|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |

**Институт информационных технологий (ИТ)**

**Кафедра инструментального и прикладного программного обеспечения (ИиППО)**

**КУРСОВАЯ РАБОТА**

по дисциплине: «Объектно-ориентированное программирование»

по профилю: Информатизация организаций

направления профессиональной подготовки: Прикладная информатика, бакалавриат

Тема: «Разработка программы для построения графиков функций»

Студент (ф.и.о. полностью): Кузин Максим Валерьевич

Группа: ИНБО-04-18

Работа представлена к защите\_\_\_\_\_\_\_\_\_\_\_(дата)\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

(подпись и ф.и.о. студента)

Руководитель: Хлебникова Валерия Леонидовна ассистент кафедры ИиППО

Работа допущена к защите\_\_\_\_\_\_\_\_\_\_\_(дата)\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

(подпись ф.и.о. руководителя)

Оценка по итогам защиты: \_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /

(подписи, дата, ф.и.о., должность, звание, уч. степень двух преподавателей, принявших защиту)

М. МИРЭА. 2019г.

# СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 2](#_Toc8806154)

[Введение 4](#_Toc8806155)

[Техническое задание 5](#_Toc8806156)

[1.1 Общие положения 5](#_Toc8806157)

[1.1.1 Полное наименование системы и ее условное обозначение 5](#_Toc8806158)

[1.1.2 Номер договора (контракта) 5](#_Toc8806159)

[1.1.3 Наименования организации-заказчика и организаций-участников работ 5](#_Toc8806160)

[1.1.4 Перечень документов, на основании которых создается система 5](#_Toc8806161)

[1.1.5 Плановые сроки начала и окончания работы по созданию системы 5](#_Toc8806162)

[1.1.6 Источники и порядок финансирования работ 5](#_Toc8806163)

[1.1.7 Порядок оформления и предъявления заказчику результатов работ по созданию системы 5](#_Toc8806164)

[1.1.8 Перечень нормативно-технических документов, методических материалов, использованных при разработке ТЗ 6](#_Toc8806165)

[1.2 НАЗНАЧЕНИЕ И ЦЕЛИ СОЗДАНИЯ СИСТЕМЫ 6](#_Toc8806166)

[1.2.1 Назначение системы 6](#_Toc8806167)

[1.2.2 Цели создания системы 6](#_Toc8806168)

[1.3 ХАРАКТЕРИСТИКА ОБЪЕКТА АВТОМАТИЗАЦИИ 6](#_Toc8806169)

[1.4 ТРЕБОВАНИЯ К СИСТЕМЕ 6](#_Toc8806170)

[1.4.1 Требования к системе в целом 6](#_Toc8806171)

[1.4.1.1 Требования к структуре и функционированию системы 7](#_Toc8806172)

[1.4.1.1.1 Перечень подсистем, их назначение и основные характеристики 7](#_Toc8806173)

[1.4.1.1.2 Требования к способам и средствам связи для информационного обмена между компонентами системы 7](#_Toc8806174)

[1.4.1.2 Требования к численности и квалификации персонала системы 8](#_Toc8806175)

[1.4.1.3 Показатели назначения 8](#_Toc8806176)

[1.4.1.4 Требования к надежности 8](#_Toc8806177)

[1.4.1.5 Требования к безопасности 8](#_Toc8806178)

[1.4.1.6 Требования к эргономике и технической эстетике 8](#_Toc8806179)

[1.4.1.7 Требования к транспортабельности для подвижных АС 8](#_Toc8806180)

[1.4.1.8 Требования к эксплуатации, техническому обслуживанию, ремонту и хранению компонентов системы 8](#_Toc8806181)

[1.4.1.9 Требования к защите информации от несанкционированного доступа 9](#_Toc8806182)

[1.4.1.10 Требования по сохранности информации при авариях 9](#_Toc8806183)

[1.4.1.11 Требования к защите от влияния внешних воздействий 9](#_Toc8806184)

[1.4.1.12 Требования к патентной частоте 9](#_Toc8806185)

[1.4.1.13 Требования по стандартизации и унификации 9](#_Toc8806186)

[1.4.1.14 Дополнительные требования 9](#_Toc8806187)

[1.4.2 Требования к функциям (задачам), выполняемым системой 9](#_Toc8806188)

[1.4.3 Требования к видам обеспечения 10](#_Toc8806189)

[1.4.3.1 Требования к математическому обеспечению системы 10](#_Toc8806190)

[1.4.3.2 Требования информационному обеспечению системы 10](#_Toc8806191)

[1.4.3.3 Требования к лингвистическому обеспечению системы 10](#_Toc8806192)

[1.4.3.4 Требования к программному обеспечению системы 10](#_Toc8806193)

[1.4.3.5 Требования к техническому обеспечению 10](#_Toc8806194)

[1.4.3.6 Требования к метрологическому обеспечению 10](#_Toc8806195)

[1.4.3.7 Требования к организационному обеспечению 11](#_Toc8806196)

[1.4.3.8 Требования к методическому обеспечению 11](#_Toc8806197)

[1.5 СОСТАВ И СОДЕРЖАНИЕ РАБОТ ПО СОЗДАНИЮ (РАЗВИТИЮ) СИСТЕМЫ 11](#_Toc8806198)

[1.6 ПОРЯДОК КОНТРОЛЯ И ПРИЕМКИ СИСТЕМЫ 12](#_Toc8806199)

[1.6.1 Виды, состав, объем и методы испытаний системы 12](#_Toc8806200)

[1.6.2 Общие требования к приемке работ по стадиям 12](#_Toc8806201)

[1.6.3 Статус приемочной комиссии 12](#_Toc8806202)

[1.7 ТРЕБОВАНИЯ К СОСТАВУ И СОДЕРЖАНИЮ РАБОТ ПО ПОДГОТОВКЕ ОБЪЕКТА АВТОМАТИЗАЦИИ К ВВОДУ СИСТЕМЫ В ДЕЙСТВИЕ 12](#_Toc8806203)

[1.8 ТРЕБОВАНИЯ К ДОКУМЕНТИРОВАНИЮ 13](#_Toc8806204)

[Проектирование и разработка программы 14](#_Toc8806205)

[1.9 Проектирование программы 14](#_Toc8806206)

[1.9.1 Функциональные требования к программе 14](#_Toc8806207)

[1.9.2 Проектирование классов 14](#_Toc8806208)

[1.9.3 Разработка программы 17](#_Toc8806209)

[1.9.3.1 Описание проектного решения 18](#_Toc8806210)

[1.9.3.2 Тестирование программы 18](#_Toc8806211)

[Руководство пользователя 19](#_Toc8806212)

[Заключение 28](#_Toc8806213)

[Список литературы 29](#_Toc8806214)

[Приложение №1. Исходный код программы 30](#_Toc8806215)

[Приложение №2. UML-диаграмма 40](#_Toc8806216)

# Введение

Программным продуктом, реализованным в данной курсовой работе, является «Программа для построения графиков функций» - программа, которая позволяет строить определенный пользователем тип графика с указанными координатами, а также изменять цент оси координат.

Языком программирования, выбранным для реализации, стал объектно-ориентированный язык программирования C++. C++ один из самых широко распространенных языков. Преимуществами данного языка являются его мощность и гибкость, обусловленные такими составляющими, как указатели, шаблоны и множественное наследование.

Унифицированный язык моделирования, использованный в данной курсовой работе, представляет собой графический язык, включающий в себя множество различных диаграмм, помогающий специалистам по системному анализу создавать алгоритмы, а программистам в принципах работы программы. UML является мощным инструментом, позволяющим сделать процесс программирования более легким и эффективным.

# Техническое задание

## Общие положения

### Полное наименование системы и ее условное обозначение

Полное наименование: Программа для построения графиков функций.

### Номер договора (контракта)

Работа выполняется на основании Задания на выполнение курсовой работы.

### Наименования организации-заказчика и организаций-участников работ

Заказчик: РТУ МИРЭА

Исполнитель: Кузин Максим Валерьевич студент группы ИНБО-04-18

### Перечень документов, на основании которых создается система

Документы, на основании которых создается система:

* Учебный план (№1382.9 09.03.03 ПИ Очн, пс, 4 года (8 сессий)

УМУ\_09.03.03\_ИО\_ИИТ\_2018).

### Плановые сроки начала и окончания работы по созданию системы

Плановый срок начала работы по созданию системы: 26 февраля 2019 года.

Плановый срок окончания работы по созданию системы: 20 мая 2019 года.

### Источники и порядок финансирования работ

Разработка Системы финансируется РТУ МИРЭА, а, в частности,

Федеральным бюджетом Российской Федерации.

### Порядок оформления и предъявления заказчику результатов работ по созданию системы

Система передаётся в виде готового программного обеспечения на базе средств вычислительной техники Заказчика в сроки, установленные в п. 1.5 данного ТЗ. Приёмка осуществляется в составе Исполнителя и уполномоченных представителей Заказчика.

Порядок предъявления системы, её испытаний и окончательной приёмки

определён в п.7 настоящего ТЗ. Совместно с предъявлением Системы Исполнителем производится сдача разработанного комплекта документации согласно п. 9 данного ТЗ.

### Перечень нормативно-технических документов, методических материалов, использованных при разработке ТЗ

При разработке системы Исполнитель должен руководствоваться

требованиями следующих документов:

* ГОСТ 34.601-90 Комплекс стандартов Автоматизированные системы.

Стадии создания;

* Методические указания по выполнению курсовой работы для

бакалавров.

## НАЗНАЧЕНИЕ И ЦЕЛИ СОЗДАНИЯ СИСТЕМЫ

### Назначение системы

Система предназначена для построения графика указанной функции с указанными коэффициентами, а также построения графика для измененных коэффициентов.

### Цели создания системы

Цели создания Системы:

* демонстрация прикладного программного обеспечения;
* сдача курсовой работы;
* приобретение опыта разработки приложений средней сложности;
* изучение объектно-ориентированного программирования;
* приобретение навыков написания технической документации.

## ХАРАКТЕРИСТИКА ОБЪЕКТА АВТОМАТИЗАЦИИ

Объектом автоматизации данной системы является построение графика по указанным коэффициентам..

## ТРЕБОВАНИЯ К СИСТЕМЕ

### Требования к системе в целом

#### Требования к структуре и функционированию системы

##### Перечень подсистем, их назначение и основные характеристики

Разрабатываемая система, должна включать в себя несколько классов для построения графика:

* Function – абстрактный класс для хранения центра оси и его управлением координат и методов общих для всех типов графиков;
* Const – класс производный от Function, который хранит значение константы и строит указанный тип графика.
* Pryamaya – класс производный от Function, который хранит коэффициенты k и b и строит указанный тип графика.
* Sin – класс производный от Function, который строит указанный тип графика.
* Cos – класс производный от Function, который строит указанный тип графика.
* Tg – класс производный от Function, который строит указанный тип графика.
* Ctg – класс производный от Function, который строит указанный тип графика.
* Arcsin – класс производный от Function, который строит указанный тип графика.
* Arccos – класс производный от Function, который строит указанный тип графика.
* Exp – класс производный от Function, который строит указанный тип графика.
* Ax – класс производный от Function, который хранит значение параметра а и строит указанный тип графика.

##### Требования к способам и средствам связи для информационного обмена между компонентами системы

Связь между компонентами системы обеспечивается функциями, определёнными в компонентах системы.

#### Требования к численности и квалификации персонала системы

Для работы разрабатываемой системы необходимо достаточно одного человека. К работе с системой должны допускаться сотрудники, имеющие навыки работы на персональном компьютере, ознакомленные с правилами эксплуатации и прошедшие обучение работе с системой.

#### Показатели назначения

Программа должна строить график. Должны быть реализованы функции по управлению. Требования ко времени выполнения данных функций не предъявляются.

#### Требования к надежности

Отсутствуют.

#### Требования к безопасности

Отсутствуют.

#### Требования к эргономике и технической эстетике

Графический интерфейс Системы должен разрабатываться на основе офтальмологических исследований, с использованием сочетаний цветов, комфортных для глаз человека.

#### Требования к транспортабельности для подвижных АС

Разрабатываемая Система должна запускаться на любых устройствах под управлением операционной системы не старше чем Windows 10 без процедуры установки или повторной компиляции данной Системы.

#### Требования к эксплуатации, техническому обслуживанию, ремонту и хранению компонентов системы

Для корректной работы разрабатываемая Система должна храниться в отдельном каталоге со всеми своими компонентами. Особых требований по обслуживанию данная Система не имеет.

#### Требования к защите информации от несанкционированного доступа

Требования к какой-либо защите информации в данной системе не предъявляются.

#### Требования по сохранности информации при авариях

Отсутствуют

#### Требования к защите от влияния внешних воздействий

Защита от влияния внешних воздействий должна обеспечиваться средствами программно-технического комплекса Заказчика.

#### Требования к патентной частоте

Данная Система должна являться интеллектуальной собственностью Исполнителя и быть патентно-чистой по отношению ко всем странам мира.

#### Требования по стандартизации и унификации

Система должна быть реализована методами объектно-ориентированного программирования. Взаимодействие пользователей с прикладным программным обеспечением, входящим в состав системы должно осуществляться посредством визуального консольного приложения. Интерфейс системы должен быть понятным и удобным, не должен быть перегружен графическими элементами и должен обеспечивать быстрое отображение экранных форм. Навигационные элементы должны быть выполнены в удобной для пользователя форме. Интерфейс должен соответствовать современным эргономическим требованиям и обеспечивать удобный доступ к основным функциям и операциям системы.

#### Дополнительные требования

Дополнительные требования не предъявляются.

### Требования к функциям (задачам), выполняемым системой

Система должна выполнять следующие функции:

* Строить ось координат;
* Строить сами графики;
* Перестраивать графики с другим центром координат.

### Требования к видам обеспечения

Отсутствуют.

#### Требования к математическому обеспечению системы

Отсутствуют.

#### Требования информационному обеспечению системы

Система будет получать входную информацию в виде значений коэффициентов в виде сигналов нажатия кнопок клавиатуры.

#### Требования к лингвистическому обеспечению системы

Для создания Системы должен быть использован низкоуровневый язык программирования C++. Для взаимодействия Системы с пользователем, руководства по использованию Системы и документации Системы должен использоваться русский и английский языки.

#### Требования к программному обеспечению системы

Данная Система должна являться кроссплатформенным приложением и работать на всех операционных системах с поддержкой графического интерфейса. Для корректной работы Системы не требуется загрузки сторонних приложений или библиотек.

#### Требования к техническому обеспечению

Используемое при разработке программное обеспечение и библиотеки программных кодов должны иметь широкое распространение, быть общедоступными и использоваться в промышленных масштабах. Базовой программной платформой должна являться операционная система MS Windows.

#### Требования к метрологическому обеспечению

Требования к метрологическому обеспечению не предъявляется.

#### Требования к организационному обеспечению

Организационное обеспечение системы должно быть достаточным для эффективного выполнения персоналом возложенных на него обязанностей при осуществлении автоматизированных и связанных с ними неавтоматизированных функций системы.

К работе с системой должны допускаться сотрудники, имеющие навыки работы на персональном компьютере, ознакомленные с правилами эксплуатации и прошедшие обучение работе с системой.

#### Требования к методическому обеспечению

Данная система должна поставляться с определённым пакетом документации, состоящем из:

* технического задания (ГОСТ);
* пояснительной записки (ГОСТ);
* задания на выполнение курсовой работы;
* руководства пользователя.

## СОСТАВ И СОДЕРЖАНИЕ РАБОТ ПО СОЗДАНИЮ (РАЗВИТИЮ) СИСТЕМЫ

Таблица 1 Содержание Этапов работ

|  |  |  |
| --- | --- | --- |
| № этапа | Содержание работ | Срок |
| 1 | Разработка структуры входных и выходных данных | 27.02.19-03.03.19 |
| 2 | Разработка алгоритма решения задачи | 04.03.19-17.03.19 |
| 3 | Разработка структуры программы | 18.03.19-1.04.19 |
| 4 | Разработка тестов | 01.04.19-07.04.19 |
| 5 | Написание текста программы | 08.04.19-28.04.19 |
| 6 | Отладка программы | 29.04.19-04.05.19 |
| 7 | Написание и оформление отчета о выполненной в виде работе пояснительной записки | 05.05.19-13.05.19 |

## ПОРЯДОК КОНТРОЛЯ И ПРИЕМКИ СИСТЕМЫ

### Виды, состав, объем и методы испытаний системы

Для разработанной системы будут сформированы тесты для проверки функций сортировки и поиска. Тестовые входные записи будут иметь корректные выходные данные, сравнение выходных данных системы с которыми покажет правильность реализации алгоритмов.

### Общие требования к приемке работ по стадиям

Разработка данной Системы делится на шесть стадий:

* получение задания на выполнение курсовой работы;
* составление и согласование технического задания;
* создание и тестирование Системы Исполнителем;
* написание технической документации для Системы;
* демонстрация Системы Заказчику;
* защита курсовой работы.

### Статус приемочной комиссии

Статус приемочной комиссии определяется Заказчиком до проведения испытаний.

## ТРЕБОВАНИЯ К СОСТАВУ И СОДЕРЖАНИЮ РАБОТ ПО ПОДГОТОВКЕ ОБЪЕКТА АВТОМАТИЗАЦИИ К ВВОДУ СИСТЕМЫ В ДЕЙСТВИЕ

В ходе выполнения проекта на объекте автоматизации требуется выполнить работы по подготовке к вводу системы в действие. При подготовке к вводу в эксплуатацию системы должен обеспечить выполнение следующих работ:

* Определить подразделение и ответственных должностных лиц, ответственных за внедрение и проведение опытной эксплуатации системы;
* Обеспечить присутствие пользователей на обучении работе с системой, проводимом Исполнителем;
* Обеспечить соответствие помещений и рабочих мест пользователей системы в соответствии с требованиями, изложенными в настоящем ЧТЗ;
* Обеспечить выполнение требований, предъявляемых к программно-техническим средствам, на которых должно быть развернуто программное обеспечение;
* Совместно с Исполнителем подготовить план развертывания системы на технических средствах Заказчика;
* Привести поступающую в систему информацию (в соответствии с требованиями к информационному и лингвистическому обеспечению) к виду, пригодному для обработки с помощью ЭВМ;
* Провести опытную эксплуатацию системы.

Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие, включая перечень основных мероприятий и их исполнителей должны быть уточнены на стадии подготовки рабочей документации и по результатам опытной эксплуатации.

## ТРЕБОВАНИЯ К ДОКУМЕНТИРОВАНИЮ

Кроме создания работоспособной Системы Исполнитель должен составить пакет документации, состоящий из:

* технического задания;
* пояснительной записки.

# Проектирование и разработка программы

## Проектирование программы

### Функциональные требования к программе

Функциональные требования к программе описаны в пункте 2.4.2 Технического задания.

### Проектирование классов

Программа написана на языке программирования C++. Для реализации функциональной части системы были реализованы следующие классы (Рис. 1):

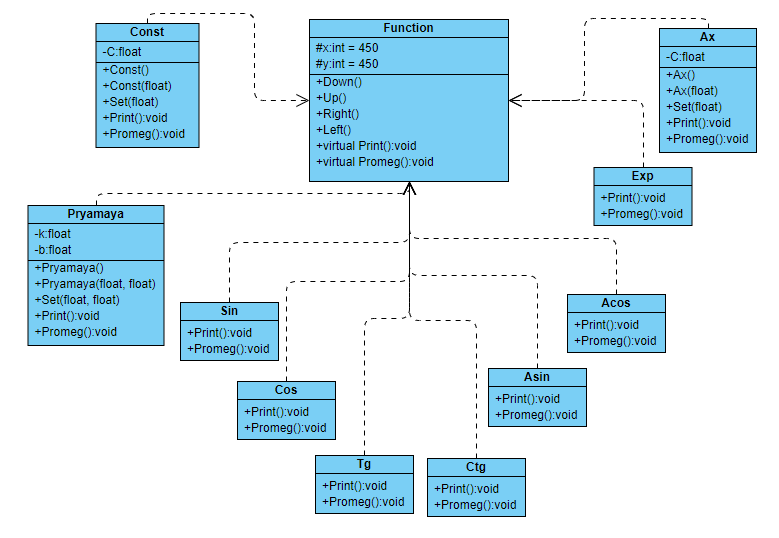


Рис. 1 Общая UML диаграмма

1. Класс Function (Рис. 2)

Абстрактный класс, описывающий центр оси координат. Также в нем присутствую методы управления осью координат (Up(), Down(), Left(), Right()) и виртуальные методы построения графика(Print()) и выведения промежутка(Promeg());

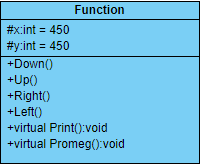


Рис. 2 Класс Function

1. Const (Рис. 3)

Класс наследуемый от Function предназначенный для построения графика функции f(x)=C, хранящий в себе значение коэффициента константы(С), 2 конструктора, метод установки коэффициента (Set()),а также методы построения графика(Print()) и выведения промежутка(Promeg());

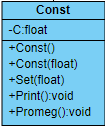


Рис. 3 Класс Const

1. Pryamaya (Рис. 4)

Класс наследуемый от Function предназначенный для построения графика функции f(x)=kx+b, хранящий в себе значение коэффициентов констант (k, b), 2 конструктора, метод установки коэффициентов (Set()),а также методы построения графика(Print()) и выведения промежутка(Promeg());

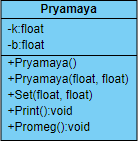


Рис. 4 Класс Pryamaya

1. Sin (Рис.5)

Класс наследуемый от Function предназначенный для построения графика функции f(x)=sin(x), хранящий в себе методы построения графика (Print()) и выведения промежутка(Promeg());



Рис. 5 Класс Sin

1. Cos (Рис.6)

Класс наследуемый от Function предназначенный для построения графика функции f(x)=cos(x), хранящий в себе методы построения графика (Print()) и выведения промежутка(Promeg());



Рис. 6 Класс Cos

1. Tg (Рис.7)

Класс наследуемый от Function предназначенный для построения графика функции f(x)=tg(x), хранящий в себе методы построения графика (Print()) и выведения промежутка (Promeg());



Рис. 7 Класс Tg

1. Ctg (Рис.8)

Класс наследуемый от Function предназначенный для построения графика функции f(x)=ctg(x), хранящий в себе методы построения графика (Print()) и выведения промежутка (Promeg());



Рис. 8 Класс Ctg

1. Asin (Рис.9)

Класс наследуемый от Function предназначенный для построения графика функции f(x)=arcsin(x), хранящий в себе методы построения графика (Print()) и выведения промежутка (Promeg());



Рис. 9 Класс Asin

1. Acos (Рис.10)

Класс наследуемый от Function предназначенный для построения графика функции f(x)=arccos(x), хранящий в себе методы построения графика(Print()) и выведения промежутка(Promeg());



Рис. 10 Класс Acos

1. Exp (Рис.11)

Класс наследуемый от Function предназначенный для построения графика функции f(x)=e^x, хранящий в себе методы построения графика(Print()) и выведения промежутка(Promeg());



Рис. 11 Класс Exp

1. Ax (Рис.12)

Класс наследуемый от Function предназначенный для построения графика функции f(x)=a^x, хранящий в себе значение коэффициента константы(a), 2 конструктора, метод установки коэффициента (Set()),а также методы построения графика(Print()) и выведения промежутка(Promeg());

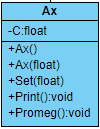


Рис. 12 Класс Ax

### Разработка программы

#### Описание проектного решения

После запуска программы открывается консольное приложение, на котором пользователю предлагается выбрать один из десяти типов графиков (z – константа; x – прямая; c - синус; v - косинус; b - тангенс; n - котангенс; m - арксинус; g - арккосинус; h- экспонента ; j – значение а в степени х). После нажатия нужной клавиши указателю на базовый класс Function присваивается ссылка на соответствующий производный класс (z – Const, x – Pryamaya, c – Sin, v – Cos, b – Tg, n – Ctg, m – Asin, g – Acos, h – Exp, j - Ax).

После выбора нужного пользователю типа программа, если это необходимо запрашивает значение указанных констант с помощью метода Set().

Далее консольное приложение отображает кнопки с помощью которых можно управлять графиком, изменяя отрезки на которых расположен график (w – метод Up() , a – метод Left(), s – метод Down(), d – метод Right()), перестраивание его заново(r – метод Draw()), также можно заново показать информацию по управлению(q) или закрыть программу(~).

#### Тестирование программы

Программа была протестирована путем проверки всех ее функциональных возможностей. Все выходные данные соответствуют ожиданиям. Сбоев при работе программы не наблюдалось.

# Руководство пользователя

Программа запускается через exe-файл в папке со всеми необходимыми компонентами.

После запуска программы открывается консольное приложение, на котором пользователю предлагается выбрать один из десяти типов графиков( z – константа; x – прямая; c - синус; v - косинус; b - тангенс; n - котангенс; m - арксинус; g - арккосинус; h- экспонента ; j – значение а в степени х), путем нажатия нужной клавиши на клавиатуре. (Рис. 13).

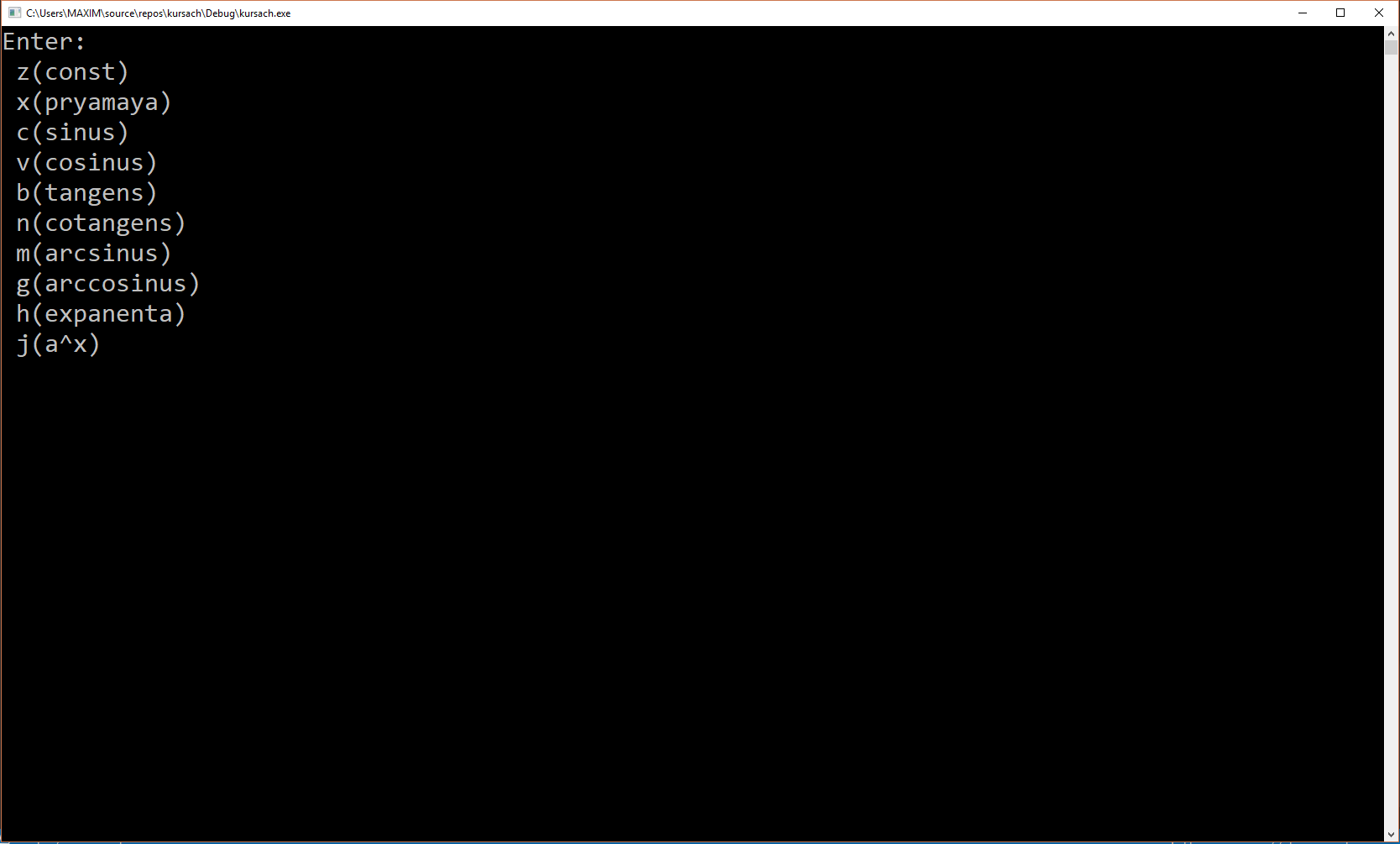


Рис. 13 Консольное приложение запрашивает выбор типа графика

После выбора нужного пользователю типа программа, если это необходимо запрашивает значение указанных констант. (Рис. 14).

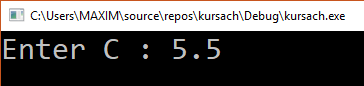


Рис. 14 Консольное приложение запрашивает значение коэффициентов

Далее консольное приложение отображает клавиши с помощью которых можно управлять графиком. (Рис. 15).

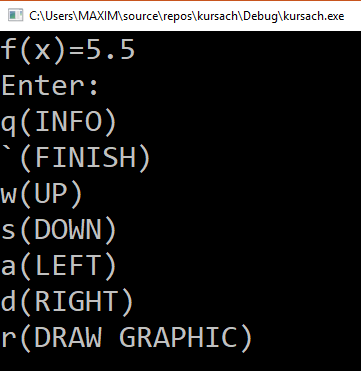


Рис. 15 Консольное приложение отображает систему управления

При нажатии на r программа строит сам график. (Рис. 16).

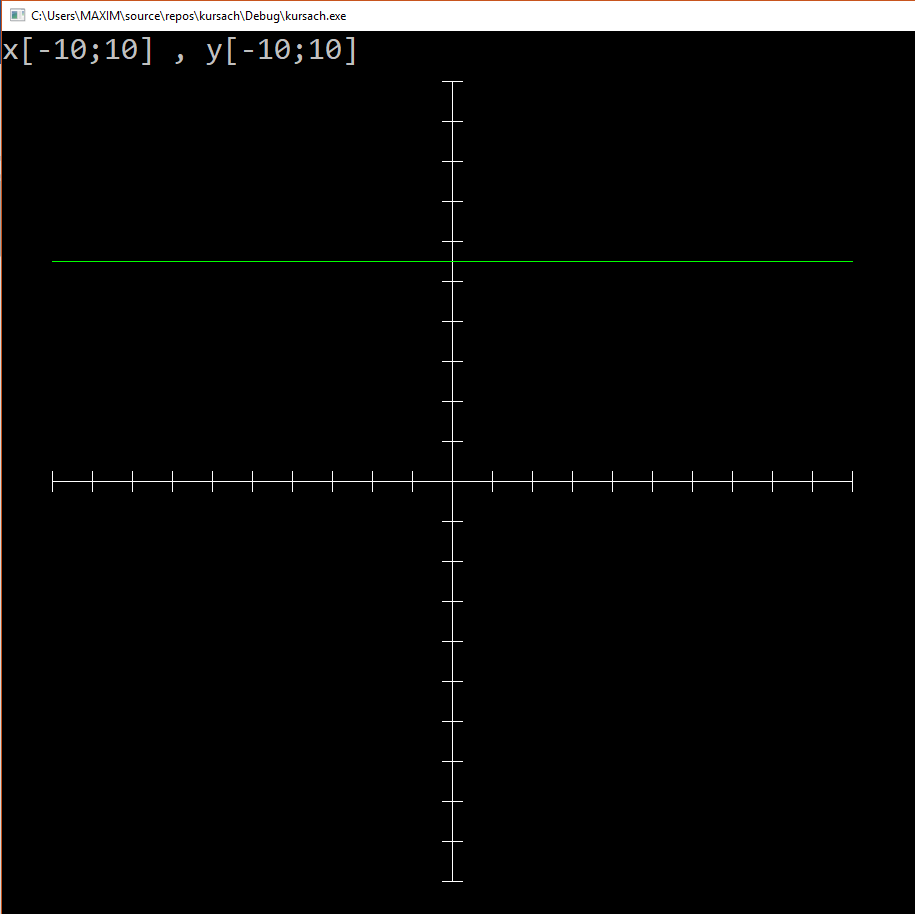


Рис. 16 построенный график

С помощью кнопок w a s d можно управлять центром оси координат.(Рис.17)

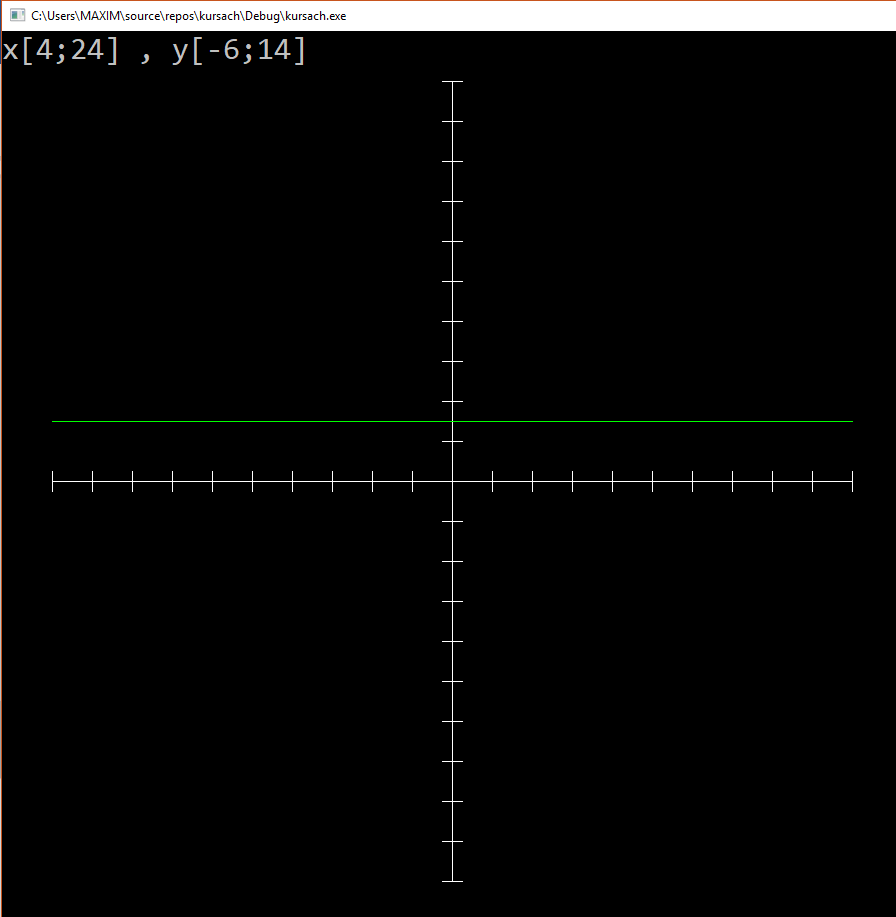


Рис. 17 построенный график после изменения центра координат

При нажатии на q график исчезает и появляется информация по системе управления (Рис. 18).

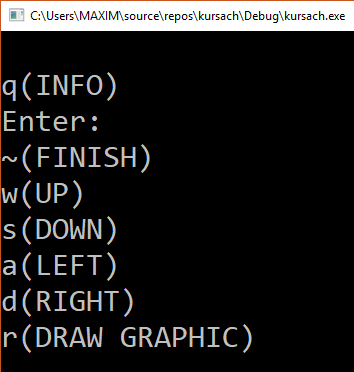


Рис. 18 Информация по системе управления

Далее представлены изображения других типов графиков (Рис. 19).

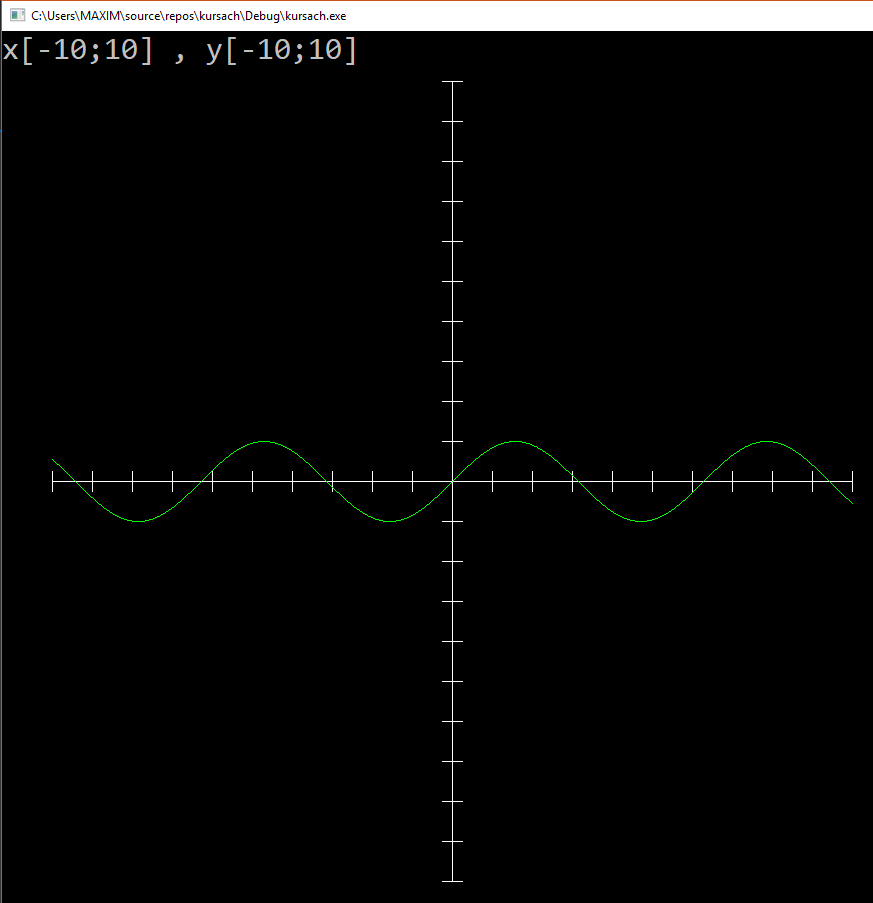


Рис. 19 Построение синуса

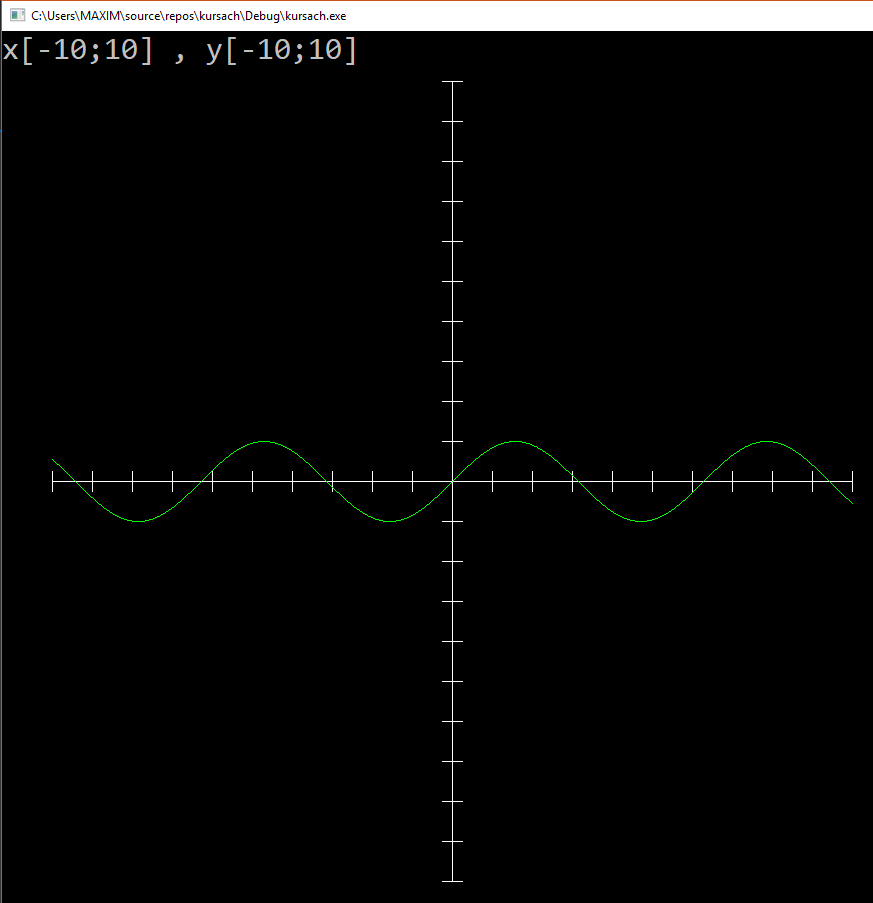


Рис. 20 Построение синуса

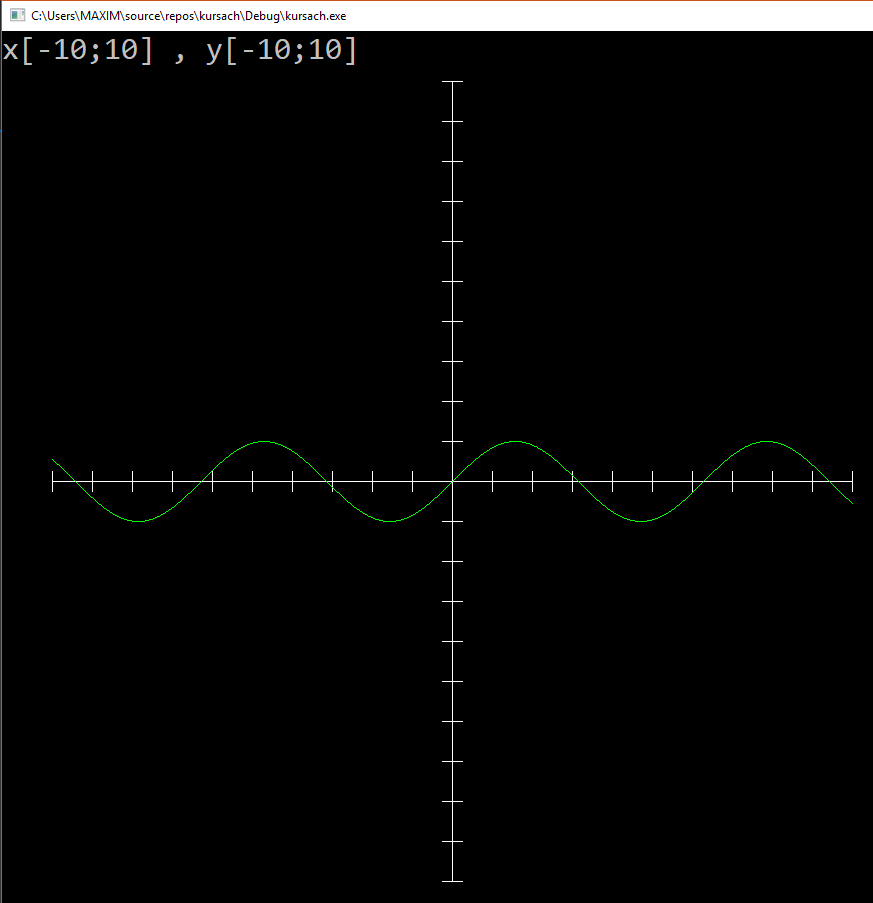


Рис. 21 Построение синуса

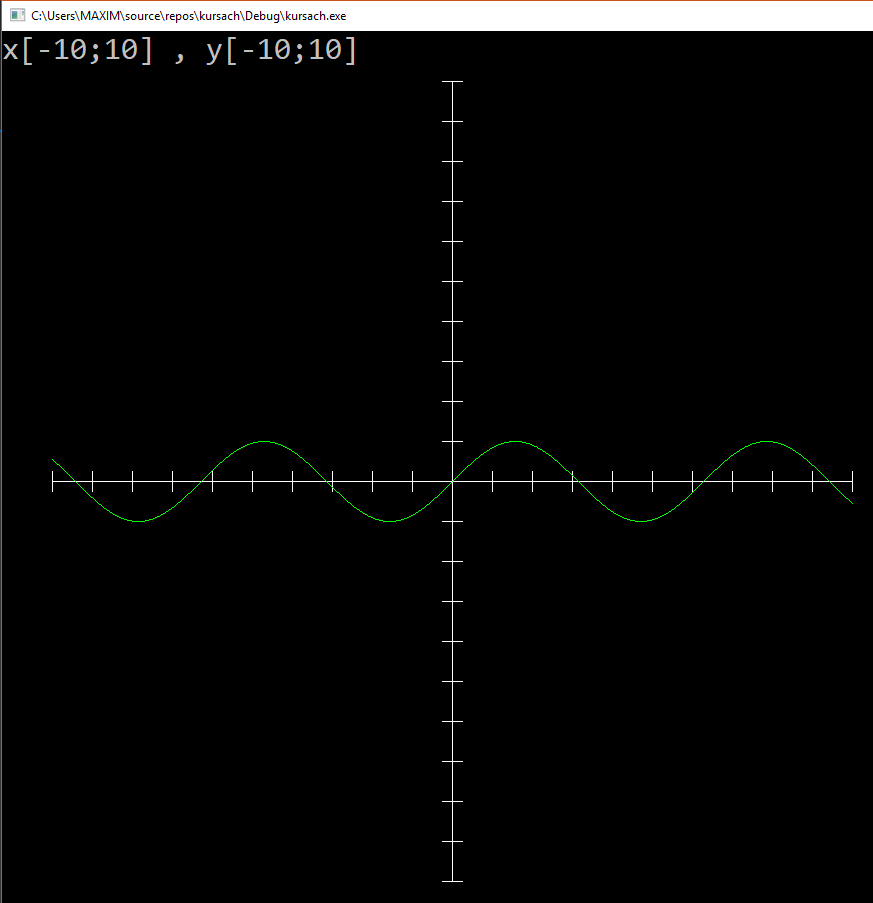


Рис. 22 Построение синуса

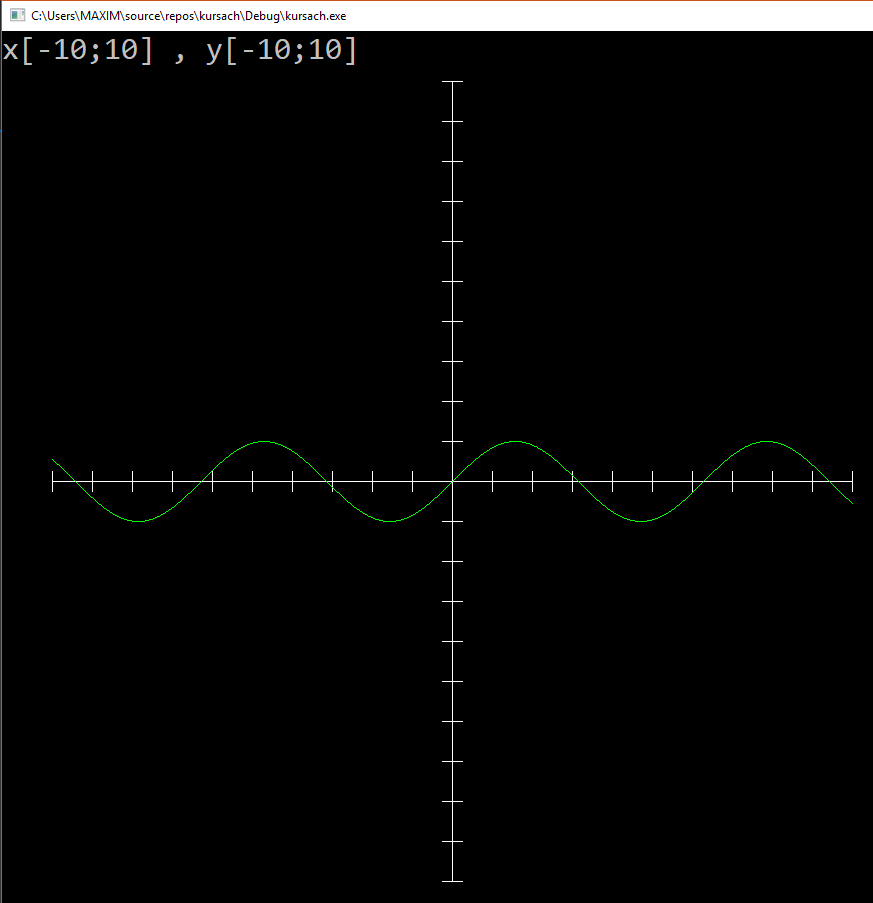


Рис. 23 Построение синуса

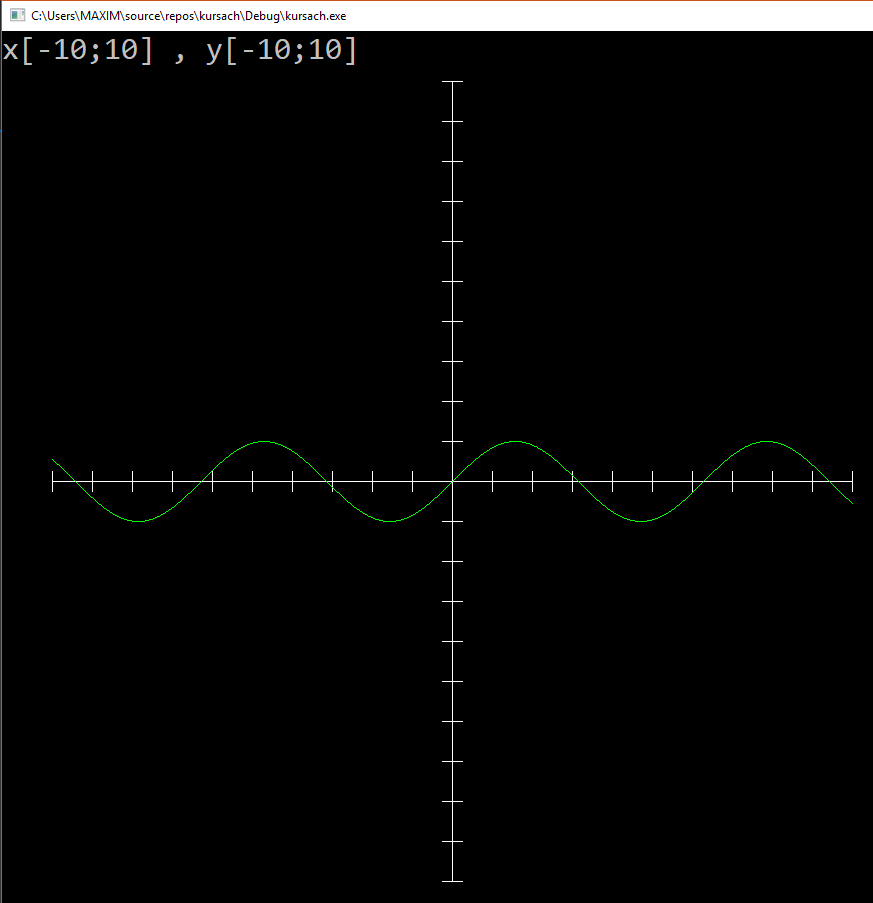


Рис. 24 Построение синуса

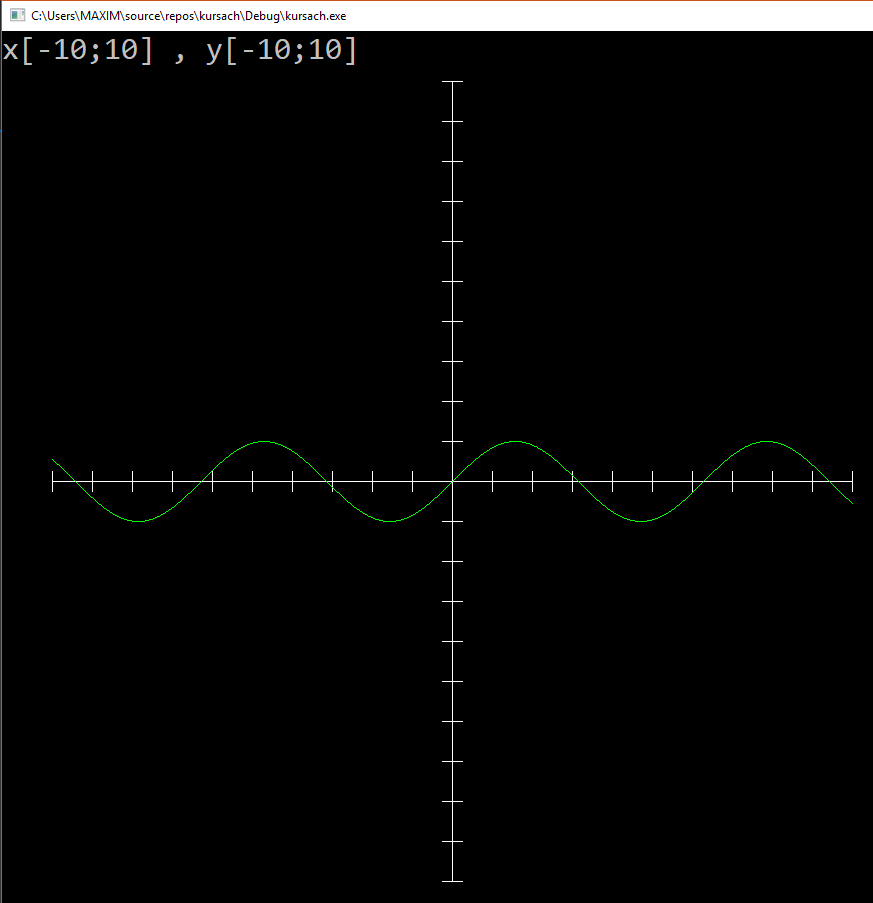


Рис. 25 Построение синуса

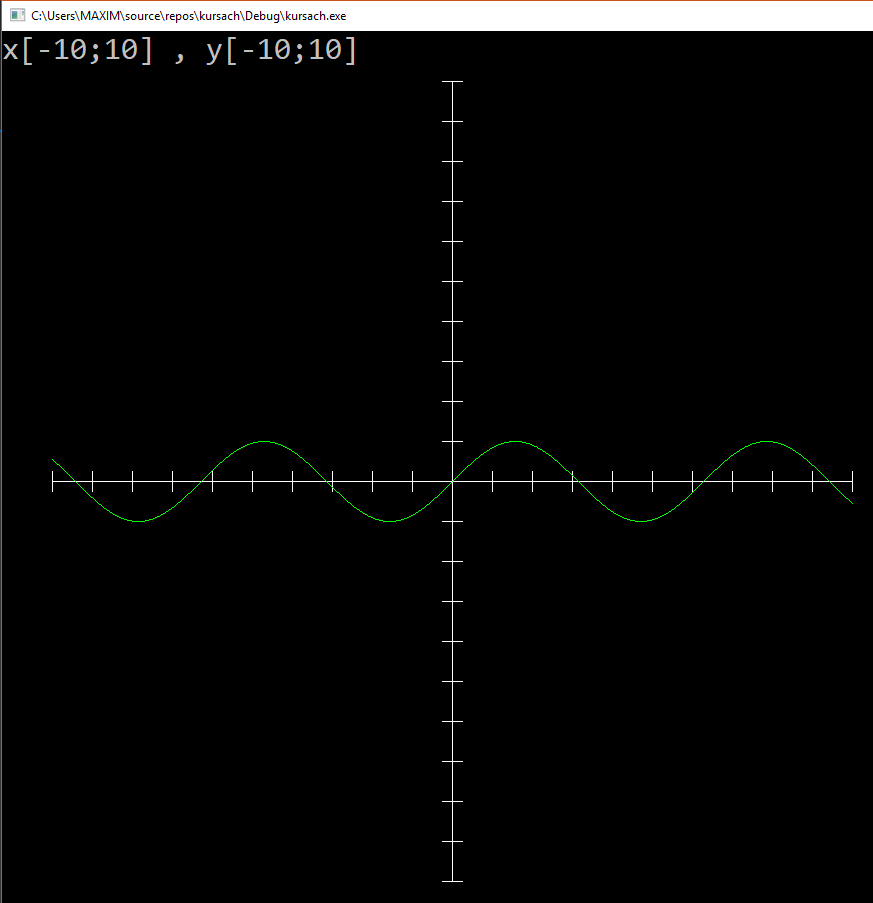


Рис. 26 Построение синуса

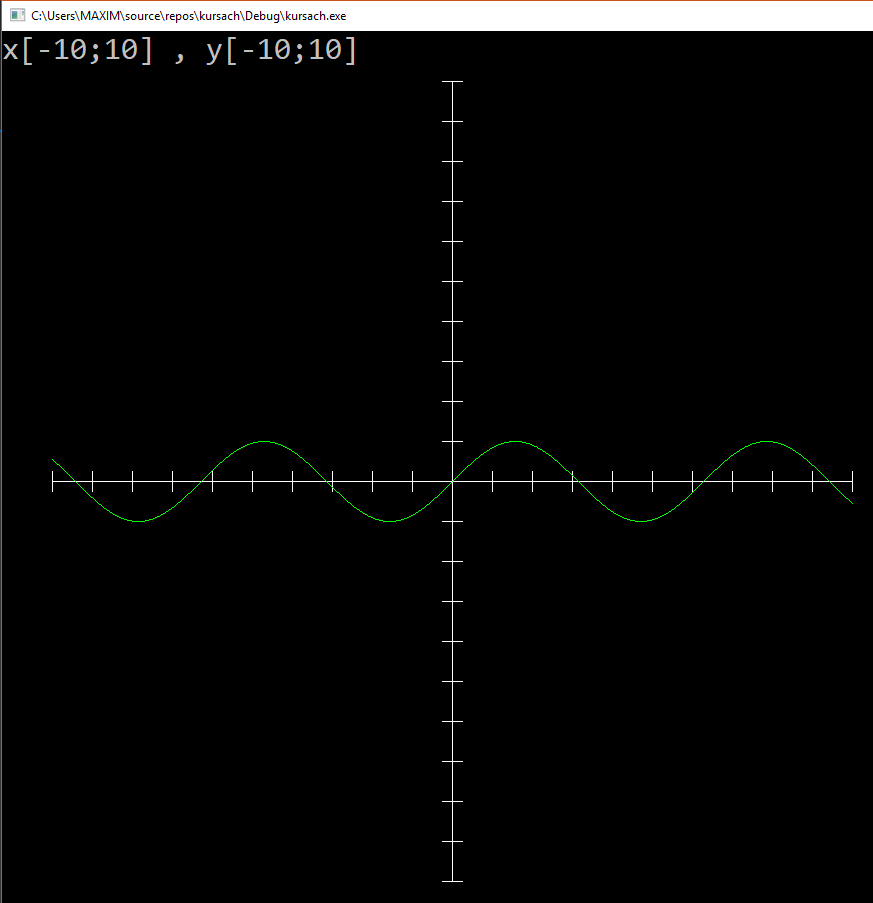


Рис. 27 Построение синуса

При нажатии на кнопку ~ программа закрывается.

# Заключение

В результате выполнения курсовой работы был разработан программный продукт, позволяющий строить график. При разработке программного продукта были построены UML диаграммы, на основе, которых реализованы классы в самой программе. Для реализации методов классов были составлены алгоритмы построения графиков по точкам. Для проверки корректности результатов, получаемых после исполнения алгоритмов, были сформированы выходные тестовые данные. Код проекта разбит на отдельные файлы на основе логики работы отдельных частей кода. Все требования, предъявленные в Техническом задании, удовлетворены.

# Список литературы

В качестве источников разработки использовались данные ресурсы:

* Зорина, Н.В. Методические указания по выполнению курсовой работы для бакалавров, обучающихся по направлениям 09.03.02«Информационные системы и технологии», 09.03.04 «Программная инженерия» / Н.В. Зорина, Л.Б. Зорин, О.В. Соболев,- Москва, 2017 - 41 с.
* Лафоре Р. Объектно-ориентированное программирование C++. Классика Computer Science. 4-е изд. – СПБ.: Питер, 2018. – 928 с.: ил. – (Серия «Классика Computer Science»).

# Приложение №1. Исходный код программы

**Header.h**

#pragma once

class Function

{

protected:

int x = 450, y = 450;

public:

void Down();

void Up();

void Left();

void Right();

virtual void Print() = 0;

virtual void Promeg() = 0;

};

class Const :public Function {

float C;

public:

Const();

Const(float C);

void Set(float C);

void Print()override;

void Promeg()override;

};

class Pryamaya :public Function {

float k, b;

public:

Pryamaya();

Pryamaya(float k, float b);

void Set(float k, float b);

void Print()override;

void Promeg()override;

};

class Sin :public Function {

public:

void Print()override;

void Promeg()override ;

};

class Cos : public Function {

public:

void Print()override ;

void Promeg()override ;

};

class Tg : public Function {

public:

void Print()override ;

void Promeg()override ;

};

class Ctg : public Function {

public:

void Print()override ;

void Promeg()override ;

};

class Asin : public Function {

public:

void Print()override ;

void Promeg()override ;

};

class Acos : public Function {

public:

void Print()override ;

void Promeg()override ;

};

class Exp : public Function {

public:

void Print()override ;

void Promeg()override ;

};

class Ax : public Function {

public:float a;

public:

Ax();

Ax(float a);

void Set(float a);

void Print()override ;

void Promeg()override ;

};

void Draw();

**Header.cpp**

#define \_USE\_MATH\_DEFINES

#include <iostream>

#include "Complex.h"

#include "windows.h"

#include <iostream>

#include "Header.h"

using namespace std;

COLORREF white = RGB(255, 255, 255);

COLORREF green = RGB(0, 255, 0);

HWND console = GetConsoleWindow();

HDC hdc = GetDC(console);

void Function::Down()

{

this->y = this->y - 40;

}

void Function::Up() {

this->y = this->y + 40;

}

void Function::Left() {

this->x = this->x + 40;

}

void Function::Right() {

this->x = this->x - 40;

}

Const::Const() {

this->C = 0;

};

Const::Const(float C) {

this->C = C;

}

void Const::Set(float C)

{

this->C = C;

}

void Const::Print()

{

for (int i = 50; i <= 850; i++)

{

SetPixel(hdc, i, -(C \* 40) + y, green);

}

}

void Const::Promeg()

{

std::cout << "x[" << -10 - (x - 450) / 40 << ";" << 10 - (x - 450) / 40 << "] , y[" << -10 + (y - 450) / 40 << ";" << 10 + (y - 450) / 40 << "]" << endl;

};

Pryamaya::Pryamaya()

{

this->k = 0;

this->b = 0;

}

Pryamaya::Pryamaya(float k, float b)

{

this->k = k;

this->b = b;

}

void Pryamaya::Set(float k, float b)

{

this->k = k;

this->b = b;

}

void Pryamaya::Print()

{

if (abs(k) <= 1)

{

for (float i = 50; i <= 850; i++)

{

int j = y - int(k\*(i - x) + b \* 40);

SetPixel(hdc, i, j, green);

}

}

else {

for (float j = 50; j <= 850; j++)

{

int i = int((y - 40 \* b - j + k \* x) / k);

SetPixel(hdc, i, j, green);

}

}

}

void Pryamaya::Promeg()

{

std::cout << "x[" << -10 - (x - 450) / 40 << ";" << 10 - (x - 450) / 40 << "] , y[" << -10 + (y - 450) / 40 << ";" << 10 + (y - 450) / 40 << "]" << endl;

};

void Sin::Print()

{

for (float i = 50; i <= 850; i++)

{

float min = 1;

int q = 0;

for (float j = 50; j <= 850; j++)

{

if (abs(-(j - y) - 40 \* sin(-(x - i) / 40)) < min) {

min = abs(-(j - y) - 40 \* sin(-(x - i) / 40));

q = j;

}

}

SetPixel(hdc, i, q, green);

}

}

void Sin::Promeg()

{

std::cout << "x[" << -10 - (x - 450) / 40 << ";" << 10 - (x - 450) / 40 << "] , y[" << -10 + (y - 450) / 40 << ";" << 10 + (y - 450) / 40 << "]" << endl;

}

void Cos::Print()

{

for (float i = 50; i <= 850; i++)

{

float min = 1;

int q = 0;

for (float j = 50; j <= 850; j++)

{

if (abs(-(j - y) - 40 \* cos(-(x - i) / 40)) < min) {

min = abs(-(j - y) - 40 \* cos(-(x - i) / 40));

q = j;

}

}

SetPixel(hdc, i, q, green);

}

}

void Cos::Promeg()

{

std::cout << "x[" << -10 - (x - 450) / 40 << ";" << 10 - (x - 450) / 40 << "] , y[" << -10 + (y - 450) / 40 << ";" << 10 + (y - 450) / 40 << "]" << endl;

}

void Tg::Print()

{

for (float z = 50; z < 954; z = z + M\_PI \* 40)

{

for (double j = 50; j <= 850; j++)

{

float min = 1000000000;

int q;

for (double i = z - M\_PI \* 40; i < z; i++)

{

if (abs(-(j - y) - 40 \* tan(-(x - i) / 40)) < min) {

min = abs(-(j - y) - 40 \* tan(-(x - i) / 40));

q = i;

}

}

SetPixel(hdc, q, j, green);

}

}

}

void Tg::Promeg()

{

std::cout << "x[" << -10 - (x - 450) / 40 << ";" << 10 - (x - 450) / 40 << "] , y[" << -10 + (y - 450) / 40 << ";" << 10 + (y - 450) / 40 << "]" << endl;

}

void Ctg::Print()

{

for (float z = 50; z < 954; z = z + M\_PI \* 40)

{

for (double j = 50; j <= 850; j++)

{

float min = 1000000000;

int q;

for (double i = z - M\_PI \* 40; i < z; i++)

{

if (abs(-(j - y) - 40 \* 1 / tan(-(x - i) / 40)) < min) {

min = abs(-(j - y) - 40 \* 1 / tan(-(x - i) / 40));

q = i;

}

}

SetPixel(hdc, q, j, green);

}

}

}

void Ctg::Promeg()

{

std::cout << "x[" << -10 - (x - 450) / 40 << ";" << 10 - (x - 450) / 40 << "] , y[" << -10 + (y - 450) / 40 << ";" << 10 + (y - 450) / 40 << "]" << endl;

}

void Asin::Print()

{

for (float j = 50; j <= 850; j++)

{

float min = 5;

int q = -1;

for (float i = 50; i <= 850; i++)

{

if (abs(-(j - y) - 40 \* asin(-(x - i) / 40)) < min) {

min = abs(-(j - y) - 40 \* asin(-(x - i) / 40));

q = i;

}

}

SetPixel(hdc, q, j, green);

}

}

void Asin::Promeg()

{

std::cout << "x[" << -10 - (x - 450) / 40 << ";" << 10 - (x - 450) / 40 << "] , y[" << -10 + (y - 450) / 40 << ";" << 10 + (y - 450) / 40 << "]" << endl;

}

void Acos::Print()

{

for (float j = 50; j <= 850; j++)

{

float min = 5;

int q = -1;

for (float i = 50; i <= 850; i++)

{

if (abs(-(j - y) - 40 \* acos(-(x - i) / 40)) < min) {

min = abs(-(j - y) - 40 \* acos(-(x - i) / 40));

q = i;

}

}

SetPixel(hdc, q, j, green);

}

}

void Acos::Promeg()

{

std::cout << "x[" << -10 - (x - 450) / 40 << ";" << 10 - (x - 450) / 40 << "] , y[" << -10 + (y - 450) / 40 << ";" << 10 + (y - 450) / 40 << "]" << endl;

}

void Exp::Print()

{

for (float j = 50; j < 850; j++)

{

float min = 10;

int q = -1;

for (float i = 50; i < 850; i++)

{

if (abs(-(j - y) - 40 \* exp(-(x - i) / 40)) < min) {

min = abs(-(j - y) - 40 \* exp(-(x - i) / 40));

q = i;

}

}

SetPixel(hdc, q, j, green);

}

for (float i = 50; i < 850; i++)

{

float min = 10;

int q = -1;

for (float j = 50; j < 850; j++)

{

if (abs(-(j - y) - 40 \* exp(-(x - i) / 40)) < min) {

min = abs(-(j - y) - 40 \* exp(-(x - i) / 40));

q = j;

}

}

SetPixel(hdc, i, q, green);

}

}

void Exp::Promeg()

{

std::cout << "x[" << -10 - (x - 450) / 40 << ";" << 10 - (x - 450) / 40 << "] , y[" << -10 + (y - 450) / 40 << ";" << 10 + (y - 450) / 40 << "]" << endl;

}

Ax::Ax()

{

this->a = 0;

}

Ax::Ax(float a) {

this->a = a;

}

void Ax::Set(float a)

{

this->a = a;

}

void Ax::Print()

{

for (float j = 50; j <= 850; j++)

{

float min = 10;

int q = -1;

for (float i = 50; i <= 850; i++)

{

if (abs(-(j - y) - 40 \* pow(a, (-(x - i) / 40))) < min) {

min = abs(-(j - y) - 40 \* pow(a, (-(x - i) / 40)));

q = i;

}

}

SetPixel(hdc, q, j, green);

}

for (float i = 50; i <= 850; i++)

{

float min = 10;

int q = -1;

for (float j = 50; j <= 850; j++)

{

if (abs(-(j - y) - 40 \* pow(a, (-(x - i) / 40))) < min) {

min = abs(-(j - y) - 40 \* pow(a, (-(x - i) / 40)));

q = j;

}

}

SetPixel(hdc, i, q, green);

}

}

void Ax::Promeg()

{

std::cout << "x[" << -10 - (x - 450) / 40 << ";" << 10 - (x - 450) / 40 << "] , y[" << -10 + (y - 450) / 40 << ";" << 10 + (y - 450) / 40 << "]" << endl;

}

void Draw()

{//графическая ось

for (int i = 50; i <= 850; i++)

{

SetPixel(hdc, i, 450, white);

}

for (int i = 50; i <= 850; i++)

{

SetPixel(hdc, 450, i, white);

}

for (int i = 50; i <= 850; i = i + 40)

{

for (int j = 440; j <= 460; j++)

{

SetPixel(hdc, i, j, white);

}

}

for (int i = 50; i <= 850; i = i + 40)

{

for (int j = 440; j <= 460; j++)

{

SetPixel(hdc, j, i, white);

}

}

}

**Kursach.cpp**

#define \_USE\_MATH\_DEFINES

#include <iostream>

#include <math.h>

#include "windows.h"

#include "conio.h"

#include "Header.h"

using namespace std;

int main() {

std::cout << "Enter:\n z(const)\n x(pryamaya)\n c(sinus)\n v(cosinus)\n b(tangens)\n n(cotangens)\n m(arcsinus)\n g(arccosinus)\n h(expanenta)\n j(a^x)" << endl;

char ch = 0;

char u = 0;

Const a1;

Pryamaya a2;

Sin a3;

Cos a4;

Tg a5;

Ctg a6;

Asin a7;

Acos a8;

Exp a9;

Ax a10;

Function \*q = nullptr;

u = \_getch();

switch (u)

{

case 'z':

std::system("cls");

std::cout << "Enter C : ";

float C;

cin >> C;

std::system("cls");

a1.Set(C);

q = &a1;

cout << "f(x)=" << C;

break;

case 'x':

std::system("cls");

float p, t;

std::cout << "Enter k : ";

cin >> p;

std::cout << "Enter b : ";

cin >> t;

std::system("cls");

a2.Set(p, t);

q = &a2;

if (t>=0)

{

cout << "f(x)=" << p << "x+" << t;

}

else

{

cout << "f(x)=" << p << "x" << t;

}

break;

case 'c':

std::system("cls");

q = &a3;

cout << "f(x)=sin(x)";

break;

case 'v':

std::system("cls");

q = &a4;

cout << "f(x)=cos(x)";

break;

case 'b':

std::system("cls");

q = &a5;

cout << "f(x)=th(x)";

break;

case 'n':

std::system("cls");

q = &a6;

cout << "f(x)=cth(x)";

break;

case 'm':

std::system("cls");

q = &a7;

cout << "f(x)=arcsin(x)";

break;

case 'g':

std::system("cls");

q = &a8;

cout << "f(x)=arcsin(x)";

break;

case 'h':

std::system("cls");

q = &a9;

cout << "f(x)=e^x";

break;

case 'j':

std::system("cls");

std::cout << "Enter a : ";

float y;

cin >> y;

std::system("cls");

a10.Set(y);

q = &a10;

cout << "f(x)="<<y<<"^x";

break;

}

cout << "\nEnter:\nq(INFO)\n`(FINISH)\nw(UP)\ns(DOWN)\na(LEFT)\nd(RIGHT)\nr(DRAW GRAPHIC)";

while (ch!='k') {

ch = \_getch();

switch (ch)

{

case 'q':

std::system("cls");

cout << "\nq(INFO)\nEnter: \n`(FINISH)\nw(UP)\ns(DOWN)\na(LEFT)\nd(RIGHT)\nr(DRAW GRAPHIC)";

break;

case 'r'://нарисовать ось с графиком

std::system("cls");

q->Promeg();

Draw();

q->Print();

break;

case 'w'://сместить вверх

q->Up();

std::system("cls");

q->Promeg();

Draw();

q->Print();

break;

case 's'://сместить вниз

q->Down();

std::system("cls");

q->Promeg();

Draw();

q->Print();

break;

case 'd'://сместить вправо

q->Right();

std::system("cls");

q->Promeg();

Draw();

q->Print();

break;

case 'a'://сместить влево

q->Left();

std::system("cls");

q->Promeg();

Draw();

q->Print();

break;

case '`':

std::system("cls");

std::cout << "FINISH" << endl;

return 0;

break;

}

};

}

# Приложение №2. UML-диаграмма

