******

**Отчёт по экзамену «Участие в интеграции программных модулей».**

Билет №5.

Выполнил студент: Герасимов Дмитрий Александрович

Группа: П1-18

Преподаватели: Наталья Юрьевна Бобкова,

Попов Вячеслав Николаевич

Королев, 2022

**Оглавление**

[IDEF схема программы 5](#_Toc101130462)

[1 Техническое задание 8](#_Toc101130463)

[1.1 Введение 8](#_Toc101130464)

[1.1.1 Наименование программы 8](#_Toc101130465)

[1.2 Краткая характеристика области 8](#_Toc101130466)

[1.3 Основания для разработки 8](#_Toc101130467)

[1.4 Назначение разработки 8](#_Toc101130468)

[1.4.1 Функциональное назначение 8](#_Toc101130469)

[1.4.2 Эксплуатационное назначение 8](#_Toc101130470)

[1.5 Требования к программе или программному изделию 9](#_Toc101130471)

[1.5.1 Требования к функциональным характеристикам 9](#_Toc101130472)

[1.5.2 Отказы из-за некорректных действий оператора 9](#_Toc101130473)

[1.6 Условия эксплуатации 10](#_Toc101130474)

[1.7 Климатические условия эксплуатации 10](#_Toc101130475)

[1.8 Требования к численности и квалификации 10](#_Toc101130476)

[1.8.1 Требования к численности и квалификации персонала 10](#_Toc101130477)

[1.9 Требования к составу и параметрам технических средств 10](#_Toc101130478)

[1.9.1 Состав технических средств: 10](#_Toc101130479)

[1.9.2 Требования к информационной и программной совместимости 10](#_Toc101130480)

[1.9.3 Требования к маркировке и упаковке 10](#_Toc101130481)

[1.9.4 Требования к транспортированию и хранению 10](#_Toc101130482)

[1.9.5 Специальные требования 10](#_Toc101130483)

[1.9.6 Требования к программной документации 11](#_Toc101130484)

[1.10 Технико-экономические показатели 11](#_Toc101130485)

[1.11 Стадии и этапы разработки 11](#_Toc101130486)

[1.12 Порядок контроля и приемки 11](#_Toc101130487)

[2 Руководство оператора 12](#_Toc101130488)

[2.1 Назначение программы 12](#_Toc101130489)

[2.1.1 Функциональное назначение программы 12](#_Toc101130490)

[2.1.2 Эксплуатационное назначение программы 12](#_Toc101130491)

[2.2 Условия выполнения программы 12](#_Toc101130492)

[2.2.1 Минимальный состав аппаратных средств 12](#_Toc101130493)

[2.2.2 Требования к пользователю 12](#_Toc101130494)

[2.3 Выполнение программы 13](#_Toc101130495)

[2.3.1 Загрузка и запуск программы 13](#_Toc101130496)

[2.4 Запуск программы 13](#_Toc101130497)

[2.4.1 Завершение работы программы 13](#_Toc101130498)

[2.5 Сообщения оператору 13](#_Toc101130499)

# IDEF схема программы

Ниже представлены IDEF-схемы программы:

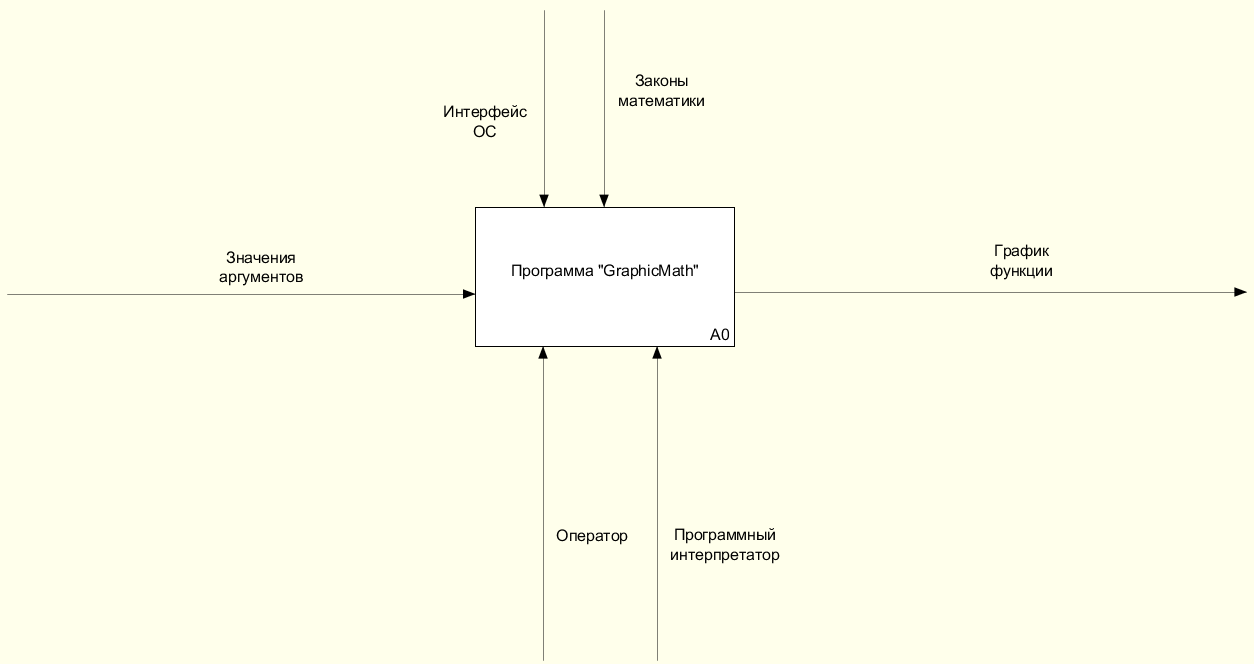


Рисунок . Первый уровень

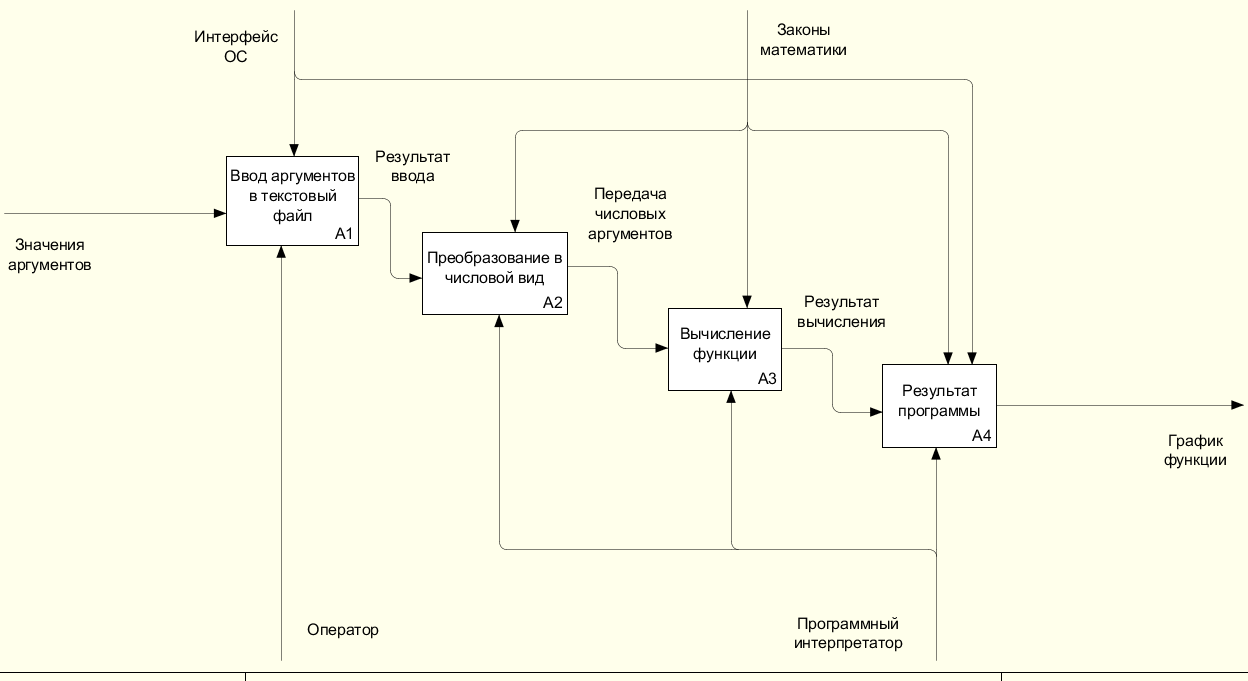


Рисунок . Второй уровень

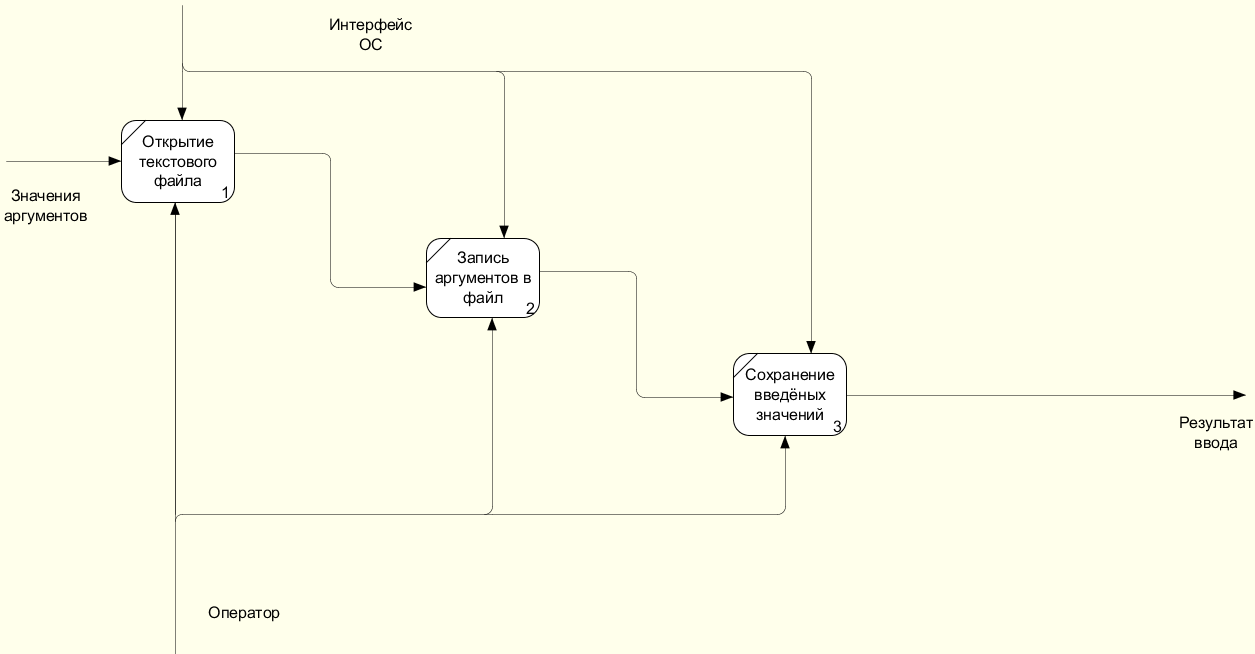


Рисунок . Третий уровень

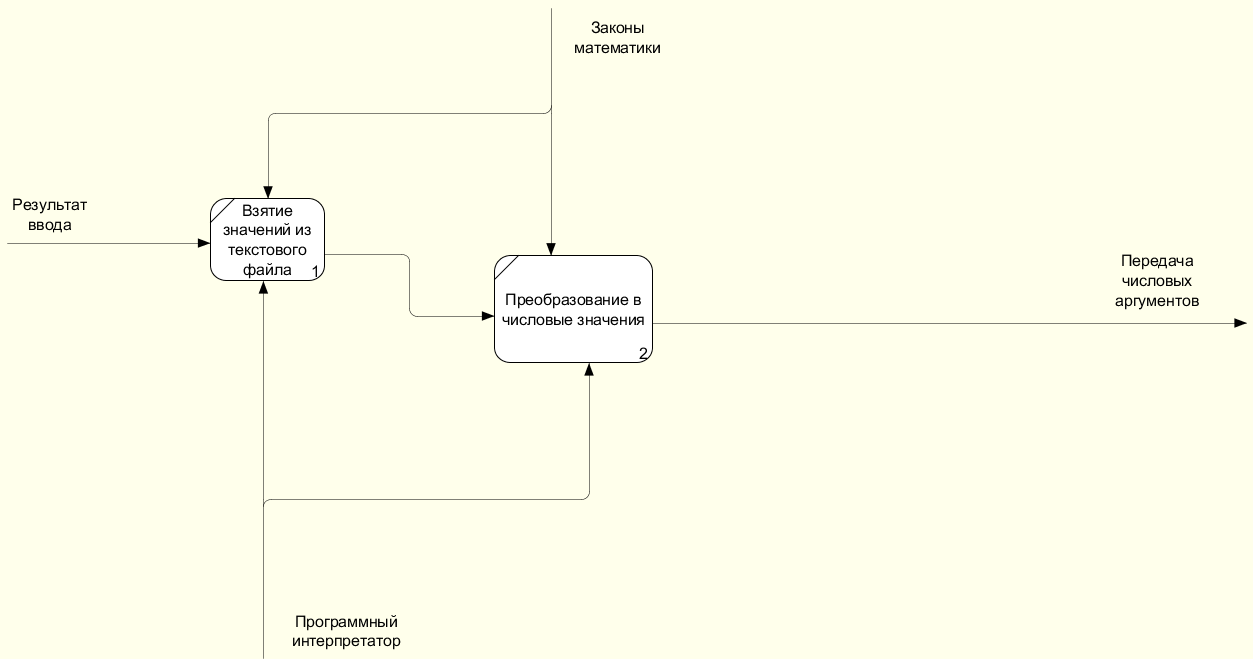


Рисунок . Четвертый уровень

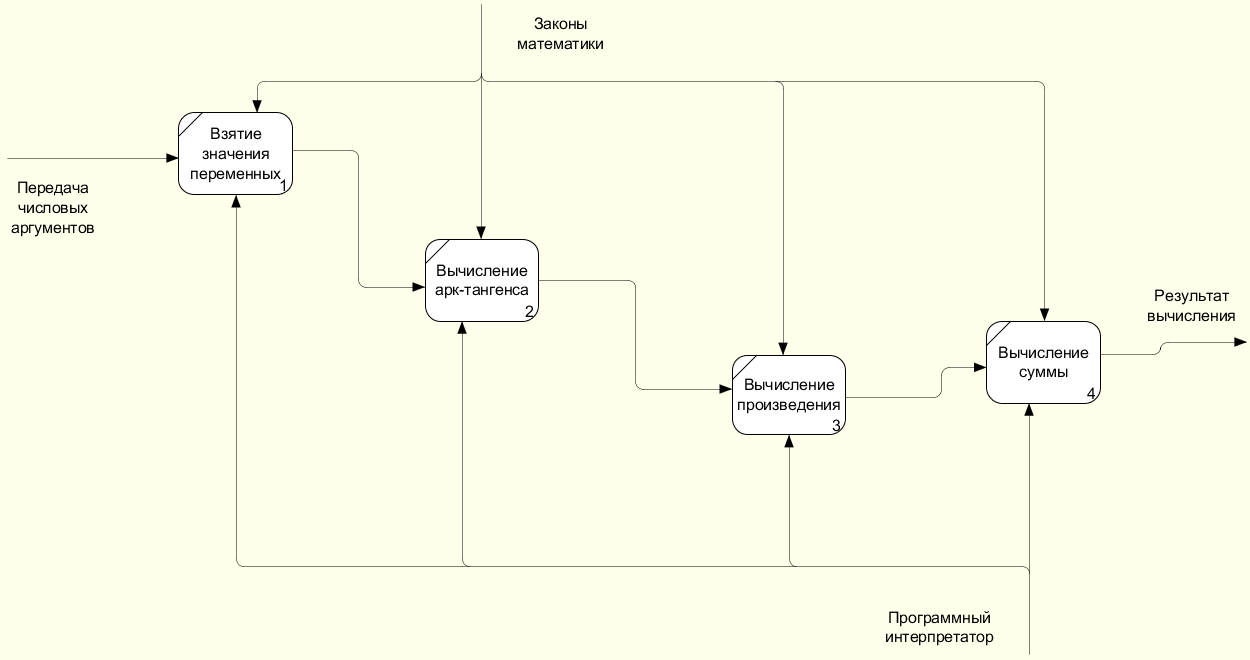


Рисунок . Пятый уровень

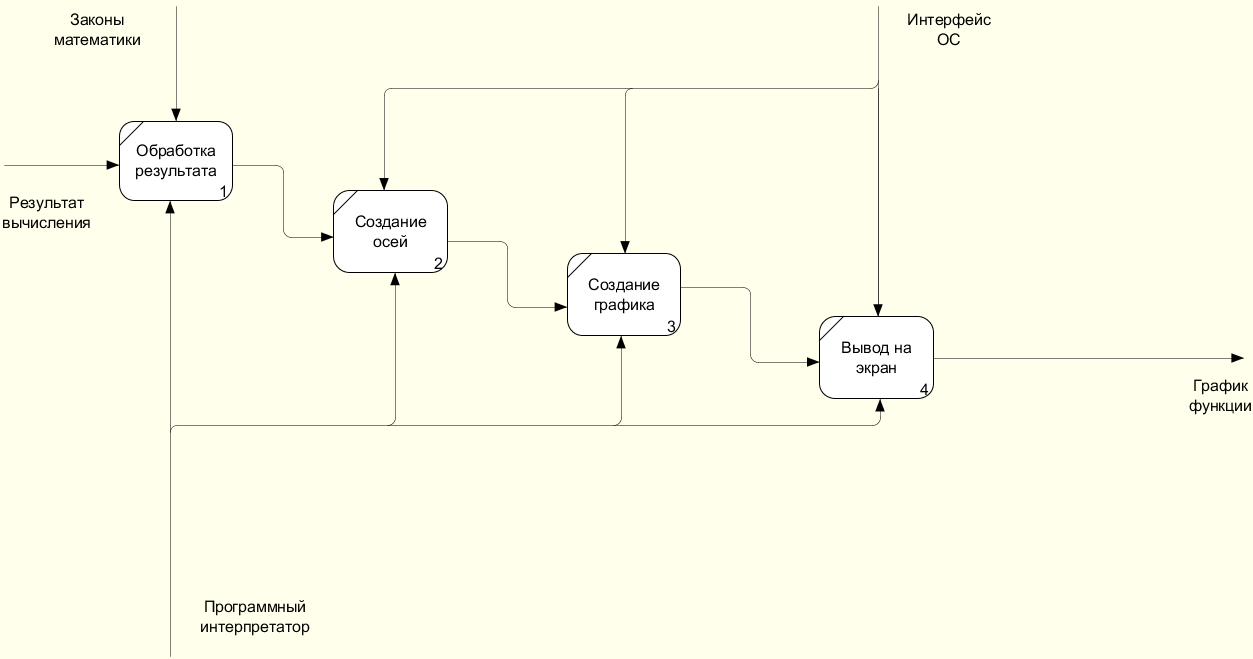


Рисунок . DFD уровень

# Техническое задание

## Введение

### Наименование программы

Наименование программы – “GraphicMath”.

## Краткая характеристика области

Программа “GraphicMath” предназначена для построения графика функции на координатной оси.

## Основания для разработки

Основанием для разработки является Договор 12 от 01.08.2020. Договор утвержден Директором Технологического Университета имени дважды Героя Советского Союза, летчика-космонавта А.А. Леонова, ККМТ. Ниже именуемым в дальнейшем Заказчиком и Жак Фреско, именуемым в дальнейшем исполнителем.

Согласно Договору, Исполнитель обязан разработать и установить программу “GraphicMath” на оборудовании Заказчика не позднее 19.04.2021, предоставить исходные коды и документацию к разработанной системе не позднее 19.04.2021.

Наименование темы разработки – “ GraphicMath”

## Назначение разработки

Программа “GraphicMath” предназначается для использования в колледже студентами и преподавателями.

### Функциональное назначение

Программа позволяет выполнить построение графика любой функции на координатной оси, по точечно, и с любым указанным шагом.

### Эксплуатационное назначение

Приложение должно эксплуатироваться только в учебных целях.

## Требования к программе или программному изделию

### Требования к функциональным характеристикам

#### Требования к составу выполняемых функций

После запуска программы пользователю отображается график. В графике показано расположение функции на координатной оси.

#### Требование к организации входных и выходных данных

Входные данные являются цифры, переменные и математические символы. Подаются значение в текстовый файл, после чего выводится на график функций.

#### Требования к временным характеристикам

После того, как преподаватель или студент написали значения в текстовый файл, выводятся данные и рисуется график функций.

#### Требования к надёжности

Вероятность безотказной работы системы должна составлять не менее 99.99% при условии исправности версии программы

#### Требование к обеспечению надёжного функционирования программы

Требования отсутствуют

#### Время восстановления после отказа

Время восстановления после отказа неопределённо даже при условии соблюдения условий эксплуатации технических и программных средств. Так как отказ может быть вызвано неисправностью технических средств, фатальным сбоем операционной системы.

### Отказы из-за некорректных действий оператора

Отказы программы возможен из-за внутренних багов или неполадок самого приложения.

## Условия эксплуатации

Программа запускается на компьютере. Окно программы должно быть открыто в любой IDE, где есть Python, должна быть возможность запустить текстовый файл вместе с программой. Запуск программы осуществляться студентом или преподавателем.

## Климатические условия эксплуатации

Специальные условия не требуются.

## Требования к численности и квалификации

Программа не требует проведения каких-либо видов обслуживания.

### Требования к численности и квалификации персонала

Специальные условия не требуются.

## Требования к составу и параметрам технических средств

### Состав технических средств:

Состав технических средств:

* процессор с тактовой частотой, не менее 1 ГГц;
* оперативная память объемом, не менее 512 Мб;
* Свободного места на диске ~10 Мб;

### Требования к информационной и программной совместимости

Должен быть соблюден правильный порядок цифр и их количество в текстовом файле.

### Требования к маркировке и упаковке

Программа передается как скрипт на языке Python.

### Требования к транспортированию и хранению

Для транспортировки программы нужно интернет-соединение.

### Специальные требования

Специальные требования отсутствуют.

### Требования к программной документации

Предварительный состав программной документации:

• техническое задание (включает описание применения);

• руководство оператора;

## Технико-экономические показатели

Программа создана с некоммерческой целью.

## Стадии и этапы разработки

Всего этапов разработки:

* Разработка требований
* Проектирование
* Реализация
* Тестирование
* Внедрение

## Порядок контроля и приемки

Приемосдаточные испытания программы должны проводиться согласно разработанной исполнителем и согласованной заказчиком «Программы и методики испытаний».

Ход проведения приемо-сдаточных испытаний заказчик и исполнитель документируют в протоколе испытаний. На основании протокола испытаний исполнитель совместно с заказчиком подписывают акт приемки-сдачи программы в эксплуатацию.

# Руководство оператора

## Назначение программы

### Функциональное назначение программы

Программа “GraphicMath” будет использоваться в колледже студентами и преподавателями. Эта программа позволяет студентам или преподавателям посмотреть график функции.

Функциональные возможности:

* Добавление, удаление и изменение текстового файла
* Просмотр графика функций

### Эксплуатационное назначение программы

Программа должна эксплуатироваться студентами и преподавателями колледжа. Программу можно начать использовать, когда вписаны определенные значения в текстовый файл.

## Условия выполнения программы

### Минимальный состав аппаратных средств

Минимальный состав используемых технических (аппаратных) средств:

* Процессор с тактовой частотой, не менее 1 ГГц;
* Оперативная память объемом, не менее 512 Мб;
* Свободного места на диске ~10 Мб;

### Требования к пользователю

Конечный пользователь программы (оператор) должен обладать практическими навыками работы с графическим пользовательским интерфейсом операционной системы Windows.

## Выполнение программы

### Загрузка и запуск программы

Для запуска программы требуется запустить скрипт, написанный на языке Python и имеющий расширение .py. После этого откроется консоль программы для ввода-вывода данных.

## Запуск программы

При запуске программы выводится окно с направляющими линиями и кнопками: «Загрузить файл» и «Закрыть».

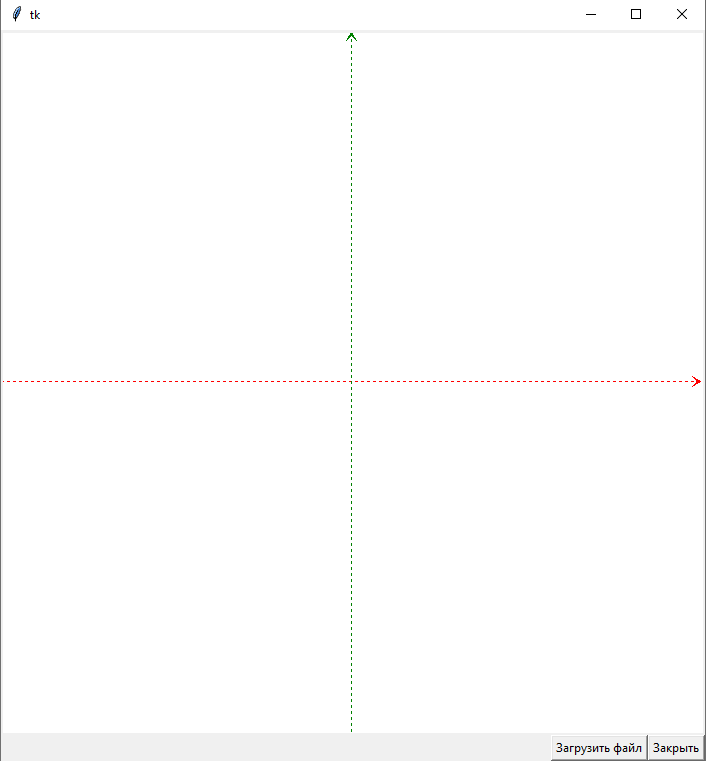


Рисунок . Стартовое окно

При нажатии на «Загрузить файл», открывается диалоговое окно, где выбирается файл с форматом «txt».

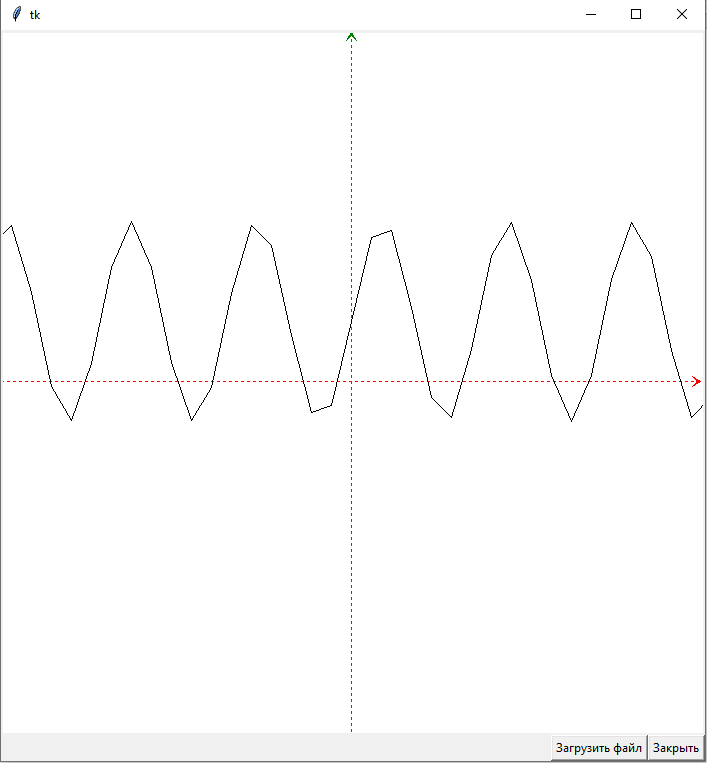


Рисунок . Загруженный файл

При нажатии кнопки «Закрыть», программа закроется.

### Завершение работы программы

Для завершения работы программы достаточно закрыть консоль программы.

## Сообщения оператору

Сообщения оператору отсутствуют.

# Отладка и тестирование готового программного средства

Функционал программы тестировался встроенными средствами PyCharm.

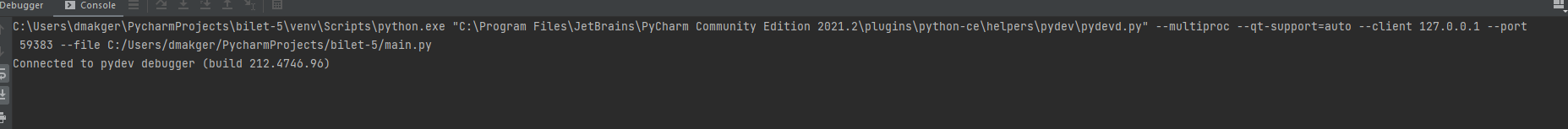


Рисунок . Отладка PyCharm

Листинг 1. Главный модуль (main.py)

import math

from app import App

def main():

# Коэффициенты

input\_str = open('input.txt').read()

a, b, c, step = list(map(float, input\_str.split()))

# Функция

func = lambda x: a \* math.sin(x) + b

# Отрисовка

app = App(func)

app.create\_canvas(step)

app.draw()

if \_\_name\_\_ == '\_\_main\_\_':

main()

Листинг 2. Модуль отрисовки графика (app.py)

from tkinter import \*

class App:

# Размеры экрана

WIDTH = 700

HEIGHT = 700

# Цвет фона

BG = 'white'

ARROWSHAPE = "5 10 5"

# Шаг

STEP = 20

def \_\_init\_\_(self, func):

self.root = Tk()

self.func = func

self.canvas = None

def create\_canvas(self, step=None):

"""Создание canvas'a. В ней отрисовывается функция"""

if step is None:

step = self.STEP

self.canvas = Canvas(self.root, width=self.WIDTH, height=self.HEIGHT, bg=self.BG)

self.canvas.pack()

self.set\_guide\_lines()

# Отрисовка точек

points = self.get\_points(self.func)

self.rendering\_points(points, step)

def set\_guide\_lines(self):

"""Создание направляющих (x, y) линий"""

# создание направляющих x y линий

self.canvas.create\_line(0, self.HEIGHT // 2, self.WIDTH, self.HEIGHT // 2, fill='red',

width=1, arrow=LAST, dash=(1, 1),

activefill='red4',

arrowshape=self.ARROWSHAPE)

self.canvas.create\_line(self.WIDTH // 2, self.HEIGHT, self.WIDTH // 2, 0, fill='green',

width=1, arrow=LAST, dash=(1, 1),

activefill='darkgreen',

arrowshape=self.ARROWSHAPE)

@staticmethod

def get\_points(func):

"""Вернет points для отрисовки"""

points = list()

for x in range(-100, 100):

try:

points.append((x, func(x)))

except Exception:

points.append((x, None))

return points

def rendering\_points(self, points, step=None):

"""Вернет отрисует points"""

if step is None:

step = self.STEP

last\_point = points[0]

for i in range(len(points)):

x1, y1 = last\_point

x2, y2 = points[i]

clr = 'black'

if y1 is None or y2 is None:

last\_point = points[i]

continue

self.canvas.create\_line(

x1 \* step + self.WIDTH // 2, -y1 \* step + self.HEIGHT // 2,

x2 \* step + self.WIDTH // 2, -y2 \* step + self.HEIGHT // 2,

fill=clr

)

last\_point = points[i]

def draw(self):

"""Запуск приложения"""

self.root.mainloop()

Листинг 3. Модуль входных данных (input.txt)

10 1 5