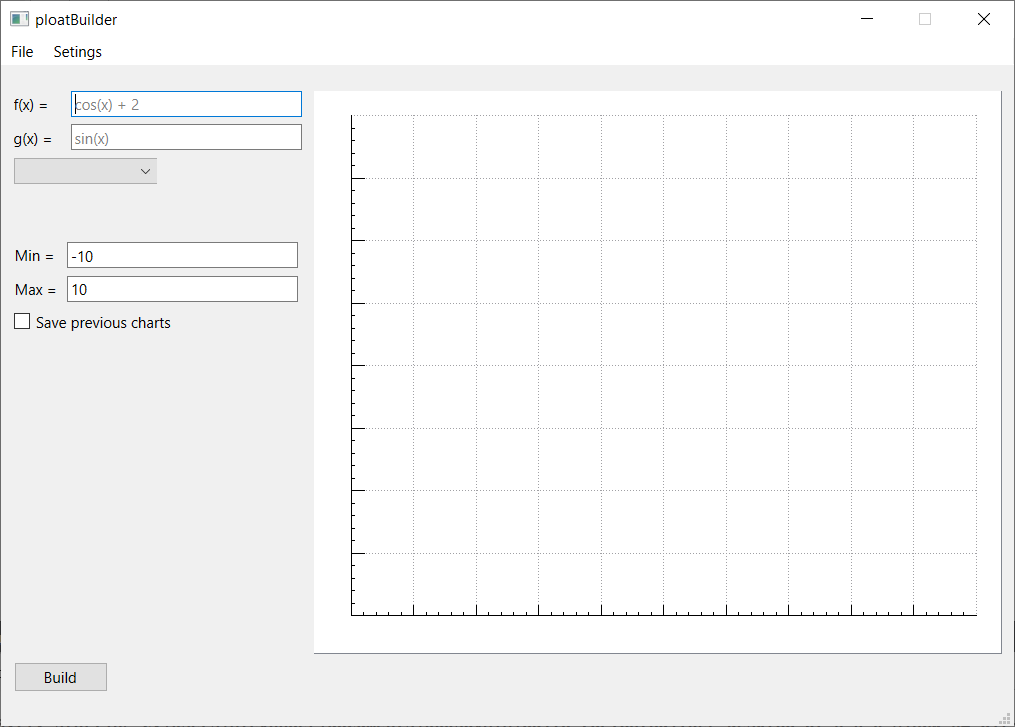


Рисунок 5.1 – Интрефейс программы

На интерфейсе представленно:

1. Поля для ввода функций:

* f(x) – функций для рассчета y (обязательная).
* g(x) – функций для рассчета x (опциональная).

1. Выпадающее окно с различными примерами функций
2. Поля для установки минимального(min) и максимального(max) значения по x.
3. Флажок “Save previous charts” при активации которого сохраняются ранее отрисованные графики. (рис 5.3)
4. Кнопка “Build” по нажатию которой происходит отрисовка графика. (рис. 5.2).

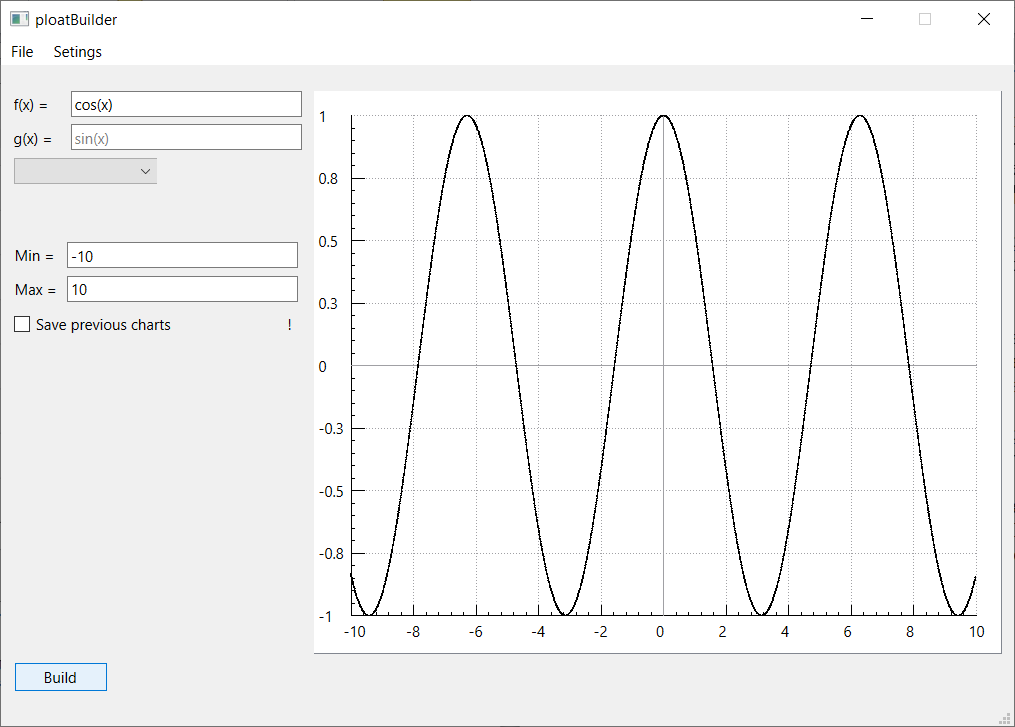


Рисунок 5.2 – Отрисованный график

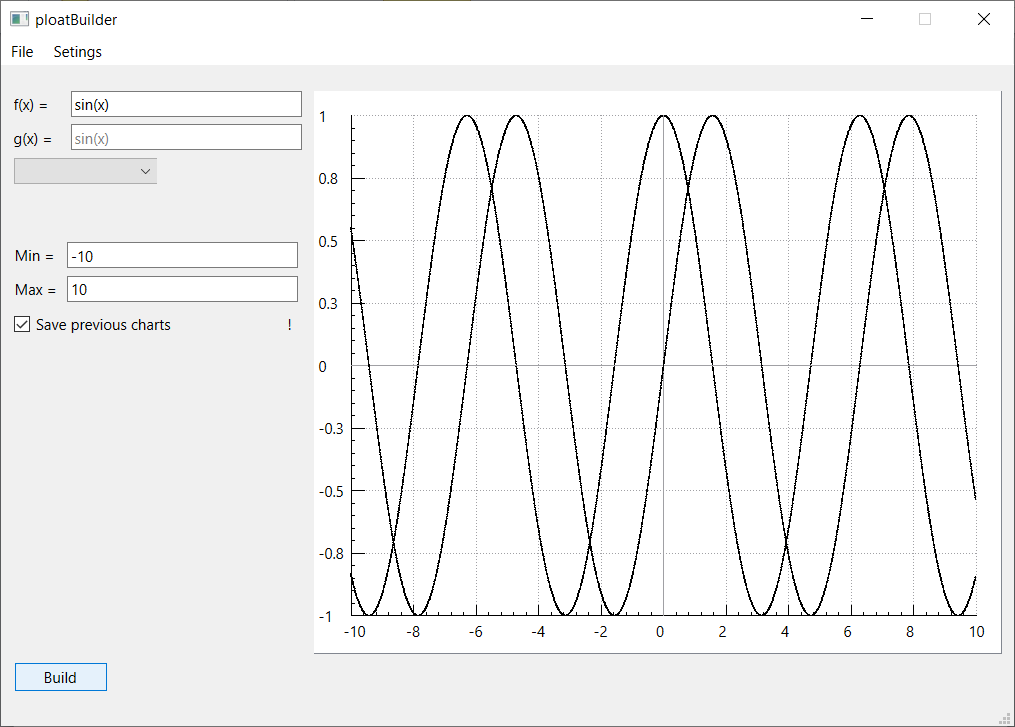


Рисунок 5.3 – Сохранение ранее отрисованных графиков

Переходе по “Setings 🡪 parameters” открывается окно с найстройками графика (рис. 5.4)

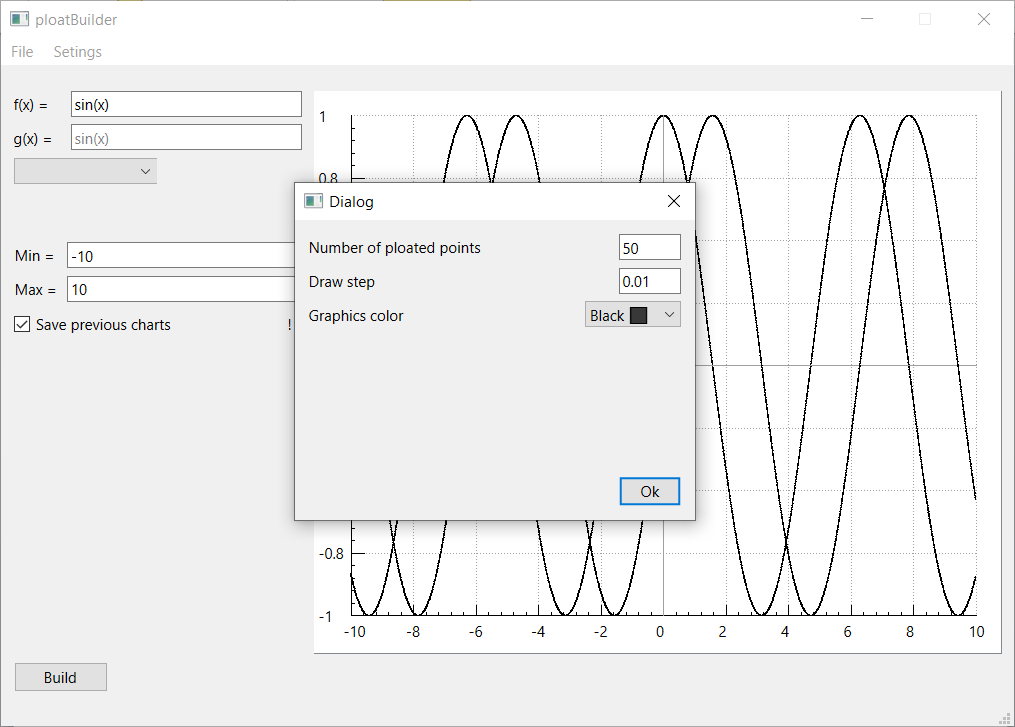


Рисунок 5.4 – Окно настроек графика

**ЗАКЛЮЧЕНИЕ**

В заключение проекта хочу подчеркнуть важность достигнутых результатов и усилий, вложенных в разработку графического приложения для построения математических графиков. В ходе работы было достигнуто несколько ключевых моментов:

1. Функциональность и удобство использования:

- Создано интуитивно понятное приложение с простым и понятным интерфейсом, что делает построение графиков доступным даже для пользователей без специальных навыков.

2. Стабильность и безопасность:

- Реализованы механизмы обработки ошибок, что обеспечивает стабильную работу программы и предотвращает возможные проблемы при отображении графиков.

3. Масштабирование и адаптация:

- Разработан механизм автоматического масштабирования, учитывающий форму и размер виджета, что обеспечивает корректное отображение графиков при любых условиях.

5. Алгоритмы и оптимизация:

- Реализованы эффективные алгоритмы для преобразования выражений и вычисления значений функций, что улучшило общую производительность приложения.

Этот проект был увлекательным вызовом, и результаты, достигнутые в ходе работы, приносят удовлетворение. Программа готова к использованию и может быть полезной для тех, кто интересуется визуализацией математических функций.

**7. ЛИТЕРАТУРА**

[1] Бьерн Страуструп. Язык программирования С++. Пер. с англ.- М.: «Издательство БИНОМ», 2004. – 1098 с.

2. Скляров В.А. Язык С++ и объектно-ориентированное программирование. Мн.: Выш.шк., 1997г. – 478 с.: ил.

3. Луцик Ю.А., Ковальчук А.М., Лукьянова И.В. Объектноориентированное программирование на языке С++. Мн.: БГУИР, 2003. – 203 с.: ил.

**ПРИЛОЖЕНИЕ А**

**(**обязательное**)**

Диаграмма классов