**Московский Государственный Университет Геодезии и Картографии**

**Факультет геоинформатики и информационной безопасности**

**Кафедра информационно-измерительных систем (ИС)**

Направление: Информационные системы и технологии

**Отчет**

**о выполненных практических работах по дисциплине ИНФОРМАТИКА**

*(индивидуальные задания №4a, №4b, №5a, №5b, №5c, №6, №7)*

Проверила: Выполнил:

доц. кафедры ИС студент ФГиИБ 2023 ИСиТ-2(б)

Лапчинская М.П. Утягулов А.В.

**МОСКВА – 2024**

**СОДЕРЖАНИЕ**

**стр.**

**ВВЕДЕНИЕ**

**1. ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ №4 (ВАРИАНТ …) 3**

**1.1. Структурное (императивное) программирование на Python 4**

**1.1.1. Программирование без пользовательских функций (№4a)**

1.1.1.1. Постановка задачи №4а

1.1.1.2. Используемые инструкции в программе

1.1.1.3. Блок-схема решения задачи

1.1.1.4. Результаты работы программы (скриншоты)

1.1.1.5. Текст отлаженной программы

**1.1.2. Программирование с пользовательскими функциями (№4b)**

1.1.2.1. Постановка задачи №4b

1.1.2.2. Используемые инструкции в программе

1.1.2.3. Блок-схема решения задачи

1.1.2.4. Результаты работы программы (скриншоты)

1.1.2.5. Текст отлаженной программы

**2. ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ № 5 (ВАРИАНТ …)**

**2.1. Структурное (императивное) программирование на Python**

**2.1.1. Программирование без пользовательских функций (№5a)**

2.1.1.1. Постановка задачи №5а

2.1.1.2. Используемые инструкции в программе

2.1.1.3 Блок-схема решения задачи

2.1.1.4. Результаты работы программы (скриншоты)

2.1.1.5. Текст отлаженной программы

**2.1.2. Программирование с пользовательскими функциями (№5b)**

2.1.2.1. Постановка задачи №5b

2.1.2.2. Используемые инструкции в программе

2.1.2.3. Блок-схема решения задачи

2.1.2.4. Результаты работы программы (скриншоты)

2.1.2.5. Текст отлаженной программы

**2.2. Объектно-ориентированное программирование (№5c)**

2.2.1. Постановка задачи №5c

2.2.2. Используемые инструкции в программе

2.2.3. Результаты работы программы (скриншоты)

2.2.4. Текст отлаженной программы

**3. ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ № 6 (ВАРИАНТ …)**

**3.1. Объектно-ориентированное программирование на Python**

3.1.1. Постановка задачи №6 в программе перевода систем счисления

3.1.2. Используемый(е) класс(ы) и метод(ы) в программе

3.1.3. Используемые стандартные и пользовательские функции в программе

3.1.4. Результаты работы программы (скриншоты)

3.1.5. Текст отлаженной программы

**4. ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ № 7 (ВАРИАНТ …)**

**4.1. Объектно-ориентированное программирование на Python**

4.1.1. Постановка задачи №7 в программе

4.1.2. Разработка пользовательского интерфейса программы с использованием   
виджетов PyQt

4.1.3. Используемые библиотеки в программе

4.1.4. Результаты работы программы (скриншоты из Qt дизайнера форм)

4.1.5. Текст отлаженной программы

**ВЫВОДЫ**

**СПИСОК ИСПОЛЬЗОВАННЫХ ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ**

**1. ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ №4 (ВАРИАНТ 26)**

Дан двумерный массив **A** размером **n\*m** элементов, заполненный целыми случайными числами из диапазона **(r1, r2)**. Вывести на экран исходный массив и результаты вычислений/преобразований с точностью до **3-го** знака.

**1.1. Структурное (императивное) программирование (№4a)**

**1.1.1. Программирование без пользовательских функций**

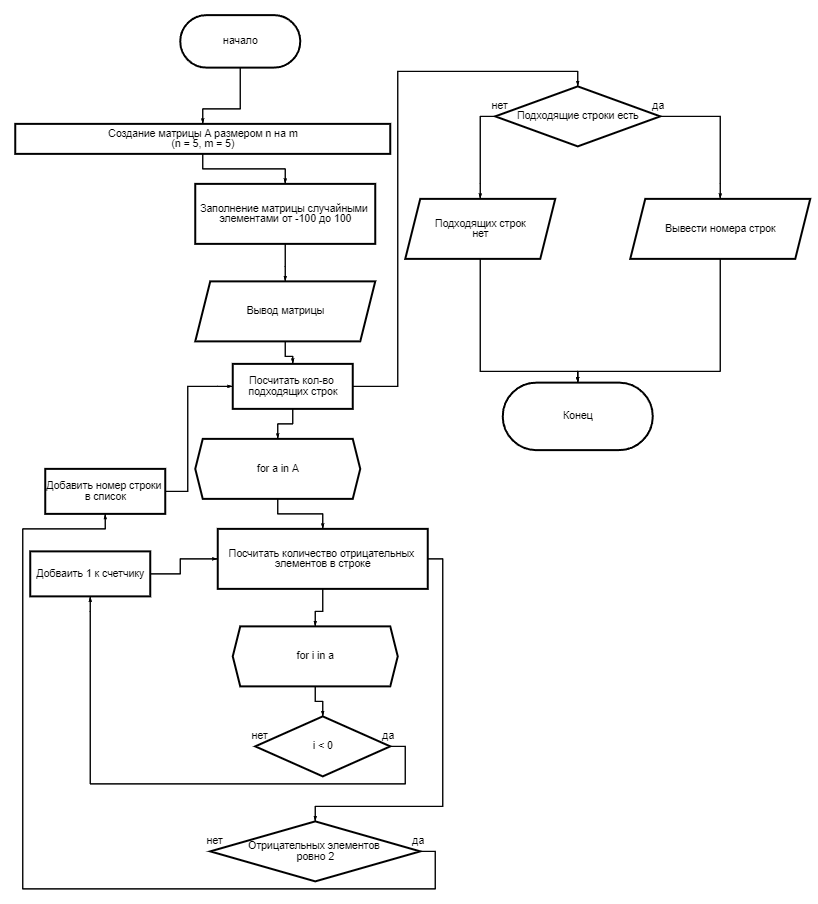
1.1.1.1. Постановка задачи №4а

Дан двумерный массив A размером 5\*5 элементов, заполненный случайными целыми числами из диапазона (-100,100). Определить, есть ли в данном массиве строка, в которой ровно два отрицательных элемента. Если имеются, то вывести их номера, если не имеются – сообщение «Строк с отрицательными элементами в массиве нет».

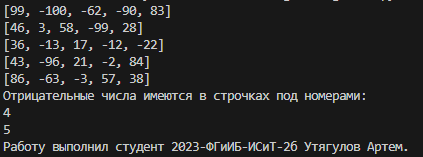
**1.1.1.2. Используемые инструкции в программе**

|  |  |  |
| --- | --- | --- |
| **№** | **Инструкция  (из кода программы)** | **Название и назначение инструкции** |
| **1.** | **from random import randint** | **Импортирование функции randint из библиотеки random** |
| **2.** | **n, m = 5** | **Присваивание переменным m и n значения 5** |
| **3.** | **A = []** | **Создание пустого списка A** |
| **4.** | **for i in range(m):**  **a = []**  **for j in range(n):**  **k = randint(-100, 100)**  **a.append(k)**  **A.append(a)** | **Заполнение списка А внутренними списками, присвоение каждому элементу случайного значения в диапазоне от -100 до 100. Таким образом получается матрица n\*m, реализованная как список длинной m списков длинной n.** |
| **5.** | **for i in A:**  **print(i)** | **Вывод всех элементов матрицы методом перебора циклом for.** |
| **6.** | **lines = []**  **line\_count = 1** | **Создание списка номеров подходящих строчек и переменной для хранения номера текущей строки(начиная с 1).** |
| **7.** | **for a in A:**  **count = 0**  **for i in a:**  **if i < 0:**  **count += 1**  **if count == 2:**  **lines.append(line\_count)**  **line\_count += 1** | **Перебор строчек матрицы циклом for и перебор элементов каждой строчки тем же циклом, проверка, соответствует ли элемент условию, подсчет количества подходящих элементов и если их количество в строке равно двум – добавление номера строки в список подходящих строк.** |
| **8.** | **if len(lines) > 0:**  **print("Отрицательные числа имеются в строчках под номерами: ")**  **for i in lines:**  **print(i)**  **else:**  **print("Строк с отрицательными элементами не найдено")** | **Проверка, наличия подходящих строчек операцией условия if. Если подходящие строчки есть – вывод номеров этих строчек методом print(), иначе вывод тем же методом сообщения об отсутствии таких строк.** |

**1.1.1.3. Блок-схема решения задачи**



**1.1.1.4. Результаты работы программы (скриншоты)**



**1.1.1.5. Текст отлаженной программы**

from random import randint

n = 5

m = 5

A = []

for i in range(m):

a = []

for j in range(n):

k = randint(-100, 100)

a.append(k)

A.append(a)

for i in A:

print(i)

lines = []

line\_count = 1

for a in A:

count = 0

for i in a:

if i < 0:

count += 1

if count == 2:

lines.append(line\_count)

line\_count += 1

if len(lines) > 0:

print("Отрицательные числа имеются в строчках под номерами: ")

for i in lines:

print(i)

else:

print("Строк с отрицательными элементами не найдено")

print("Работу выполнил студент 2023-ФГиИБ-ИСиТ-2б Утягулов Артем.")

**1.1.2. Программирование с пользовательскими функциями (№4b)**

Дан двумерный массив **A** размером **n\*m** элементов, заполненный целыми случайными числами из диапазона **(r1, r2)**. Вывести на экран исходный массив и результаты вычислений/преобразований с точностью до **3-го** знака.

Результат представить в виде пользовательской функции.

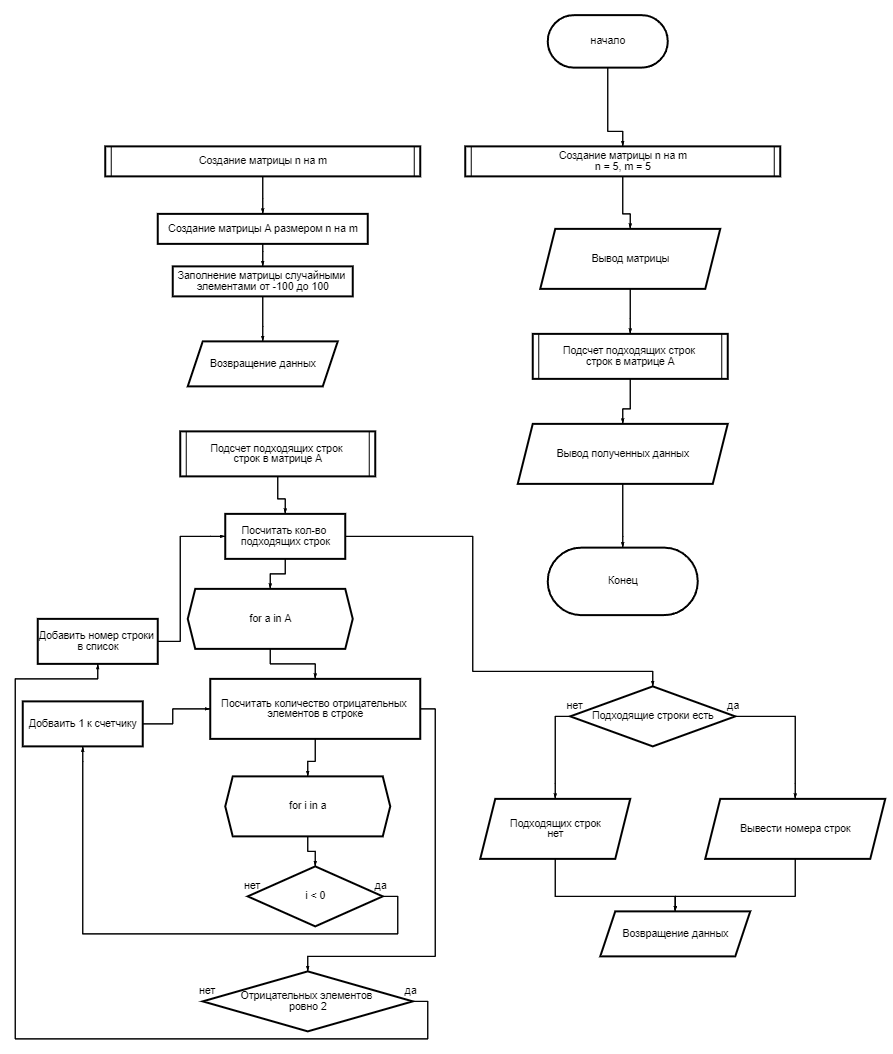
**1.1.2.1. Постановка задачи №4b**

Дан двумерный массив A размером 5\*5 элементов, заполненный случайными целыми числами из диапазона (-100,100). Определить, есть ли в данном массиве строка, в которой ровно два отрицательных элемента. Если имеются, то вывести их номера, если не имеются – сообщение «Строк с отрицательными элементами в массиве нет»

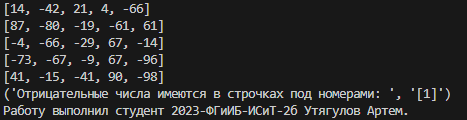
**1.1.2.2. Используемые инструкции в программе**

|  |  |  |
| --- | --- | --- |
| **№** | **Инструкция  (из кода программы)** | **Название и назначение инструкции** |
| **1.** | **def generation(n, m):** | **Создание функции генерации матрицы. Функция принимает переменные n и m являющиеся количеством столбцов и строк генерируемой матрицы.** |
| **2.** | **A = generation(5, 5)** | **Создание матрицы A размером 5 на 5 с помощью созданной ранее функции** |
| **3.** | **def count\_lines(A):** | **Функция принимает матрицу и считает кол-во подходящих под условие строк. В случае если таковых нет – возвращает сообщение об их отсутствии, иначе – возвращает список номеров подходящих строк.** |
| **4.** | **print(count\_lines(A))** | **Вызывает функцию count\_lines() передавая ей параметр – матрицу А и тут же выводит данные, возвращенные функцией.** |

**1.1.2.3. Блок-схема решения задачи**

**

**1.1.2.4. Результаты работы программы (скриншоты)**

****

**1.1.2.5. Текст отлаженной программы**

from random import randint

def generation(n, m):

A = []

for i in range(m):

a = []

for j in range(n):

k = randint(-100, 100)

a.append(k)

A.append(a)

return A

A = generation(5, 5)

for i in A:

print(i)

def count\_lines(A):

lines = []

line\_count = 1

for a in A:

count = 0

for i in a:

if i < 0:

count += 1

if count == 2:

lines.append(line\_count)

line\_count += 1

if len(lines) > 0:

return "Отрицательные числа имеются в строчках под номерами: ", str([i for i in lines])

else:

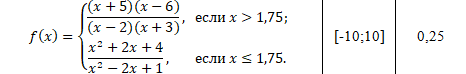
return "Строк с отрицательными элементами не найдено"

print(count\_lines(A))

print("Работу выполнил студент 2023-ФГиИБ-ИСиТ-2б Утягулов Артем.")

**2. ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ № 5 (ВАРИАНТ 26)**

**Постановка задачи**: разработать программу табулирования заданной функции с учетом области ее определения при изменении аргумента ***х***от начального значения до конечного значения постоянным шагом . Значения функции выводить с точностью **3** знака после запятой.



**2.1. Структурное (императивное) программирование**

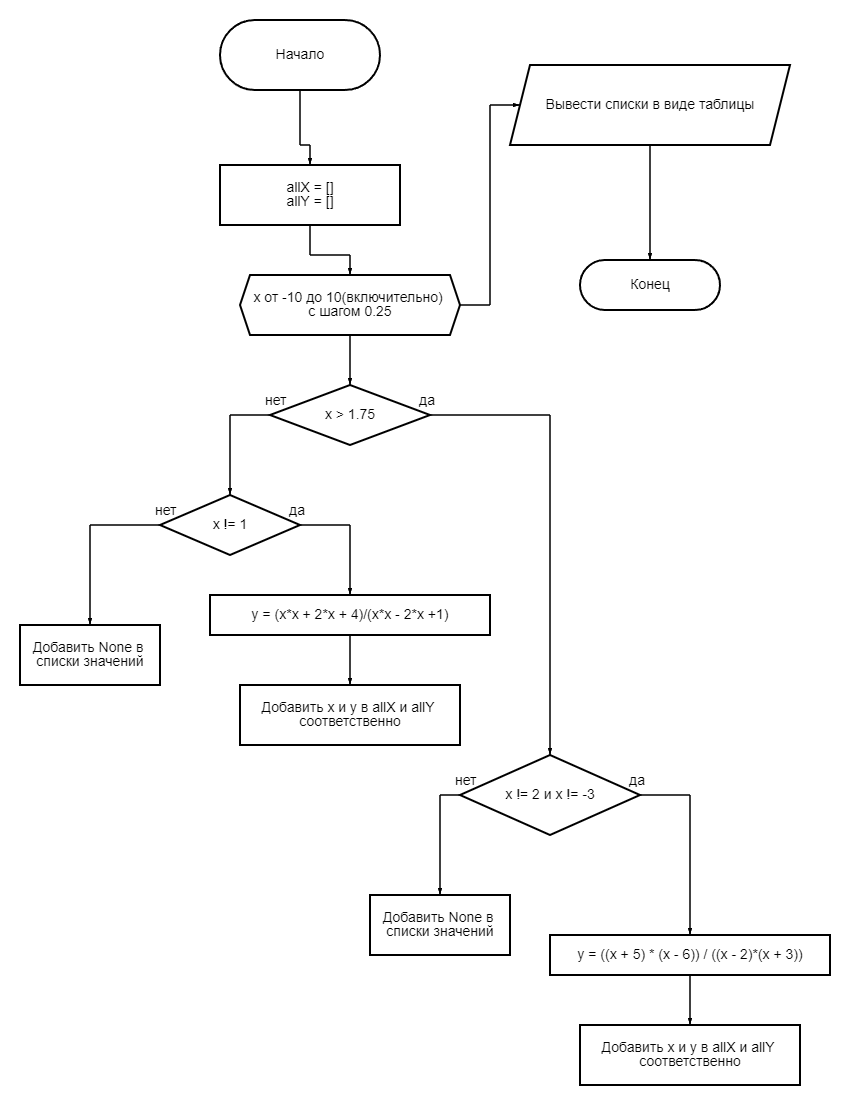
**2.1.1. Программирование без пользовательских функций (№5a)**

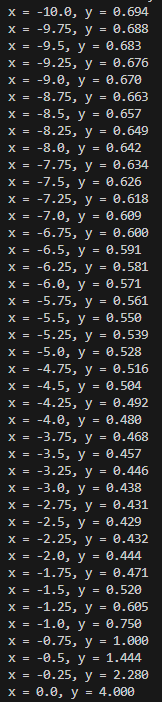
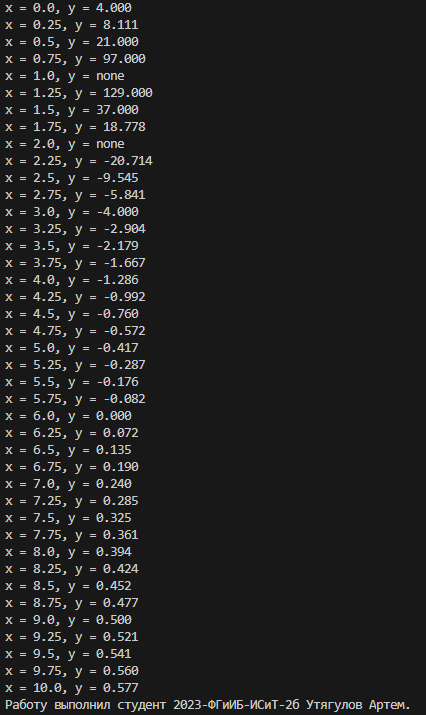
**Указание**: НЕ использовать функцию def

**2.1.1.1. Используемые инструкции в программе**

|  |  |  |
| --- | --- | --- |
| **№** | **Инструкция  (из кода программы)** | **Название и назначение инструкции** |
| **1.** | **all\_x = []**  **all\_y = []** | **Создание двух списков для хранения всех значений Х и соответствующих им значений Y.** |
| **2.** | **for x in range(-40, 41):**  **x = x / 4** | **Цикл for не может перебирать числа с шагом не являющимся целым числом, поэтому перебор идет от -40 до 41, а после число делится на 4. Таким образом мы перебираем числа от -10 до 10 включительно с шагом 0,25.** |
| **3.** | **if x > 1.75:** | **С помощью оператора if проверяется соответствие условию указанному в функции.** |
| **4.** | **if x != 2 and x != -3:** | **С помощью оператора if проверяется, не является ли число недопустимым для функции.** |
| **5.** | **y = ((x + 5) \* (x - 6)) / ((x - 2)\*(x + 3))** | **Вычисляется значение функции в точке Х.** |
| **6.** | **all\_x.append(x) all\_y.append("{j:.3f}".format(j = y))** | **В список добавляется значение Х и соответствующее ему значение Y. Обратите внимание, что Y записывается с точностью до трёх знаков после запятой с помощью метода format().** |
| **7.** | **for i in range(len(all\_x)):**  **print(f"x = {all\_x[i]}, y = {all\_y[i]}")** | **Происходит перебор всех значений Х и соответствующих им значений Y и вывод их в виде «х = {число}, y = {число}» с помощью функции print() и format()** |

**2.1.1.2. Блок-схема решения задачи**

****2.1.1.3. Результаты работы программы (скриншоты)**

**2.1.1.4. Текст отлаженной программы**

all\_x = []

all\_y = []

for x in range(-40, 41):

x = x / 4

if x > 1.75:

if x != 2 and x != -3:

y = ((x + 5) \* (x - 6)) / ((x - 2)\*(x + 3))

all\_x.append(x)

all\_y.append("{j:.3f}".format(j = y))

else:

all\_x.append(x)

all\_y.append("none")

if x <= 1.75:

if x != 1:

y = (x\*x + 2\*x + 4)/(x\*x - 2\*x +1)

all\_x.append(x)

all\_y.append("{j:.3f}".format(j = y))

else:

all\_x.append(x)

all\_y.append("none")

for i in range(len(all\_x)):

print(f"x = {all\_x[i]}, y = {all\_y[i]}")

print("Работу выполнил студент 2023-ФГиИБ-ИСиТ-2б Утягулов Артем.")

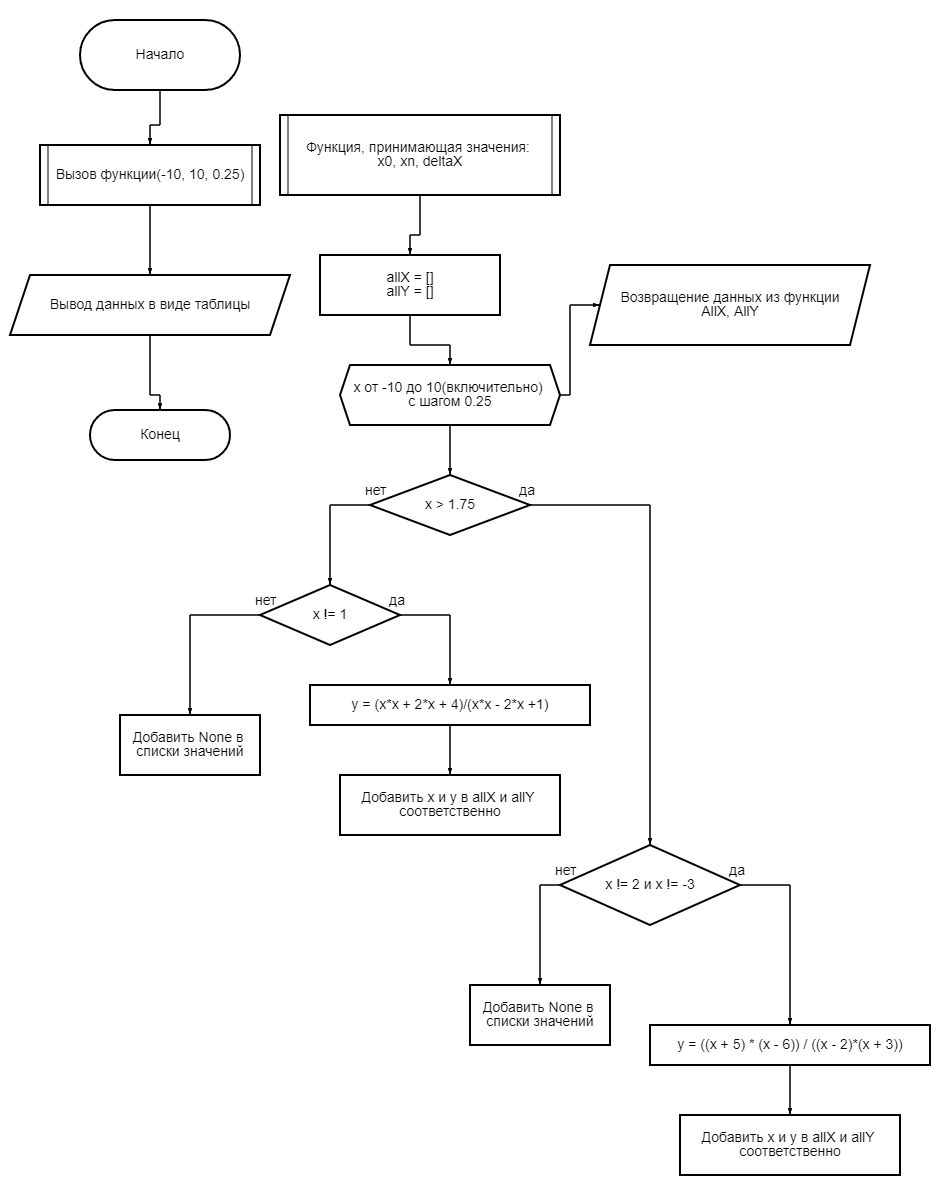
**2.1.2. Программирование с пользовательскими функциями (№5b)**

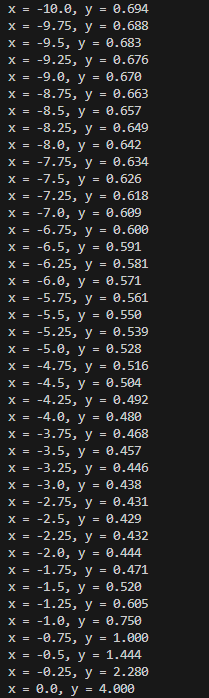
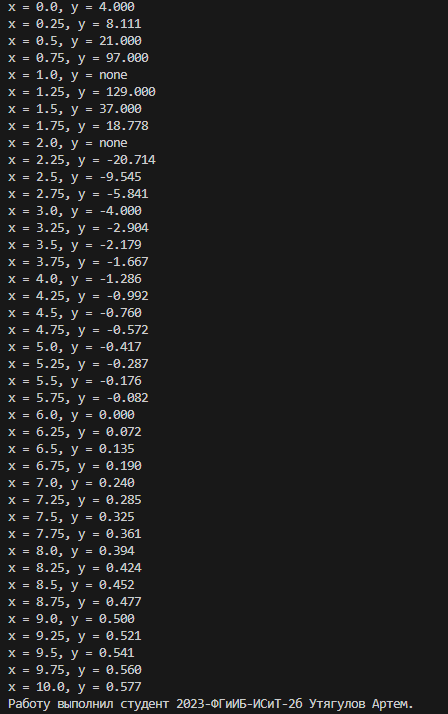
**б)** разработать аналогичную программу **№5b** табулирования, но с использованием *пользовательских функций* (точно также с учетом области ее определения при изменении аргумента значения *х* от начального до конечного значения постоянным шагом , где значения функции выводить с точностью 3 знака после запятой).

**2.1.2.1. Используемые инструкции в программе**

|  |  |  |
| --- | --- | --- |
| **№** | **Инструкция  (из кода программы)** | **Название и назначение инструкции** |
| **1.** | **x0 = -10**  **xn = 10**  **deltaX = 0.25** | **Инструкция присваивания. Создание переменных x0, xn и deltaХ.** |
| **2.** | **def function(x0, xn, deltaX):** | **Инструкция создания функции.** |
| **3.** | **function(x0, xn, deltaX)** | **Инструкция вызова функции с указанием параметров.** |
| **4.** | **all\_x, all\_y = function(x0, xn, deltaX)** | **Инструкция присваивания. Вызов функции и присваивание возвращаемых ею значений переменным all\_x и all\_y** |

**2.1.2.2. Блок-схема решения задачи**

****2.1.2.3. Результаты работы программы (скриншоты)**

**2.1.2.4. Текст отлаженной программы**

x0 = -10

xn = 10

deltaX = 0.25

def function(x0, xn, deltaX):

if deltaX < 1:

x0 = int(x0 / deltaX)

xn = int(xn / deltaX)

all\_x = []

all\_y = []

for x in range(x0, xn+1):

if deltaX < 1:

x \*= deltaX

if x > 1.75:

if x != 2 and x != -3:

y = ((x + 5) \* (x - 6)) / ((x - 2)\*(x + 3))

all\_x.append(x)

all\_y.append("{j:.3f}".format(j = y))

else:

all\_x.append(x)

all\_y.append("none")

if x <= 1.75:

if x != 1:

y = (x\*x + 2\*x + 4)/(x\*x - 2\*x +1)

all\_x.append(x)

all\_y.append("{j:.3f}".format(j = y))

else:

all\_x.append(x)

all\_y.append("none")

return all\_x, all\_y

all\_x, all\_y = function(x0, xn, deltaX)

for i in range(len(all\_x)):

print(f"x = {all\_x[i]}, y = {all\_y[i]}")

print("Работу выполнил студент 2023-ФГиИБ-ИСиТ-2б Утягулов Артем.")

**2.2. Объектно-ориентированное программирование(№5с)**

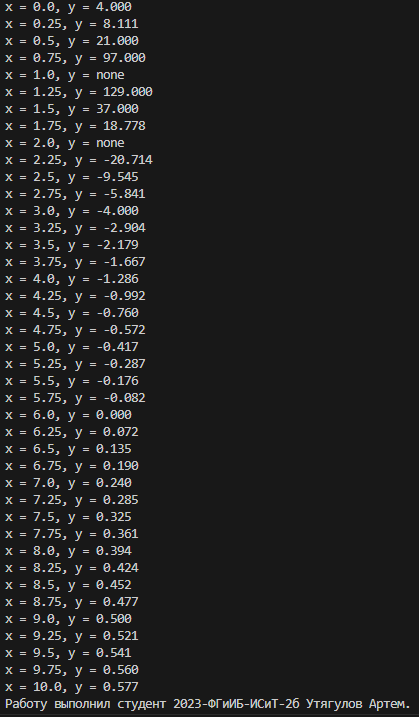
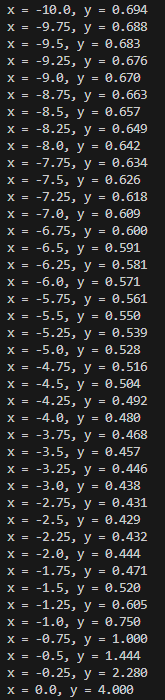
**2.2.1. Постановка задачи №5c**

**c)** разработать аналогичную программу **№5c** табулирования, но с использованием *объектно-ориентированного подхода* (и точно также с учетом области ее определения при изменении аргумента значения *х* от начального до конечного значения постоянным шагом , где значения функции выводить с точностью 3 знака после запятой)

**2.2.2. Используемые инструкции в программе**

|  |  |  |
| --- | --- | --- |
| **№** | **Инструкция  (из кода программы)** | **Название и назначение инструкции** |
| **1.** | **class FifthC** | **Создание класса** |
| **2.** | **def \_\_init\_\_(self, x0, xn, deltaX):**  **self.x0 = x0**  **self.xn = xn**  **self.deltaX = deltaX** | **Метод инициализации класса (конструктор), задающий свойства класса.** |
| **3.** | **def program(self):** | **Метод класса, выполняющий табуляцию функции с использованием свойств класса, указанных ранее в конструкторе.** |
| **4.** | **self.имя\_переменной** | **Так происходит обращение к свойствам классов.** |
| **5.** | **def print\_all(self):** | **Метод класса, выводящий все протабулированные ранее значения функции в виде таблицы.** |

**2.2.3. Результаты работы программы (скриншоты)**

****

**2.2.4. Текст отлаженной программы**

class FifthC:

def \_\_init\_\_(self, x0, xn, deltaX):

self.x0 = x0

self.xn = xn

self.deltaX = deltaX

def program(self):

if self.deltaX < 1:

self.x0 = int(self.x0 / self.deltaX)

self.xn = int(self.xn / self.deltaX)

self.all\_x = []

self.all\_y = []

for x in range(self.x0, self.xn+1):

if self.deltaX < 1:

x \*= self.deltaX

if x > 1.75:

if x != 2 and x != -3:

y = ((x + 5) \* (x - 6)) / ((x - 2)\*(x + 3))

self.all\_x.append(x)

self.all\_y.append("{j:.3f}".format(j = y))

else:

self.all\_x.append(x)

self.all\_y.append("none")

if x <= 1.75:

if x != 1:

y = (x\*x + 2\*x + 4)/(x\*x - 2\*x +1)

self.all\_x.append(x)

self.all\_y.append("{j:.3f}".format(j = y))

else:

self.all\_x.append(x)

self.all\_y.append("none")

def print\_all(self):

for i in range(len(self.all\_x)):

print(f"x = {self.all\_x[i]}, y = {self.all\_y[i]}")

x0 = -10

xn = 10

deltaX = 0.25

ekz = FifthC(x0, xn, deltaX)

ekz.program()

ekz.print\_all()

print("Работу выполнил студент 2023-ФГиИБ-ИСиТ-2б Утягулов Артем.")

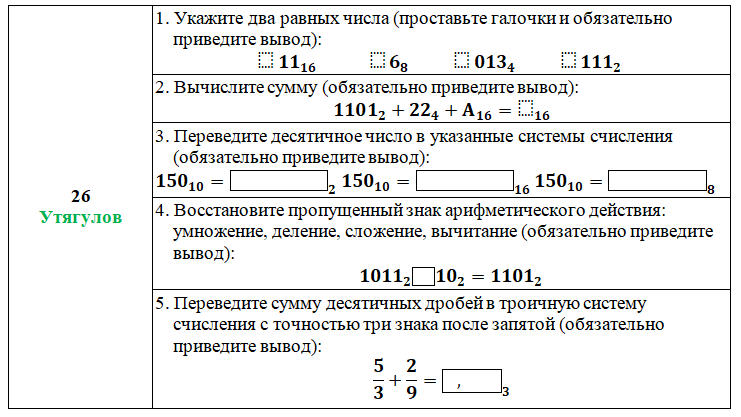
**3. ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ № 6 (ВАРИАНТ 26)**

**3.1.1. Постановка задачи №6 в программе перевода систем счисления**

Постановка задачи: написать программу на Python с использованием пользовательских функций согласно конкретным формулировкам, указанным перед каждым из 5 индивидуальных заданий Варианта.

Указание: 1) можно использовать функции и/или ООП

2) при отсутствии желания составлять программу, можно осуществить традиционный (в тетради) способ перевода (с подробным выводом!) из одной СС в другую СС.



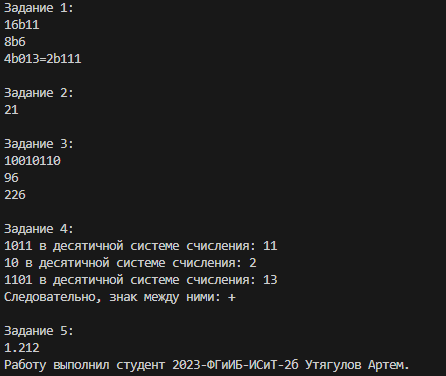
**3.1.2. Используемый(е) класс(ы) и метод(ы) в программе**

|  |  |  |
| --- | --- | --- |
| **№** | **Класс или метод** | **Описание класса или метода** |
| **1.** | **number\_operations** | **Основной класс, содержащий все функции требующиеся для выполнения задания в качестве методов. Представляет из себя инструментарий для выполнения задания.** |
| **2.** | **convert\_to\_n** | **Метод, конвертирующий целое число в произвольную систему счисления.** |
| **3.** | **convert\_from\_n** | **Метод, конвертирующий целое число из произвольной системы счисления в десятичную.** |
| **4.** | **convert\_to\_n\_float** | **Метод, конвертирующий дробное число в произвольную систему счисления.** |
| **5.** | **get\_slices** | **Метод, преобразующий служебную запись числа в произвольной системе счисления в запись числа и системы отдельно.** |
| **6.** | **find\_equals** | **Метод, который находит в списке одинаковые числа, в каких бы системах счисления они не находились.** |
| **7.** | **find\_sum\_in\_n** | **Метод, который находит сумму чисел в разных системах счисления и выводит её в любой, которую укажет пользователь.** |
| **8.** | **\_\_init\_\_** | **Конструктор класса, в данном случае выводящий строку документации.** |

**3.1.3. Используемые стандартные и пользовательские функции в программе**

|  |  |  |
| --- | --- | --- |
| **№** | **Класс или метод** | **Описание функции и/или её назначение** |
| **1.** | **\_\_init\_\_** | **Конструктор класса, в данном случае выводящий строку документации.** |
| **2.** | **convert\_to\_n** | **Метод, конвертирующий целое число в произвольную систему счисления. (Пользовательская функция)** |
| **3.** | **convert\_from\_n** | **Метод, конвертирующий целое число из произвольной системы счисления в десятичную. (Пользовательская функция)** |
| **4.** | **convert\_to\_n\_float** | **Метод, конвертирующий дробное число в произвольную систему счисления. (Пользовательская функция)** |
| **5.** | **get\_slices** | **Метод, преобразующий служебную запись числа в произвольной системе счисления в запись числа и системы отдельно. (Пользовательская функция)** |
| **6.** | **find\_equals** | **Метод, который находит в списке одинаковые числа, в каких бы системах счисления они не находились. (Пользовательская функция)** |
| **7.** | **find\_sum\_in\_n** | **Метод, который находит сумму чисел в разных системах счисления и выводит её в любой, которую укажет пользователь. (Пользовательская функция)** |
| **8.** | **str()** | **Стандартная функция, конвертирующая поступающие в неё данные в тип данных «строка»** |
|  | **chr()** | **Стандартная функция, конвертирующая поступающее в неё целое число в символ** |
|  | **ord()** | **Стандартная функция, конвертирующая поступающий в неё символ в целое число** |
|  | **int()** | **Стандартная функция, конвертирующая поступающие в неё данные в тип данных «целое число»** |
|  | **index()** | **Стандартная функция, принимающая значение и возвращающая индекс первого объекта с этим значением, найденного в итерируемом объекте из которого она вызвана.** |
|  | **append()** | **Стандартная функция, добавляющая полученный объект в итерируемый объект из которого была вызвана.** |
|  | **print()** | **Стандартная функция, выводящая в терминал полученные данные.** |
|  | **sum()** | **Стандартная функция, возвращающая сумму значений дочерних объектов, полученного итерируемого объекта.** |

**3.1.4. Результаты работы программы (скриншоты)**

**

**3.1.5. Текст отлаженной программы**

**class number\_operations:**

**"При указани чисел в какой либо системе счисления, указывайте их в формате 'nba'. Где n - это номер системы счисления, b - это служебный символ, указывающий на разделение, а а - это число в этой системе счисления."**

**def \_\_init\_\_(self):**

**print("При указани чисел в какой либо системе счисления, указывайте их в формате 'nba'. Где n - это номер системы счисления, b - это служебный символ, указывающий на разделение, а а - это число в этой системе счисления.")**

**@classmethod**

**def convert\_to\_n(cls, a: int, n: int):**

**s = ''**

**while a > 0:**

**c = a % n**

**if c < 10:**

**s = str(c) + s**

**else:**

**s = chr(ord('A')+c-10) + s**

**a //= n**

**return s**

**@classmethod**

**def convert\_from\_n(cls, a: str, n: int):**

**power = 0**

**s = 0**

**for i in a[::-1]:**

**if i < 'A':**

**s += int(i) \* (n\*\*power)**

**else:**

**s += (ord(i)-ord('A')+10) \* (n\*\*power)**

**power += 1**

**return s**

**@classmethod**

**def convert\_to\_n\_float(cls, a: float, n: int):**

**s = ''**

**sf = ''**

**sd = ''**

**af = int(a)**

**while af > 0:**

**c = af % n**

**if c < 10:**

**sf = str(c) + sf**

**else:**

**sf = chr(ord('A')+c-10) + sf**

**af //= n**

**ad = a - int(a)**

**to\_count = True**

**while to\_count:**

**ad = ad \* n**

**if int(ad) < 10:**

**sd = sd + str(int(ad))**

**else:**

**sd = sd + chr(ord('A')+int(ad)-10)**

**if int(ad) == 0:**

**to\_count = False**

**ad = ad - int(ad)**

**s = sf + "." + sd**

**return s**

**@classmethod**

**def get\_slices(cls, i: str):**

**n = int(i[:i.index('b')])**

**a = i[i.index('b')+1:]**

**return n, a**

**@classmethod**

**def find\_equals(cls, sp: list):**

**converted\_list = []**

**for i in sp:**

**n, a = cls.get\_slices(i)**

**converted\_list.append(cls.convert\_from\_n(a, n))**

**new\_list = converted\_list**

**for j in converted\_list:**

**if j != "None":**

**equals = sp[converted\_list.index(j)]**

**new\_list[new\_list.index(j)] = "None"**

**for k in new\_list:**

**if str(k) == str(j):**

**equals = equals + '=' + sp[converted\_list.index(k)]**

**new\_list[new\_list.index(k)] = "None"**

**print(equals)**

**@classmethod**

**def find\_sum\_in\_n(cls, sp: list, n1: int):**

**converted\_list = []**

**for i in sp:**

**n, a = cls.get\_slices(i)**

**converted\_list.append(cls.convert\_from\_n(a, n))**

**answer = sum(converted\_list)**

**answer = cls.convert\_to\_n(answer, n1)**

**print(answer)**

**print("Задание 1:")**

**number\_operations.find\_equals(['16b11', '8b6', '4b013', '2b111'])**

**print('')**

**print("Задание 2:")**

**number\_operations.find\_sum\_in\_n(["2b1101", "4b22", "16bA"], 16)**

**print('')**

**print("Задание 3:")**

**print(number\_operations.convert\_to\_n(150, 2))**

**print(number\_operations.convert\_to\_n(150, 16))**

**print(number\_operations.convert\_to\_n(150, 8))**

**print('')**

**print("Задание 4:")**

**print("1011 в десятичной системе счисления:", number\_operations.convert\_from\_n("1011", 2))**

**print("10 в десятичной системе счисления:", number\_operations.convert\_from\_n("10", 2))**

**print("1101 в десятичной системе счисления:", number\_operations.convert\_from\_n("1101", 2))**

**print("Следовательно, знак между ними: +")**

**print('')**

**print("Задание 5:")**

**a = 5/3**

**b = 2/9**

**c = a + b**

**c = number\_operations.convert\_to\_n\_float(c, 3)**

**print(c[:c.index(".")]+c[c.index("."):c.index(".")+4])**

**print("Работу выполнил студент 2023-ФГиИБ-ИСиТ-2б Утягулов Артем.")**

**4. ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ № 7 (ВАРИАНТ 24)**

**4.1. Объектно-ориентированное программирование на Python**

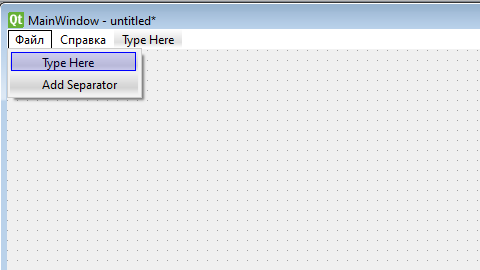
**4.1.1. Постановка задачи №7 в программе**

Разработать программу для вычисления определенного интеграла функции ***f(х)*** (согласно варианту, соответствующему номеру в журнале), включая программный код алгоритма вычисления определенного интеграла и интерфейс для его расчета.

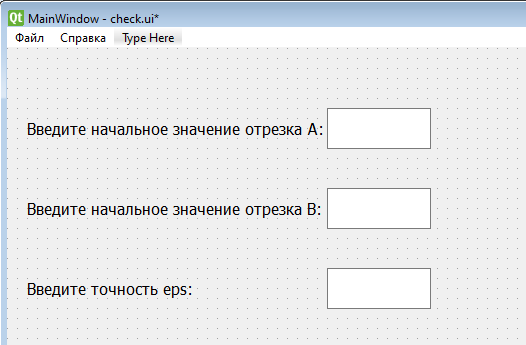
*Самостоятельно* разработать и реализовать в виде программ алгоритмы вычисления определенного интеграла по методу *левых* прямоугольников и по методу *правых* прямоугольников. Сравнить результаты алгоритмов вычисления определенного интеграла по всем трем методам, представив результаты в виде таблицы.

**4.1.2. Разработка пользовательского интерфейса программы с использованием виджетов PyQt**

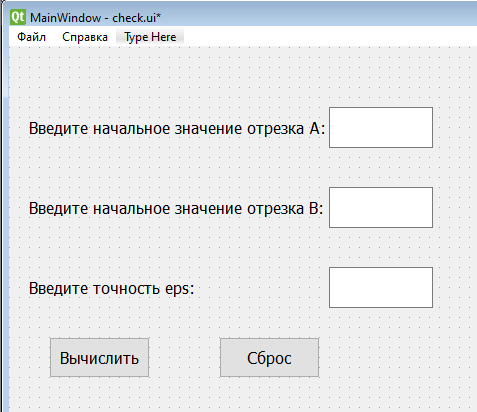
Создание верхнего меню с выходом и справкой с информацией об авторе.



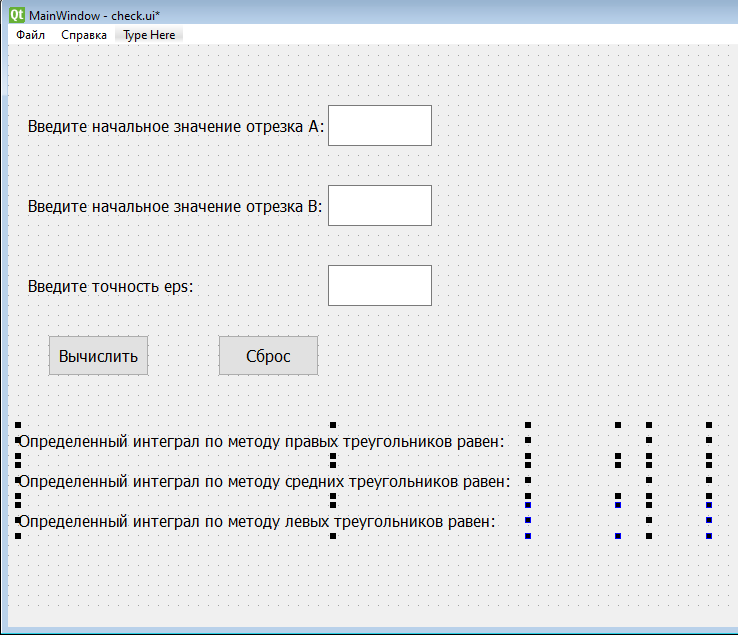
Добавление подписей к полям ввода и самих полей ввода для начального значения, конечного значения и точности eps.



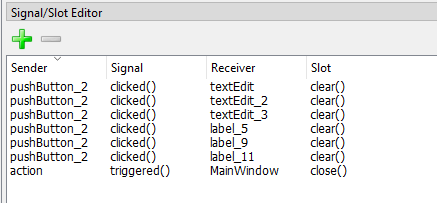
Добавление кнопок вычисления и сброса.



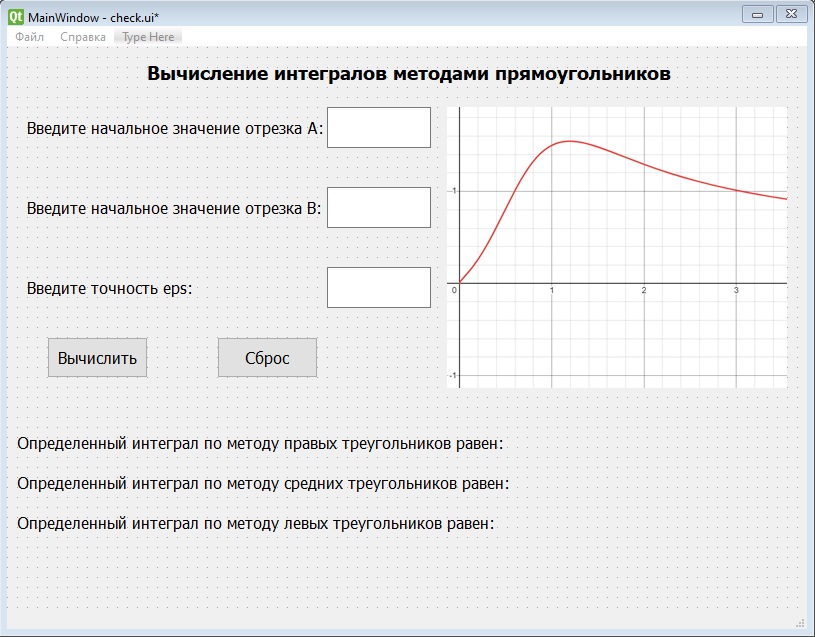
Добавление поля вывода результата для интеграла методом средних прямоугольников и самостоятельная часть: добавление полей вывода результата вычисления интеграла методами правых и левых прямоугольников.



Настройка сигналов кнопки и действий верхнего меню.



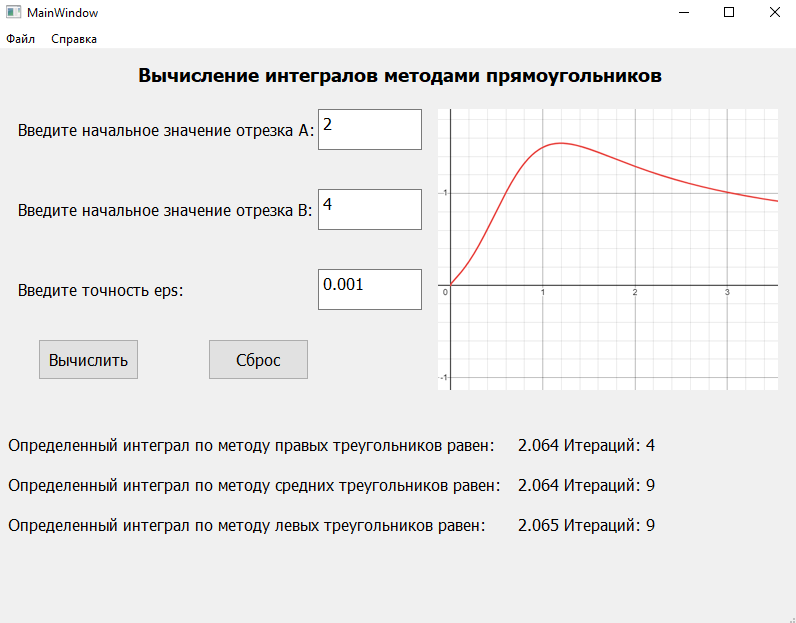
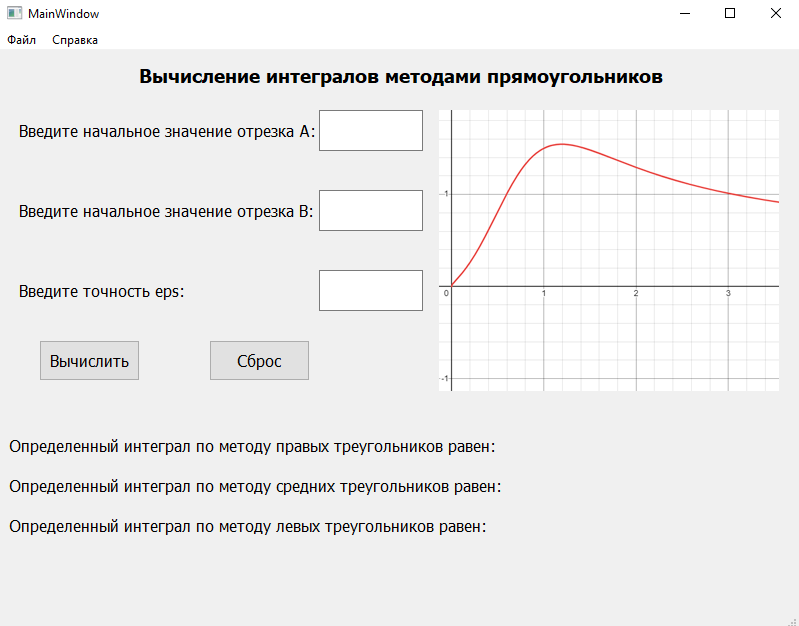
Добавление изображения графика и заголовка.

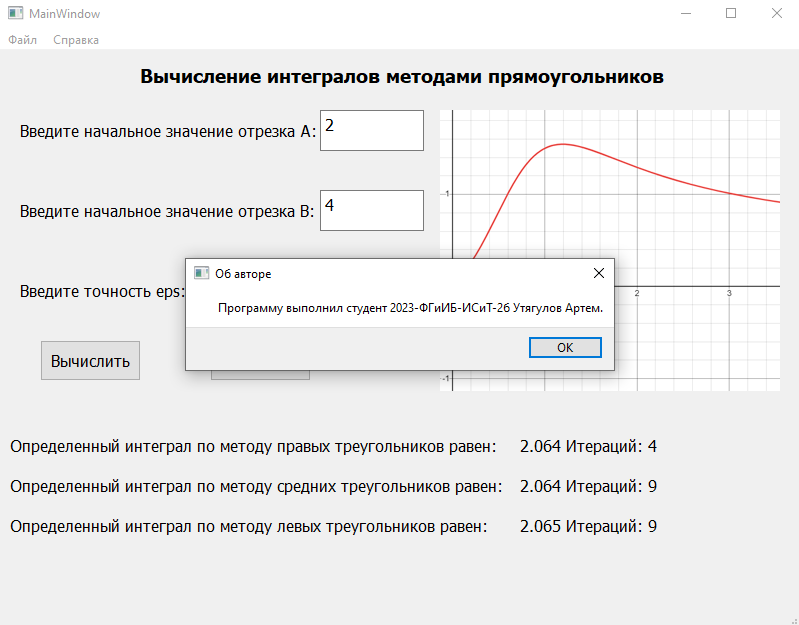


**4.1.3. Используемые библиотеки в программе**

|  |  |
| --- | --- |
| Библиотека | Назначение |
| math | Предоставляет математические функции и константы. |
| sys | Используется для работы с различными параметрами и функциями, специфичными для системы. |
| PyQt5 | Фреймворк для создания графического интерфейса пользователя (GUI). |
| PyQt5.QtWidgets | Подмодуль PyQt5, содержащий классы для элементов GUI, таких как кнопки, окна и диалоговые окна. |

**4.1.4. Результаты работы программы (скриншоты)**





**4.1.5. Текст отлаженной программы**

import sys

from math import log

import ui\_file

from PyQt5 import QtWidgets

from PyQt5.QtWidgets import QMessageBox

class MyWin(QtWidgets.QMainWindow):

def \_\_init\_\_(self, parent=None):

QtWidgets.QWidget.\_\_init\_\_(self, parent)

self.ui = ui\_file.Ui\_MainWindow()

self.ui.setupUi(self)

self.ui.pushButton.clicked.connect(self.Integral)

self.ui.action\_2.triggered.connect(self.Info)

def Info(self):

QMessageBox.about(self, "Об авторе", "Программу выполнил студент 2023-ФГиИБ-ИСиТ-2б Утягулов Артем.")

def Integral(self):

a = int(self.ui.textEdit.toPlainText())

b = int(self.ui.textEdit\_2.toPlainText())

eps = float(self.ui.textEdit\_3.toPlainText())

answer, iter = self.calculate(a, b, eps)

answer\_left, iter\_left = self.calculate\_left(a, b, eps)

answer\_right, iter\_right = self.calculate\_right(a, b, eps)

self.ui.label\_5.setText("{x:.3f}".format(x=answer) + " Итераций: " + str(iter))

self.ui.label\_9.setText("{x:.3f}".format(x=answer\_right) + " Итераций: " + str(iter\_right))

self.ui.label\_11.setText("{x:.3f}".format(x=answer\_left) + " Итераций: " + str(iter\_left))

def f(self, x):

return (0.5+x\*\*0.5)/(1+log(x)\*\*2)

def get\_area(self, a, b, n):

deltaX = (b - a) / n

multiplicator = 1

if str(deltaX).count(".") != 0:

multiplicator = 10\*\*len(str(deltaX).split(".")[1])

full\_sum = 0

for i in range(a\*multiplicator+int(deltaX\*multiplicator), b\*multiplicator+1, int(deltaX\*multiplicator)):

i = i / multiplicator

s = self.f(i-deltaX/2)\*deltaX

full\_sum += s

return full\_sum

def get\_area\_left(self, a, b, n):

deltaX = (b - a) / n

multiplicator = 1

if str(deltaX).count(".") != 0:

multiplicator = 10\*\*len(str(deltaX).split(".")[1])

full\_sum = 0

for i in range(a\*multiplicator, b\*multiplicator+1-int(deltaX\*multiplicator), int(deltaX\*multiplicator)):

i = i / multiplicator

s = self.f(i)\*deltaX

full\_sum += s

return full\_sum

def get\_area\_right(self, a, b, n):

deltaX = (b - a) / n

multiplicator = 1

if str(deltaX).count(".") != 0:

multiplicator = 10\*\*len(str(deltaX).split(".")[1])

full\_sum = 0

for i in range(a\*multiplicator+int(deltaX\*multiplicator), b\*multiplicator+1, int(deltaX\*multiplicator)):

i = i / multiplicator

s = self.f(i)\*deltaX

full\_sum += s

return full\_sum

def calculate(self, a, b, eps):

done = False

full\_sum = 0

n = 1

iter = 0

while done != True:

S1 = self.get\_area(a, b, n)

n = n \* 2

S2 = self.get\_area(a, b, n)

iter += 1

if abs(S2-S1) < eps:

full\_sum = S2

done = True

return full\_sum, iter

def calculate\_left(self, a, b, eps):

done = False

full\_sum = 0

n = 1

iter = 0

while done != True:

S1 = self.get\_area\_left(a, b, n)

n = n \* 2

S2 = self.get\_area\_left(a, b, n)

iter += 1

if abs(S2-S1) < eps:

full\_sum = S2

done = True

return full\_sum, iter

def calculate\_right(self, a, b, eps):

done = False

full\_sum = 0

n = 1

iter = 0

while done != True:

S1 = self.get\_area\_right(a, b, n)

n = n \* 2

S2 = self.get\_area\_right(a, b, n)

iter += 1

if abs(S2-S1) < eps:

full\_sum = S2

done = True

return full\_sum, iter

if \_\_name\_\_ == "\_\_main\_\_":

app = QtWidgets.QApplication(sys.argv)

myapp = MyWin()

myapp.show()

sys.exit(app.exec\_())

**ВЫВОДЫ**

1. Мною были выполнены индивидуальные задания №4a, №4b, №5a, №5b, №5c, №6, №7

2. Были обнаружены и отлажены ошибки синтаксиса, ошибки работы с классами и основные ошибки в алгоритмах.

3. Мне понравились оба вида программирования, потому, что для решения многих задач эффективнее всего было их комбинировать.

**СПИСОК ИСПОЛЬЗОВАННЫХ ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ**

1. Комп. Практикум основы ООП на ЯП Python – Лапчинская М.П.

2. Методические указания «Вычисление определенного интеграла

методом средних прямоугольников на императивном и

объектно-ориентированном языке программирования Python» – Лапчинская М.П.