**Московский Государственный Университет Геодезии и Картографии**

**Факультет геоинформатики и информационной безопасности**

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

Направление: Прикладная информатика

**Отчет**

**о выполненных практических работах по дисциплине ИНФОРМАТИКА**

*(индивидуальные задания: №4а, №4b, №5a=№3, №5b, №5c №6, №7)*

Проверила: Выполнил:

доц. кафедры ИС студент 2023-ФГиИБ-ПИ-1(б)

Лапчинская М.П. Беспечалов А.А.

**МОСКВА – 2024**

**СОДЕРЖАНИЕ**

**стр.**

**ВВЕДЕНИЕ**

**1. ПОСТАНОВКА ЗАДАЧИ ИНДИВИДУАЛЬНОГО ЗАДАНИЯ №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. ПОСТАНОВКА ЗАДАЧИ ИНДИВИДУАЛЬНОГО ЗАДАНИЯ № 5 (ВАРИАНТ …)**

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

**2.1.1. Программирование без пользовательских функций (№5a=№3)**

2.1.1.1. Используемые инструкции в программе

2.1.1.2. Блок-схема решения задачи

2.1.1.3. Результаты работы программы (скриншоты)

2.1.1.4. Текст отлаженной программы

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

2.1.2.1. Используемые инструкции в программе

2.1.2.2. Блок-схема решения задачи

2.1.2.3. Результаты работы программы (скриншоты)

2.1.2.4. Текст отлаженной программы

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

2.2.1. Используемые инструкции в программе

2.2.2. Результаты работы программы (скриншоты)

2.2.3. Текст отлаженной программы

**3. ПОСТАНОВКА ЗАДАЧИ ИНДИВИДУАЛЬНОГО ЗАДАНИЯ № 6 (ВАРИАНТ…)**

**3.1. Объектно-ориентированное программирование на Python (№6)**

3.1.1. Используемые инструкции в программе

3.1.2. Результаты работы программы (скриншоты)

3.1.3. Текст отлаженной программы

**4. ПОСТАНОВКА ЗАДАЧИ ИНДИВИДУАЛЬНОГО ЗАДАНИЯ № 7 (ВАРИАНТ …)**

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

4.1.1. Описание алгоритма вычисления определенного интеграла по методу средних прямоугольников

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)

4.1.5. Текст отлаженной программы

**ВЫВОДЫ**

**СПИСОК ИСПОЛЬЗОВАННЫХ ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ**

**1. ПОСТАНОВКА ЗАДАЧИ ИНДИВИДУАЛЬНОГО ЗАДАНИЯ №4 (ВАРИАНТ 21)**

Дан двумерный массив **A** размером **n\*m** элементов, заполненный целыми случайными целыми 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

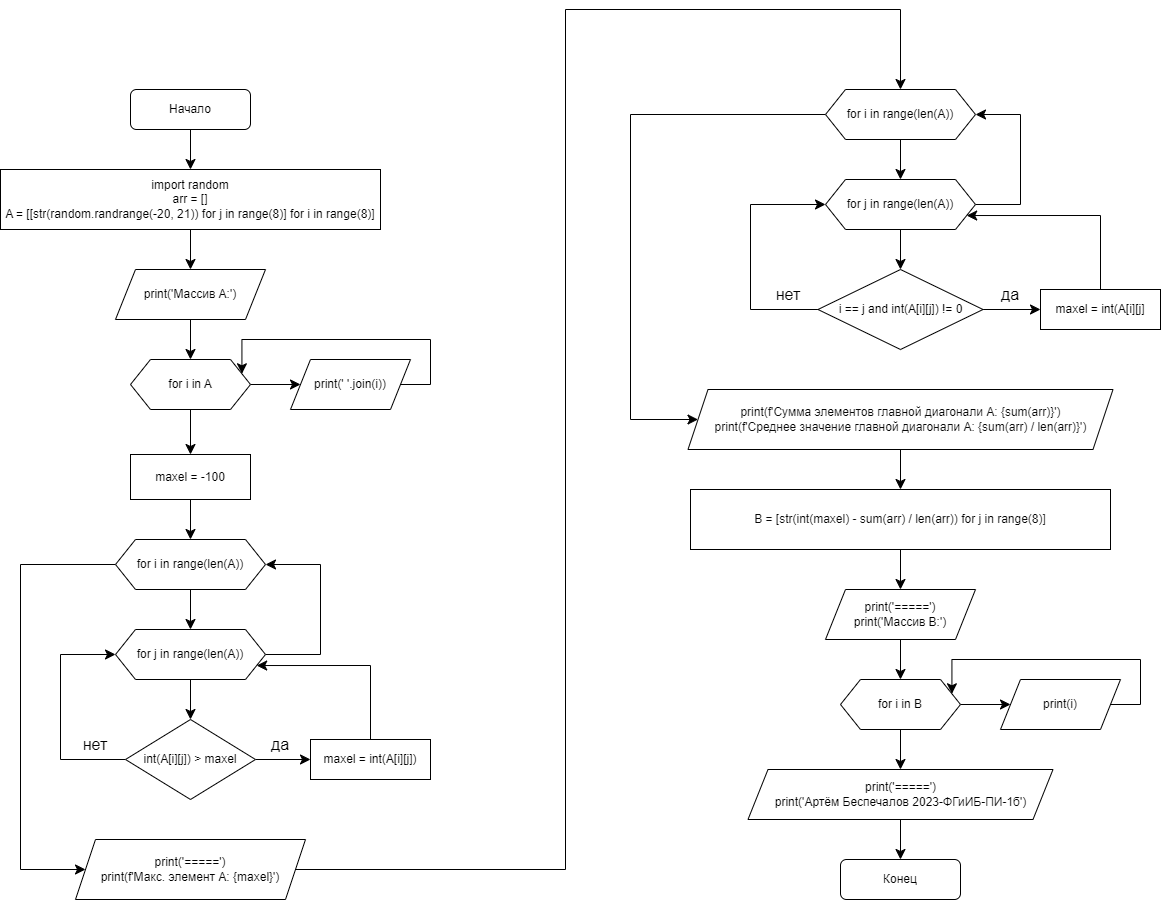
Дан двумерный массив **A** размером **n\*m** элементов, заполненный целыми случайными целыми числами из диапазона **(r1, r2)**. Вывести на экран исходный массив и результаты вычислений/преобразований с точностью до **3-го** знака.

1. Двумерный массив описать с использованием списочного типа данных.

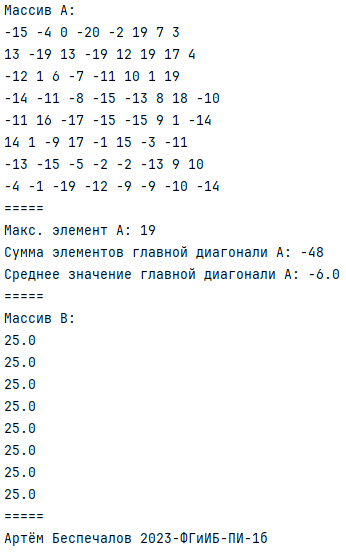
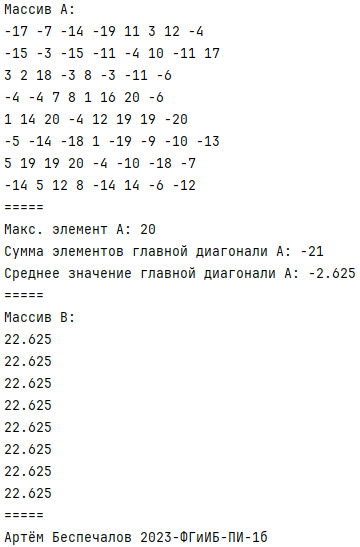
**1.1.1.2. Используемые инструкции в программе**

|  |  |  |
| --- | --- | --- |
| **№** | **Инструкция  (из кода программы)** | **Название и назначение инструкции** |
| **1.** | import random | **Импорт библиотеки “random”** |
| **2.** | arr = [] | **Инструкция присваивания. Присваиваем переменной arr пустой массив** |
| **3.** | A = [[str(random.randrange(-20, 21)) for j in range(8)] for i in range(8)] | **Инструкция цикла + инструкция присваивания. С помощью генератора создаём двумерный массив A размером 8х8, элементами которого являются целые числа в диапазоне (-20,20).** |
| **4.** | for i in A: | **Инструкция цикла. Запускаем цикл размера матрицы A.** |
| **5.** | print(' '.join(i)) | **Инструкция вывода. Выводим элементы матрицы A.** |
| **6.** | maxel = -100 | **Инструкция присваивания. Присваиваем переменной maxel значение -100.** |
| **7.** | for i in range(len(A)): | **Инструкция цикла. Запускаем цикл длиной матрицы A.** |
| **8.** | for j in range(len(A)): | **Инструкция цикла. Запускаем новый цикл, вложенный в предыдущий, длиной матрицы A.** |
| **9.** | if int(A[i][j]) > maxel:  maxel = int(A[i][j]) | **Инструкция ветвления. Если элемент матрицы A с индексами i и j больше значения переменной maxel, то выполняем инструкцию присваивания переменной maxel значения элемента матрицы A с индексами i и j.** |
| **10.** | for i in range(len(A)): | **Инструкция цикла. Запускаем цикл размера матрицы A.** |
| **11.** | for j in range(len(A)): | **Инструкция цикла. Запускаем новый цикл, вложенный в предыдущий, размера матрицы A.** |
| **12.** | if i == j and int(A[i][j]) != 0:  arr.append(int(A[i][j])) | **Инструкция ветвления. Если индекс i не равен индексу j и элемент матрицы arr с индексами i и j не равен 0, то выполняем инструкцию добавления в массив arr элемента матрицы A с индексами i и j.** |
| **13.** | print(f'Сумма элементов главной диагонали A: {sum(arr)}') | **Инструкция вывода. Выводим сумму элементов массива arr.** |
| **14.** | print(f'Среднее значение главной диагонали A: {sum(arr) / len(arr)}') | **Инструкция вывода. Выводим значение операции sum(arr) / len(arr).** |
| **15.** | B = [str(int(maxel) - sum(arr) / len(arr)) for j in range(8)] | **Инструкция цикла + инструкция присваивания. С помощью генератора создаём одномерный массив B размером 8,** каждый элемент которого является разностью, между максимальным значением двумерного массива A и средним значением главной диагонали A. |
| **16.** | for i in B: | **Инструкция цикла. Запускаем цикл длины массива B.** |
| **17.** | print(i) | **Инструкция вывода. Выводим элементы массива B.** |

**1.1.1.3. Блок-схема решения задачи**



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

**1.1.1.5. Текст отлаженной программы**

import random  
  
arr = []  
A = [[str(random.randrange(-20, 21)) for j in range(8)] for i in range(8)]  
  
print('Массив A:')  
for i in A:  
 print(' '.join(i))  
  
maxel = -100  
for i in range(len(A)):  
 for j in range(len(A)):  
 if int(A[i][j]) > maxel:  
 maxel = int(A[i][j])  
  
print('=====')  
print(f'Макс. элемент A: {maxel}')  
  
for i in range(len(A)):  
 for j in range(len(A)):  
 if i == j and int(A[i][j]) != 0:  
 arr.append(int(A[i][j]))  
  
print(f'Сумма элементов главной диагонали A: {sum(arr)}')  
print(f'Среднее значение главной диагонали A: {sum(arr) / len(arr)}')  
  
B = [str(int(maxel) - sum(arr) / len(arr)) for j in range(8)]  
  
print('=====')  
print('Массив B:')  
for i in B:  
 print(i)  
print('=====')  
print('Артём Беспечалов 2023-ФГиИБ-ПИ-1б')

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

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

**1.1.2.1. Постановка задачи №4b**

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

1. Двумерный массив описать с использованием списочного типа данных

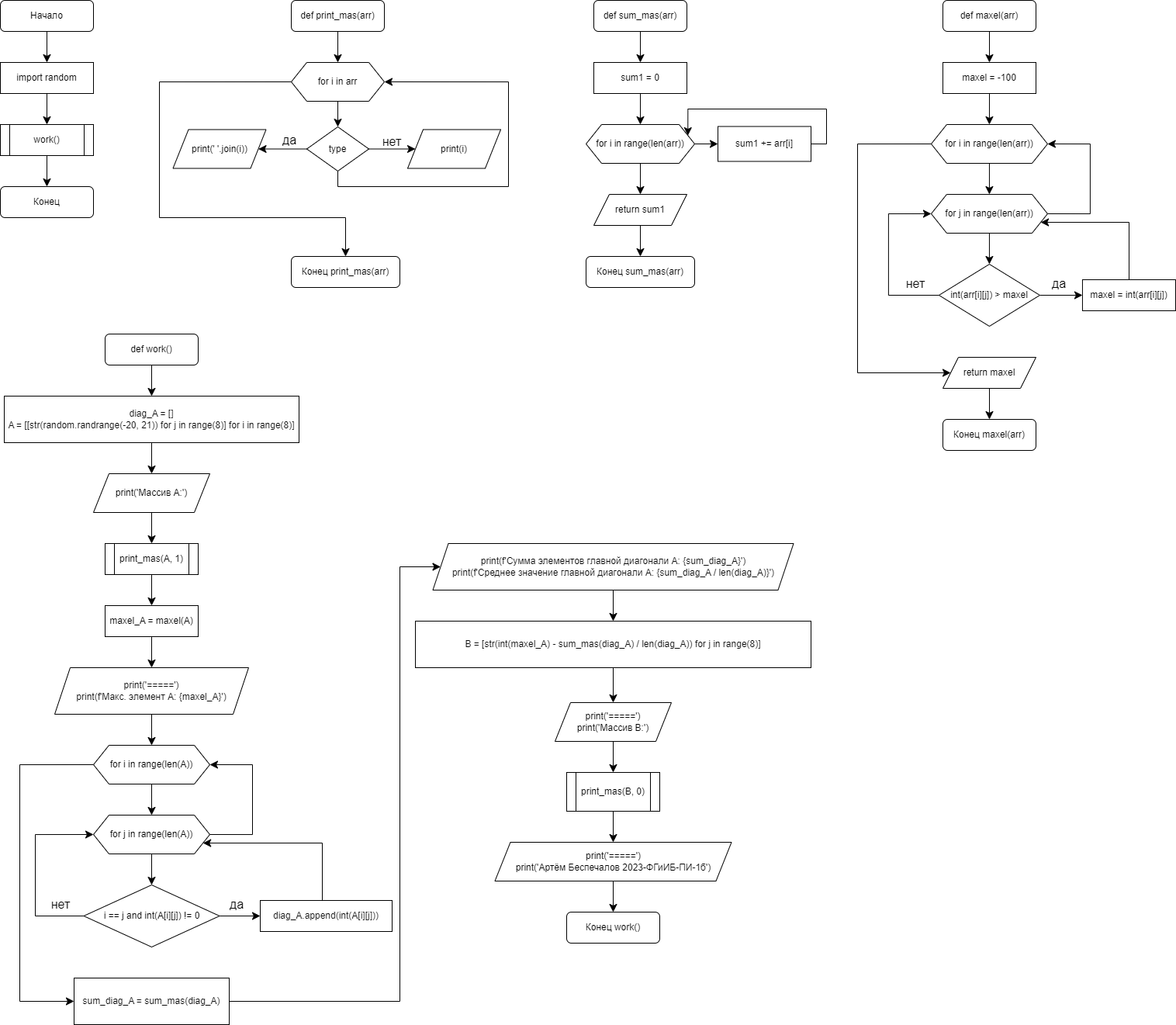
2. Не использовать стандартные функции min, max, sum и т.п.   
(написать свои функции).

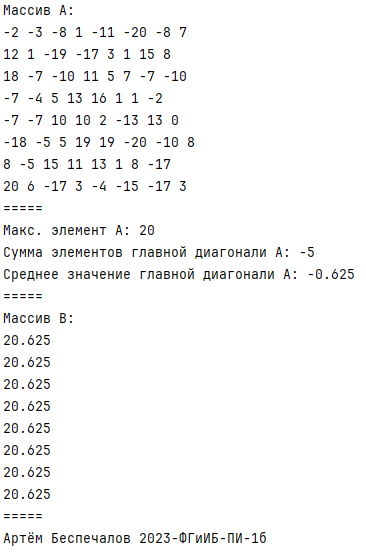
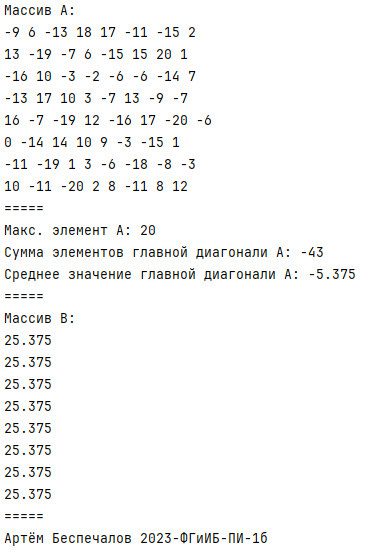
3. Результат представить в виде пользовательской функции

**1.1.2.2. Используемые инструкции в программе**

|  |  |  |
| --- | --- | --- |
| **№** | **Инструкция  (из кода программы)** | **Название и назначение инструкции** |
| **1.** | import random | **Импорт библиотеки “random”** |
| **2.** | def sum\_mas(arr): | **Инструкция процедуры. Инициализация процедуры sum\_mas(arr), принимающую на вход массив и возвращающую сумму элементов массива.** |
| **3.** | sum1 = 0 | **Инструкция присваивания. Переменной sum1 присваиваем значения 0.** |
| **4.** | for i in range(len(arr)): | **Инструкция цикла. Запускаем цикл длиной массива arr.** |
| **5.** | sum1 += arr[i] | **Инструкция присваивания. Переменной sum1 присваиваем значение результата операции sum1 + arr[i].** |
| **6.** | return sum1 | **Инструкция возврата значения. Возвращаем значение переменной sum1.** |
| **7.** | def maxel(arr): | **Инструкция процедуры. Инициализация процедуры maxel(arr), принимающую на вход матрицу arr и возвращающую значение максимального элемента матрицы arr.** |
| **8.** | maxel = -100 | **Инструкция присваивания. Переменной maxel присваиваем значение -100.** |
| **9.** | for i in range(len(arr)): | **Инструкция цикла. Запускаем цикл размера матрицы arr.** |
| **10.** | for j in range(len(arr)): | **Инструкция цикла. Запускаем новый цикл, вложенный в предыдущий, размера матрицы arr.** |
| **11.** | if int(arr[i][j]) > maxel:  maxel = int(arr[i][j]) | **Инструкция ветвления. Если элемент матрицы arr с индексами i и j больше значения переменной maxel, то выполняем инструкцию присваивания переменной maxel значения элемента матрицы arr с индексами i и j.** |
| **12.** | return maxel | **Инструкция возврата значения. Возвращаем значение переменной maxel.** |
| **13.** | def print\_mas(arr, type): | **Инструкция процедуры. Инициализация процедуры print\_mas(arr), принимающую на вход массив, тип массива(одномерный или двумерный) и использующая инструкцию вывода матрицы.** |
| **14.** | for i in arr: | **Инструкция цикла. Запускаем цикл размера матрицы arr.** |
| **15.** | if type:  print(‘ ‘.join(i)) | **Инструкция ветвления. Если массив двумерный, то выполняется инструкция вывода строки матрицы.** |
| **16.** | else:  print(i) | **Инструкция ветвления. Если не выполняется условие 15(массив одномерный), то выполняется инструкция вывода элемента массива.** |
| **17.** | def work(): | **Инструкция процедуры. Инициализация процедуры work(), ничего не принимающую на вход.** |
| **18.** | diag\_A = [] | **Инструкция присвания. Переменной diag\_A присваиваем пустой массив.** |
| **19.** | A = [[str(random.randrange(-20, 21)) for j in range(8)] for i in range(8)] | **Инструкция цикла + инструкция присваивания. С помощью генератора создаём двумерный массив A размером 8х8, элементами которого являются целые числа в диапазоне (-20,20).** |
| **20.** | print\_mas(A, 1) | **Инструкция вызова. Вызов процедуры print\_mas(arr) c матрицей A и 1(двумерный массив) в качестве аргументов.** |
| **21.** | maxel\_A = maxel(A) | **Инструкция присваивания. Присваиваем переменной maxel\_A значение результата работы процедуры maxel() c матрицей A в качестве аргумента.** |
| **22.** | for i in range(len(A)): | **Инструкция цикла. Запускаем цикл размера матрицы A.** |
| **23.** | for j in range(len(A)): | **Инструкция цикла. Запускаем новый цикл, вложенный в предыдущий, размера матрицы A.** |
| **24.** | if i == j and int(A[i][j]) != 0:  diag\_A.append(int(A[i][j])) | **Инструкция ветвления. Если индекс i не равен индексу j и элемент матрицы arr с индексами i и j не равен 0, то выполняем инструкцию добавления в массив diag\_A элемента матрицы A с индексами i и j.** |
| **25.** | sum\_diag\_A = sum\_mas(diag\_A) | **Инструкция присваивания. Присваиваем переменной sum\_diag\_A значение результата работы процедуры sum\_mas() c массивом diag\_A в качестве аргумента.** |
| **26.** | print(f'Сумма элементов главной диагонали A: {sum\_diag\_A}') | **Инструкция вывода. Выводим значение переменной sum\_diag\_A.** |
| **27.** | print(f'Среднее значение главной диагонали A: {sum\_diag\_A / len(diag\_A)}') | **Инструкция вывода. Выводим значение операции sum\_diag\_A / len(diag\_A).** |
| **28.** | B = [str(int(maxel\_A) - sum\_mas(diag\_A) / len(diag\_A)) for j in range(8)] | **Инструкция цикла + инструкция присваивания. С помощью генератора создаём одномерный массив B размером 8,** каждый элемент которого является разностью, между максимальным значением двумерного массива A и средним значением главной диагонали A. |
| **29.** | print\_mas(B, 0) | **Инструкция вызова. Вызов процедуры print\_mas(arr) c матрицей B и 0(одномерный массив) в качестве аргументов.** |
| **30.** | work() | **Инструкция вызова. Вызов процедуры work().** |

**1.1.2.3. Блок-схема решения задачи**

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

** **

**1.1.2.5. Текст отлаженной программы**

import random  
  
def sum\_mas(arr):  
 sum1 = 0  
 for i in range(len(arr)):  
 sum1 += arr[i]  
 return sum1  
  
def maxel(arr):  
 maxel = -100  
 for i in range(len(arr)):  
 for j in range(len(arr)):  
 if int(arr[i][j]) > maxel:  
 maxel = int(arr[i][j])  
 return maxel  
  
def print\_mas(arr):  
 for i in arr:  
 print(' '.join(i))  
  
def work():  
 diag\_A = []  
 A = [[str(random.randrange(-20, 21)) for j in range(8)] for i in range(8)]  
 print('Массив A:')  
 print\_mas(A)  
 maxel\_A = maxel(A)  
 print('=====')  
 print(f'Макс. элемент A: {maxel\_A}')  
 for i in range(len(A)):  
 for j in range(len(A)):  
 if i == j and int(A[i][j]) != 0:  
 diag\_A.append(int(A[i][j]))  
 sum\_diag\_A = sum\_mas(diag\_A)  
 print(f'Сумма элементов главной диагонали A: {sum\_diag\_A}')  
 print(f'Среднее значение главной диагонали A: {sum\_diag\_A / len(diag\_A)}')  
 B = [[str(int(maxel\_A) - sum\_mas(diag\_A) / len(diag\_A)) for j in range(8)] for i in range(8)]  
 print('=====')  
 print('Массив B:')  
 print\_mas(B)  
 print('=====')  
 print('Артём Беспечалов 2023-ФГиИБ-ПИ-1б')  
  
work()

**2. ПОСТАНОВКА ЗАДАЧИ ИНДИВИДУАЛЬНОГО ЗАДАНИЯ № 5 (ВАРИАНТ 21)**

Взять инд. задание **№3** (до внутрисеместровой аттестации – см. ***Таблица 1***), которая представляет собой инд. задание **№5а** *без пользовательских функций* т.е. **№5а = №3**;

**2.1. Структурное (императивное) программирование**

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

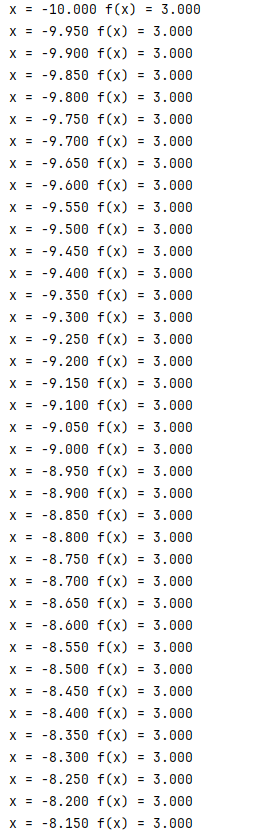
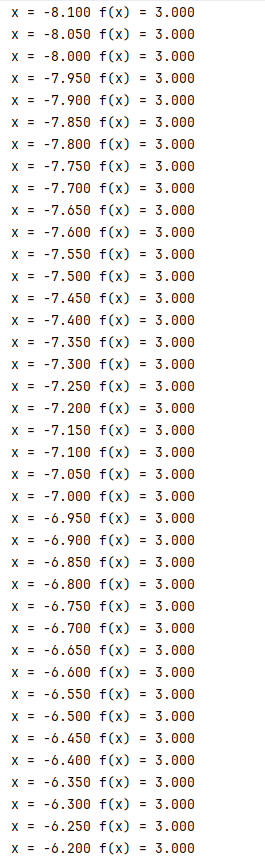
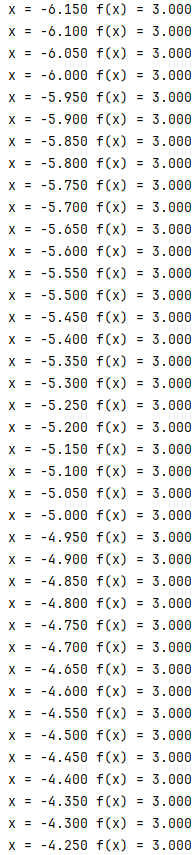
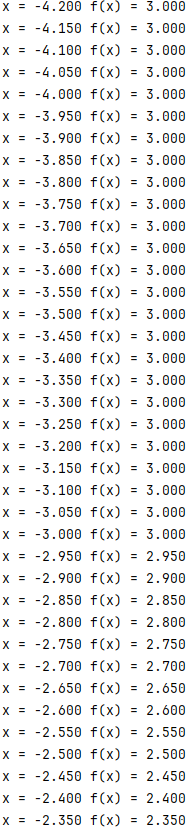
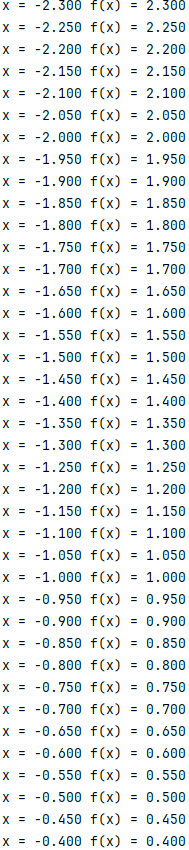
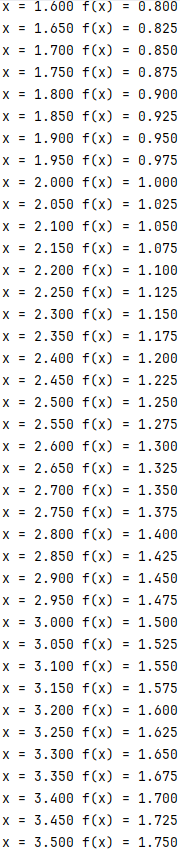
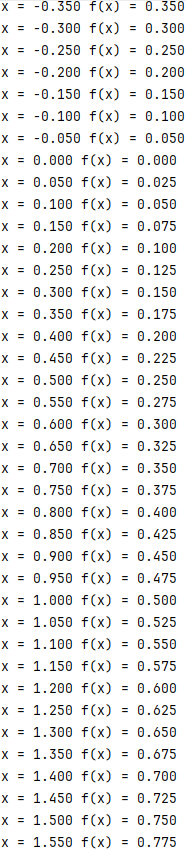
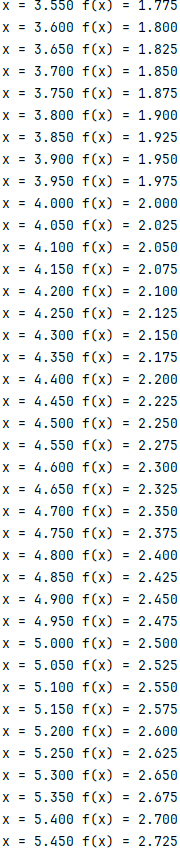
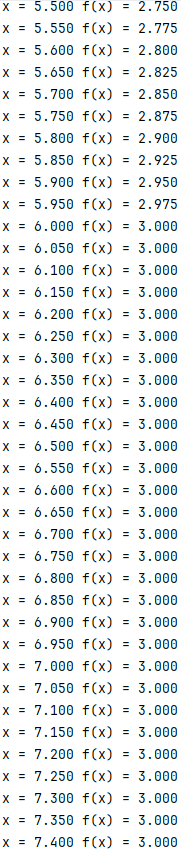
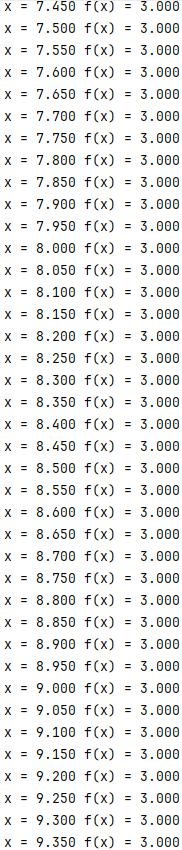
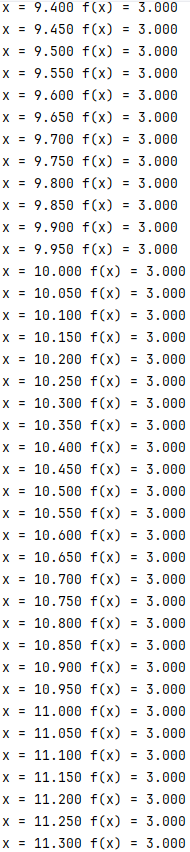
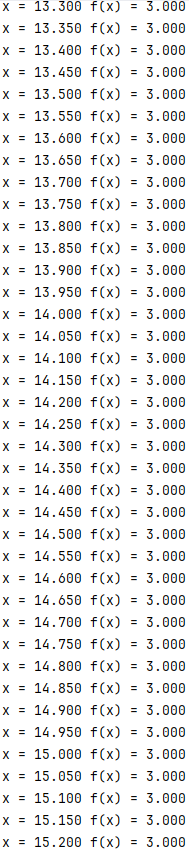
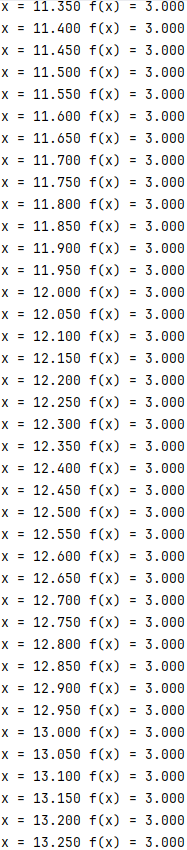
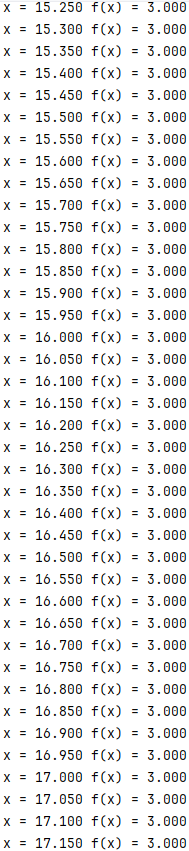
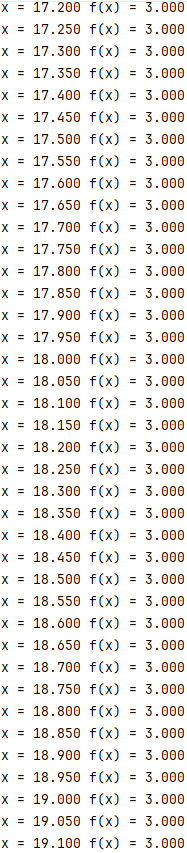
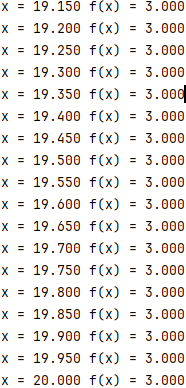
**2.1.1.1. Используемые инструкции в программе**

|  |  |  |
| --- | --- | --- |
| **№** | **Инструкция  (из кода программы)** | **Название и назначение инструкции** |
| **1.** | for x in [i / 100 for i in range(-1000, 2005, 5)]: | **Инструкция цикла for. Перебираем значения x в заданном диапазоне.** |
| **2.** | if x <= -3 or x >= 6:  print(f'x = {"{:.3f}".format(x)} f(x) = {"{:.3f}".format(3)}') | **Инструкция ветвления. Если условие истинно, выполняется инструкция вывода** |
| **3.** | elif x > -3 and x < 0:  print(f'x = {"{:.3f}".format(x)} f(x) = {"{:.3f}".format(-1 \* x)}') | **Инструкция ветвления. Если условие 2 не выполняется, то проверяется истинность этого условия. Если условие истинно, выполняется инструкция вывода.** |
| **4.** | elif x >= 0 and x < 6:  print(f'x = {"{:.3f}".format(x)} f(x) = {"{:.3f}".format(x \* 0.5)}') | **Инструкция ветвления. Если условие 3 не выполняется, то проверяется истинность этого условия. Если условие истинно, выполняется инструкция вывода.** |

**2.1.1.2. Блок-схема решения задачи**

**

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

*             *

**2.1.1.4. Текст отлаженной программы**

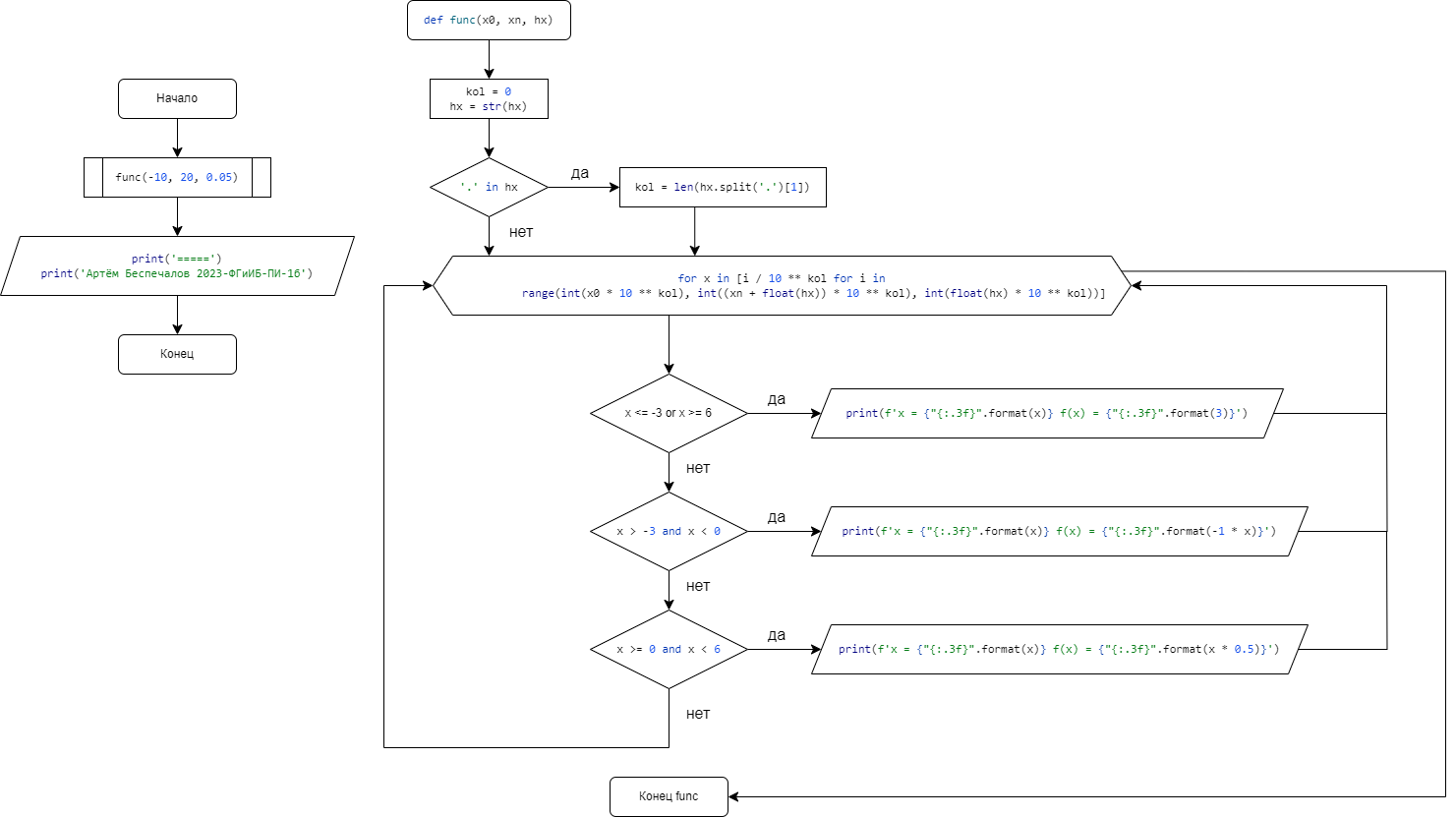
import math  
  
for x in [i / 100 for i in range(-1000, 2005, 5)]:  
 if x <= -3 or x >= 6:  
 print(f'x = {"{:.3f}".format(x)} f(x) = {"{:.3f}".format(3)}')  
 elif x > -3 and x < 0:  
 print(f'x = {"{:.3f}".format(x)} f(x) = {"{:.3f}".format(-1 \* x)}')  
 elif x >= 0 and x < 6:  
 print(f'x = {"{:.3f}".format(x)} f(x) = {"{:.3f}".format(x \* 0.5)}')

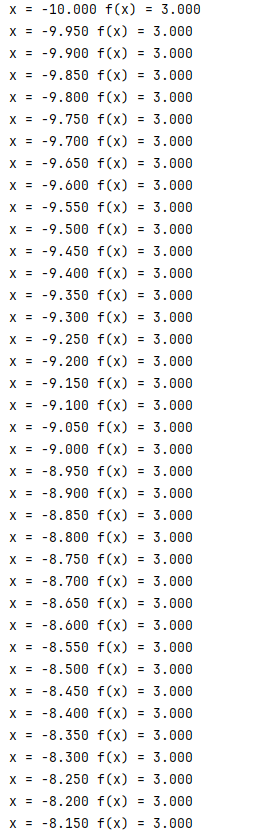
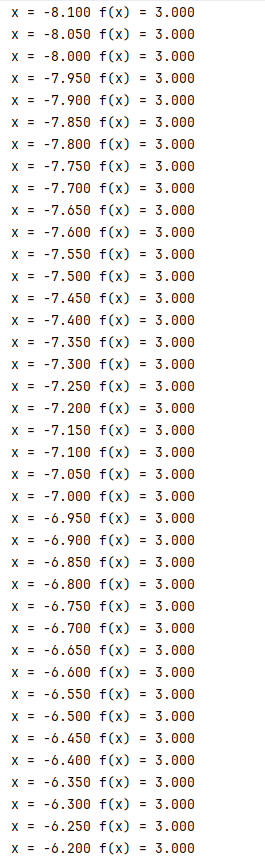
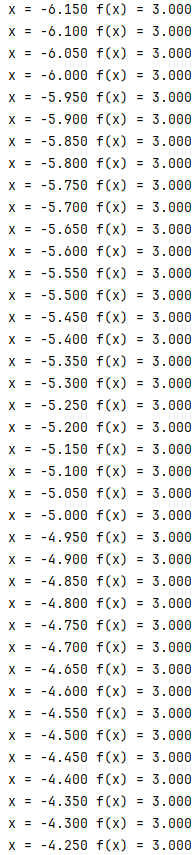
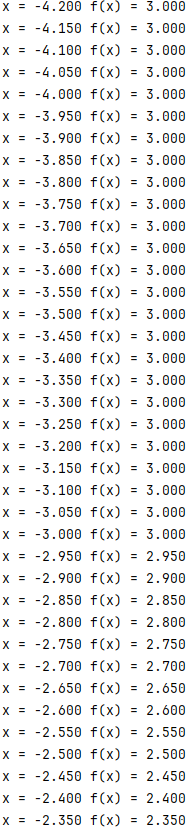
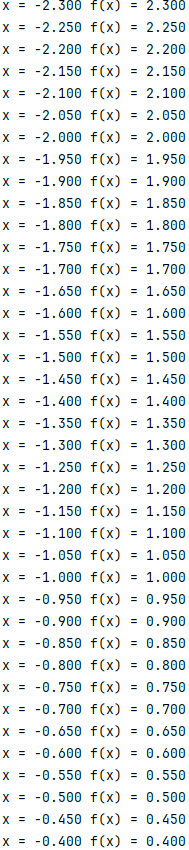
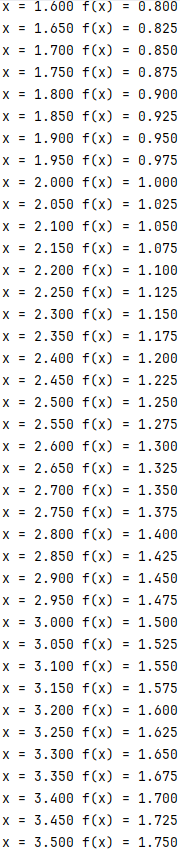
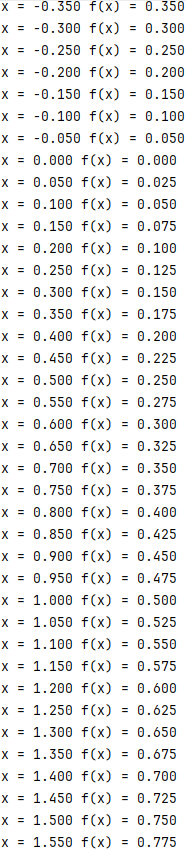
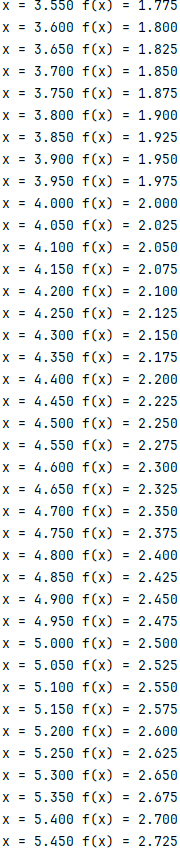
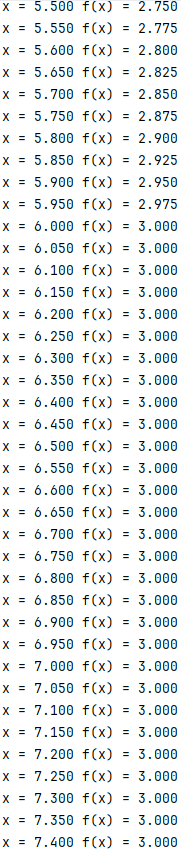
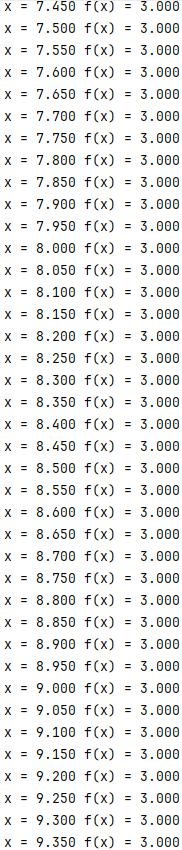
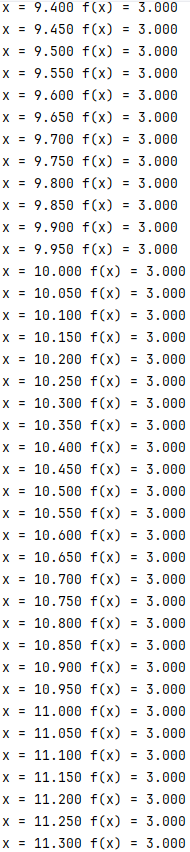
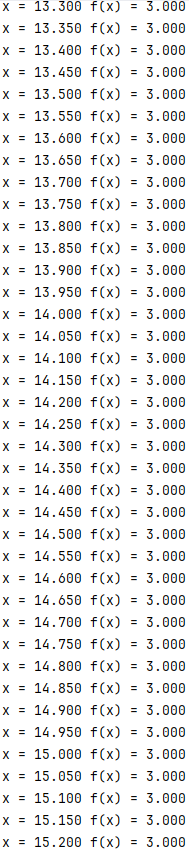
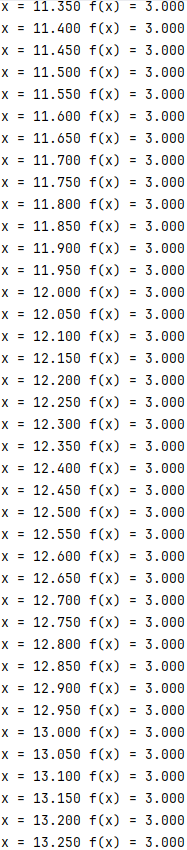
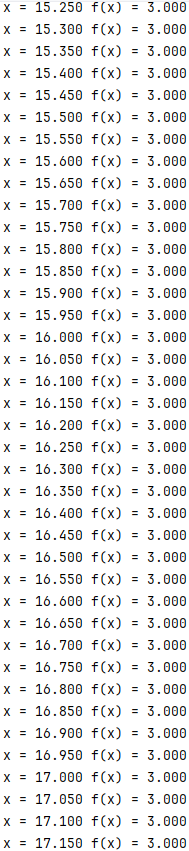
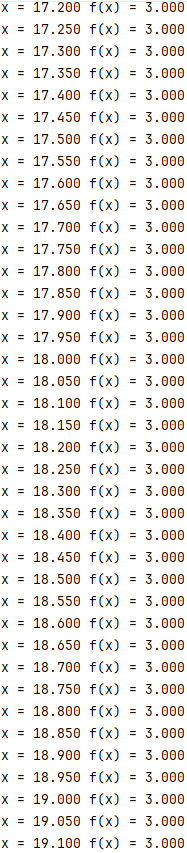
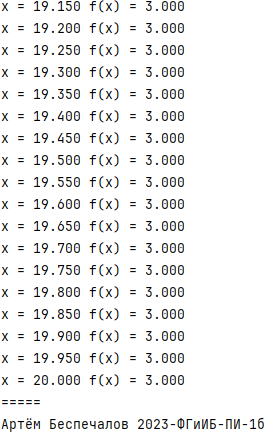
**2.1.2. Программирование с пользовательскими функциями (№5b)**

**2.1.2.1. Используемые инструкции в программе**

|  |  |  |
| --- | --- | --- |
| **№** | **Инструкция  (из кода программы)** | **Название и назначение инструкции** |
| **1.** | def func(x0, xn, hx): | **Инициализация процедуры function, получающей на вход x0, xn, hx.** |
| **2.** | kol = 0 | **Инструкция присваивания переменной kol значения 0.** |
| **3.** | hx = str(hx) | **Инструкция присваивания переменной hx значения hx, конвертированного в строку.** |
| **4.** | if '.' in hx:  kol = len(hx.split('.')[1]) | **Инструкция ветвления. Если число hx – дробное, то выполняется инструкция присваивания переменной kol значения количества знаков после запятой в числе hx.** |
| **5.** | for x in [i / 10 \*\* kol for i in  range(int(x0 \* 10 \*\* kol), int((xn + float(hx)) \* 10 \*\* kol), int(float(hx) \* 10 \*\* kol))]: | **Инструкция цикла for. Перебираем значения x в заданном диапазоне.** |
| **6.** | if x <= -3 or x >= 6:  print(f'x = {"{:.3f}".format(x)} f(x) = {"{:.3f}".format(3)}') | **Инструкция ветвления. Если условие истинно, выполняется инструкция вывода** |
| **7.** | elif x > -3 and x < 0:  print(f'x = {"{:.3f}".format(x)} f(x) = {"{:.3f}".format(-1 \* x)}') | **Инструкция ветвления. Если условие 6 не выполняется, то проверяется истинность этого условия. Если условие истинно, выполняется инструкция вывода.** |
| **8.** | elif x >= 0 and x < 6:  print(f'x = {"{:.3f}".format(x)} f(x) = {"{:.3f}".format(x \* 0.5)}') | **Инструкция ветвления. Если условие 7 не выполняется, то проверяется истинность этого условия. Если условие истинно, выполняется инструкция вывода.** |
| **9.** | func(-10, 20, 0.05) | **Вызов процедуры function с входными данными x0=-10, xn=20, hx=0.05** |

**2.1.2.2. Блок-схема решения задачи**

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

*             *

**2.1.2.4. Текст отлаженной программы**

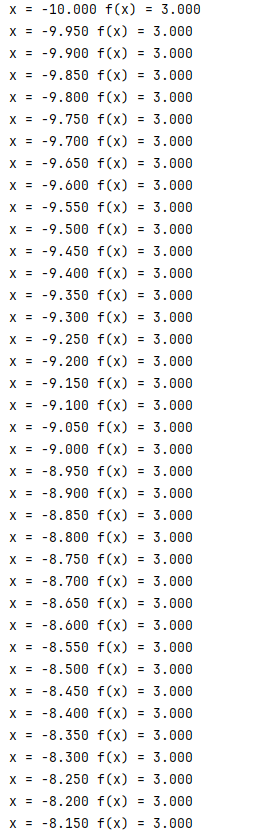
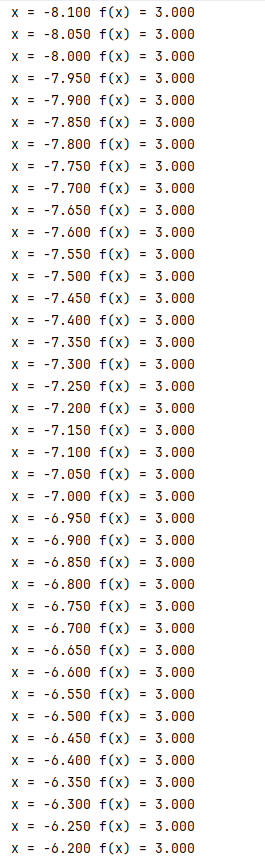
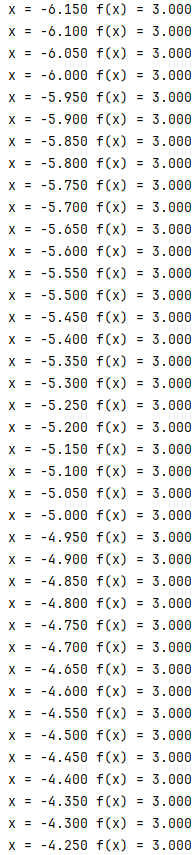
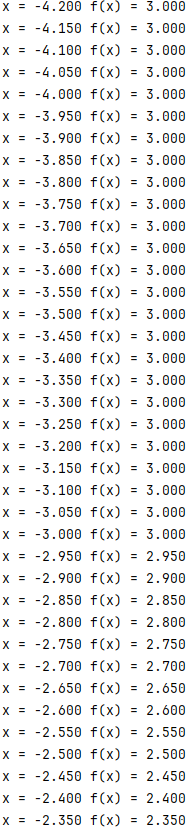
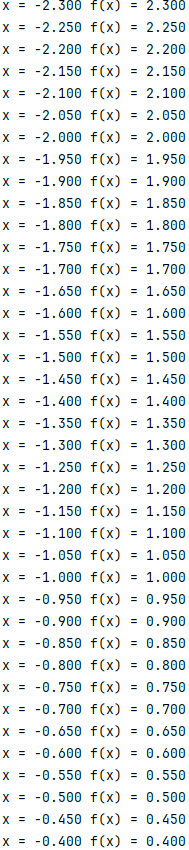
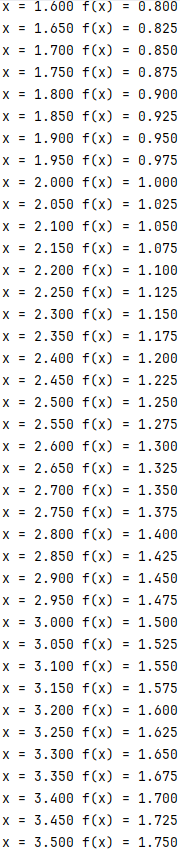
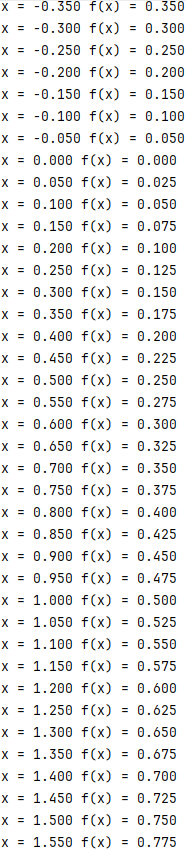
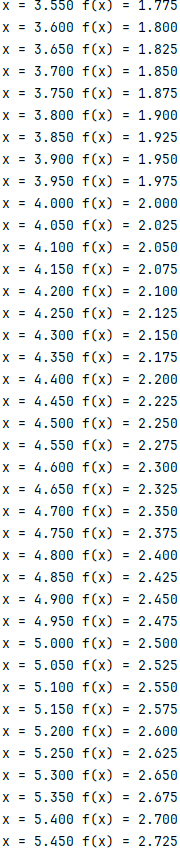
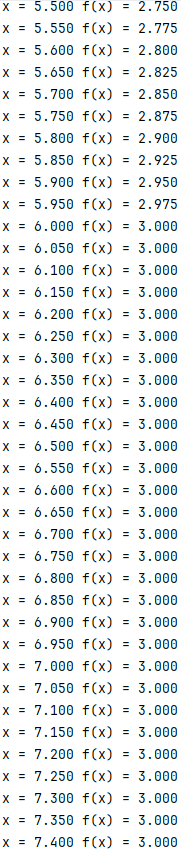
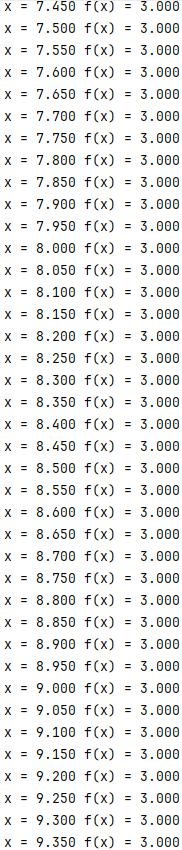
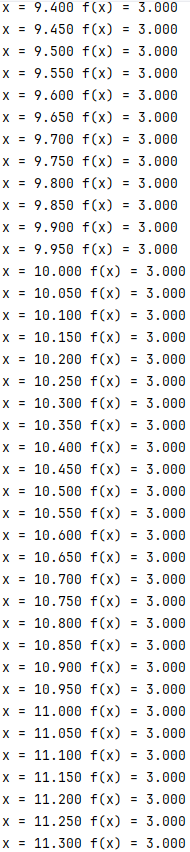
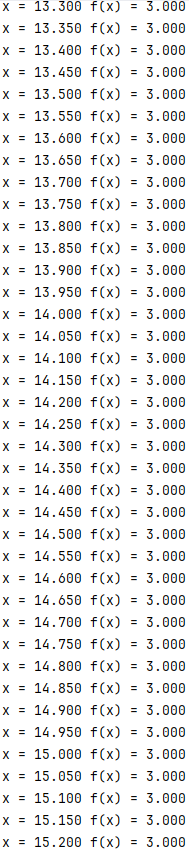
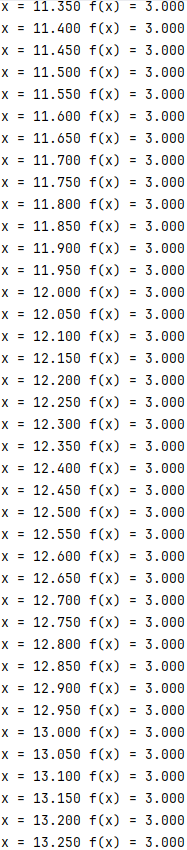
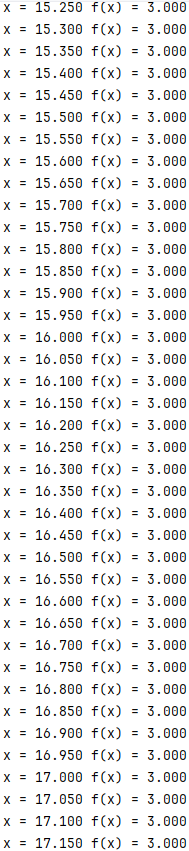
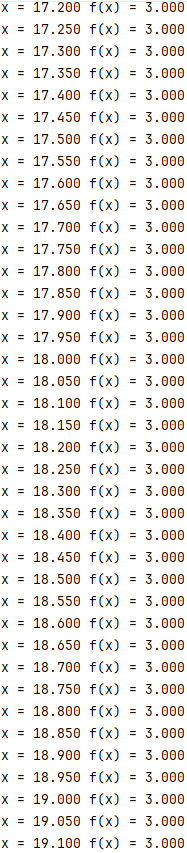
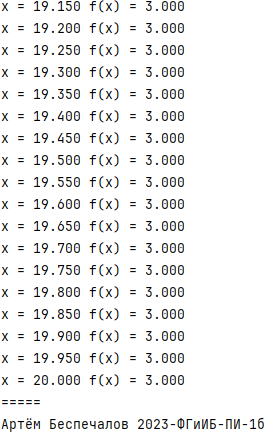
def func(x0, xn, hx):  
 kol = 0  
 hx = str(hx)  
 if '.' in hx:  
 kol = len(hx.split('.')[1])  
 for x in [i / 10 \*\* kol for i in  
 range(int(x0 \* 10 \*\* kol), int((xn + float(hx)) \* 10 \*\* kol), int(float(hx) \* 10 \*\* kol))]:  
 if x <= -3 or x >= 6:  
 print(f'x = {"{:.3f}".format(x)} f(x) = {"{:.3f}".format(3)}')  
 elif x > -3 and x < 0:  
 print(f'x = {"{:.3f}".format(x)} f(x) = {"{:.3f}".format(-1 \* x)}')  
 elif x >= 0 and x < 6:  
 print(f'x = {"{:.3f}".format(x)} f(x) = {"{:.3f}".format(x \* 0.5)}')  
  
  
func(-10, 20, 0.05)  
print('=====')  
print('Артём Беспечалов 2023-ФГиИБ-ПИ-1б')

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

**2.2.1. Используемые инструкции в программе**

|  |  |  |
| --- | --- | --- |
| **№** | **Инструкция  (из кода программы)** | **Название и назначение инструкции** |
| **1.** | сlass Function:  def \_\_init\_\_(self, x0, xn, hx):  self.x0 = x0  self.xn = xn  self.hx = str(hx) | **Инициализация класса Function. Конструктор класса. Экземпляр класса получает на вход x0, xn, hx.** |
| **2.** | def func(self): | **Метод func, который будет выполнять основной алгоритм.** |
| **3.** | kol = 0 | **Инструкция присваивания переменной kol значения 0.** |
| **4.** | if '.' in self.hx:  kol = len(self.hx.split('.')[1]) | **Инструкция ветвления. Если число hx – дробное, то выполняется инструкция присваивания переменной kol значения количества знаков после запятой в числе hx.** |
| **5.** | for x in [i / 10 \*\* kol for i in range(int(self.x0 \* 10 \*\* kol), int((self.xn + float(self.hx)) \* 10 \*\* kol), int(float(self.hx) \* 10 \*\* kol))]: | **Инструкция цикла for. Перебираем значения x в заданном диапазоне.** |
| **6.** | if x <= -3 or x >= 6:  print(f'x = {"{:.3f}".format(x)} f(x) = {"{:.3f}".format(3)}') | **Инструкция ветвления. Если условие истинно, выполняется инструкция вывода** |
| **7.** | elif x > -3 and x < 0:  print(f'x = {"{:.3f}".format(x)} f(x) = {"{:.3f}".format(-1 \* x)}') | **Инструкция ветвления. Если условие 6 не выполняется, то проверяется истинность этого условия. Если условие истинно, выполняется инструкция вывода.** |
| **8.** | elif x >= 0 and x < 6:  print(f'x = {"{:.3f}".format(x)} f(x) = {"{:.3f}".format(x \* 0.5)}') | **Инструкция ветвления. Если условие 7 не выполняется, то проверяется истинность этого условия. Если условие истинно, выполняется инструкция вывода.** |
| **9.** | graf = Function(-10, 20, 0.05) | **Создание экземпляря graf класса Fun, получающего на вход x0 = -10, xn = 20, hx = 0.05.** |
| **10.** | graf.func() | **Вызов метода func() для экземпляра graf.** |

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

*             *

**2.2.3. Текст отлаженной программы**

class Function:  
 def \_\_init\_\_(self, x0, xn, hx):  
 self.x0 = x0  
 self.xn = xn  
 self.hx = str(hx)  
  
 def func(self):  
 kol = 0  
 if '.' in self.hx:  
 kol = len(self.hx.split('.')[1])  
 for x in [i / 10 \*\* kol for i in range(int(self.x0 \* 10 \*\* kol),  
 int((self.xn + float(self.hx)) \* 10 \*\* kol),  
 int(float(self.hx) \* 10 \*\* kol))]:  
 if x <= -3 or x >= 6:  
 print(f'x = {"{:.3f}".format(x)} f(x) = {"{:.3f}".format(3)}')  
 elif x > -3 and x < 0:  
 print(f'x = {"{:.3f}".format(x)} f(x) = {"{:.3f}".format(-1 \* x)}')  
 elif x >= 0 and x < 6:  
 print(f'x = {"{:.3f}".format(x)} f(x) = {"{:.3f}".format(x \* 0.5)}')  
  
  
graf = Function(-10, 20, 0.05)  
graf.func()  
print('=====')  
print('Артём Беспечалов 2023-ФГиИБ-ПИ-1б')

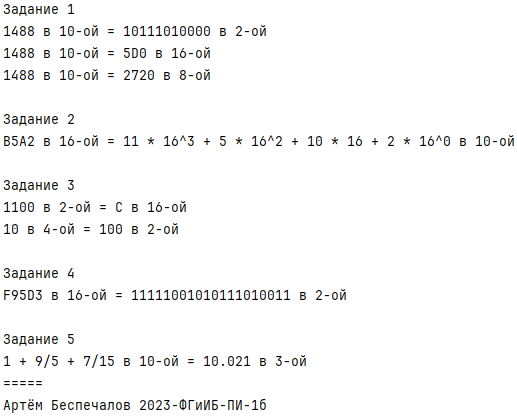
**3. ПОСТАНОВКА ЗАДАЧИ ИНДИВИДУАЛЬНОГО ЗАДАНИЯ №6 (ВАРИАНТ 21)**

**3.1. Объектно-ориентированное программирование на Python (№6)**

**3.1.1. Используемые инструкции в программе**

|  |  |  |
| --- | --- | --- |
| **№** | **Инструкция  (из кода программы)** | **Название и назначение инструкции** |
| **1.** | class Program:  def \_\_init\_\_(self):  pass | **Класс Program с конструктором класса. Объект не получает на вход ничего.** |
| **2.** | def int\_from\_n\_to\_10(self, num, sys, ans=0, abc="0123456789ABCDEF"):  num1 = num[::-1]  for i in range(len(num1)):  ans += ((sys \*\* i) \* (abc.index(num[len(num1) - 1 - i])))  return ans | **Метод для перевода целого числа из n-ой системы счисления в 10-ую. Используются инструкции присваивания, цикла.** |
| **3.** | def int\_from\_10\_to\_n(self, num, sys, ans='', abc="0123456789ABCDEF"):  if num == 0:  return 0  while num > 0:  ans += abc[num % sys]  num //= sys  ans = ans[::-1]  return ans | **Метод для перевода целого числа из 10-ой системы счисления в n-ую. Используются инструкции присваивания, цикла, ветвления.** |
| **4.** | def float\_from\_10\_to\_n(self, num, sys, precision=3):int\_part = int(num)  new\_int = ''  if int\_part == 0:  new\_int = '0'  else:  while int\_part > 0:  new\_int = str(int\_part % sys) + new\_int  int\_part //= sysfloat\_part = num - int(num)  new\_float = ''  while precision > 0 and float\_part > 0:  float\_part \*= sys  digit = int(float\_part)  new\_float += str(digit)  float\_part -= digit  precision -= 1if new\_float:  return f"{new\_int}.{new\_float}"  else:  return new\_int | **Метод для перевода дробного числа из 10-ой системы счисления в n-ую. Используются инструкции присваивания, цикла, ветвления.** |
| **4.** | def test1(self):  print(f"1488 в 10-ой = {self.int\_from\_10\_to\_n(1488, 2)} в 2-ой")  print(f"1488 в 10-ой = {self.int\_from\_10\_to\_n(1488, 16)} в 16-ой")  print(f"1488 в 10-ой = {self.int\_from\_10\_to\_n(1488, 8)} в 8-ой") | **Метод для решения задачи.** |
|  | def test2(self):  print(  f"B5A2 в 16-ой = {self.int\_from\_n\_to\_10('B', 16)} \* 16^3 + {self.int\_from\_n\_to\_10('5', 16)} \* 16^2 + {self.int\_from\_n\_to\_10('A', 16)} \* 16 + 2 \* 16^0 в 10-ой") | **Метод для решения задачи.** |
|  | def test3(self):  a = ('1100 в 2-ой', self.int\_from\_n\_to\_10('1100', 2)) b = ('10 в 4-ой', self.int\_from\_n\_to\_10('10', 4)) c = ('C в 16-ой', self.int\_from\_n\_to\_10('C', 16))d = ('100 в 2-ой', self.int\_from\_n\_to\_10('100', 2)) abc = []  abc.append(a)  abc.append(b)  abc.append(c)  abc.append(d)  for i in range(len(abc)):  for j in range(i + 1, len(abc)):  if abc[i][1] == abc[j][1]:  print(f'{abc[i][0]} = {abc[j][0]}') | **Метод для решения задачи. Используются инструкции ветвления, цикла, присваивания.** |
|  | def test4(self):  print(f"F95D3 в 16-ой = {self.int\_from\_10\_to\_n(self.int\_from\_n\_to\_10('F95D3', 16), 2)} в 2-ой") | **Метод для решения задачи.** |
|  | def test5(self):  print(f"1 + 9/5 + 7/15 в 10-ой = {self.float\_from\_10\_to\_n((1 + 9 / 5 + 7 / 15), 3)} в 3-ой") | **Метод для решения задачи.** |

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

****

**3.1.3. Текст отлаженной программы**

class Program:  
 def \_\_init\_\_(self):  
 pass  
  
 def int\_from\_n\_to\_10(self, num, sys, ans=0, abc="0123456789ABCDEF"):  
 num1 = num[::-1]  
 for i in range(len(num1)):  
 ans += ((sys \*\* i) \* (abc.index(num[len(num1) - 1 - i])))  
 return ans  
  
 def int\_from\_10\_to\_n(self, num, sys, ans='', abc="0123456789ABCDEF"):  
 if num == 0:  
 return 0  
 while num > 0:  
 ans += abc[num % sys]  
 num //= sys  
 ans = ans[::-1]  
 return ans  
  
 def float\_from\_10\_to\_n(self, num, sys, precision=3):  
 *# Переводим целую часть числа* int\_part = int(num)  
 new\_int = ''  
 if int\_part == 0:  
 new\_int = '0'  
 else:  
 while int\_part > 0:  
 new\_int = str(int\_part % sys) + new\_int  
 int\_part //= sys *# int\_part = int\_part // sys  
 # Переводим дробную часть числа* float\_part = num - int(num)  
 new\_float = ''  
 while precision > 0 and float\_part > 0:  
 float\_part \*= sys  
 digit = int(float\_part)  
 new\_float += str(digit)  
 float\_part -= digit  
 precision -= 1  
 *# Собираем полное троичное представление* if new\_float:  
 return f"{new\_int}.{new\_float}"  
 else:  
 return new\_int  
  
 def test1(self):  
 print(f"1488 в 10-ой = {self.int\_from\_10\_to\_n(1488, 2)} в 2-ой")  
 print(f"1488 в 10-ой = {self.int\_from\_10\_to\_n(1488, 16)} в 16-ой")  
 print(f"1488 в 10-ой = {self.int\_from\_10\_to\_n(1488, 8)} в 8-ой")  
  
 def test2(self):  
 print(  
 f"B5A2 в 16-ой = {self.int\_from\_n\_to\_10('B', 16)} \* 16^3 + {self.int\_from\_n\_to\_10('5', 16)} \* 16^2 + {self.int\_from\_n\_to\_10('A', 16)} \* 16 + 2 \* 16^0 в 10-ой")  
  
 def test3(self):  
 a = ('1100 в 2-ой', self.int\_from\_n\_to\_10('1100', 2)) *# 1100 из 2-ой в 10-ую* b = ('10 в 4-ой', self.int\_from\_n\_to\_10('10', 4)) *# 10 из 4-ой в 10-ую* c = ('C в 16-ой', self.int\_from\_n\_to\_10('C', 16)) *# С из 16-ой в 10-ую* d = ('100 в 2-ой', self.int\_from\_n\_to\_10('100', 2)) *# 100 из 2-ой в 10-ую* abc = []  
 abc.append(a)  
 abc.append(b)  
 abc.append(c)  
 abc.append(d)  
 for i in range(len(abc)):  
 for j in range(i + 1, len(abc)):  
 if abc[i][1] == abc[j][1]:  
 print(f'{abc[i][0]} = {abc[j][0]}')  
  
 def test4(self):  
 print(f"F95D3 в 16-ой = {self.int\_from\_10\_to\_n(self.int\_from\_n\_to\_10('F95D3', 16), 2)} в 2-ой")  
  
 def test5(self):  
 print(f"1 + 9/5 + 7/15 в 10-ой = {self.float\_from\_10\_to\_n((1 + 9 / 5 + 7 / 15), 3)} в 3-ой")  
  
  
program = Program()  
print('Задание 1')  
program.test1()  
print('\nЗадание 2')  
program.test2()  
print('\nЗадание 3')  
program.test3()  
print('\nЗадание 4')  
program.test4()  
print('\nЗадание 5')  
program.test5()  
print('=====')  
print('Артём Беспечалов 2023-ФГиИБ-ПИ-1б')

**4. ПОСТАНОВКА ЗАДАЧИ ИНДИВИДУАЛЬНОГО ЗАДАНИЯ №7 (ВАРИАНТ 21)**

**4.1. Объектно-ориентированное программирование на Python (№7)**

**4.1.1. Описание алгоритма вычисления определенного интеграла   
по методу средних прямоугольников**

Как известно, точное вычисление определенного интеграла по формуле Ньютона  Лейбница не всегда возможно, так как первообразная подынтегральной функции не всегда выражается в конечном виде через элементарные. Иногда нахождение первообразной связано с громоздкими выкладками. В таких случаях целесообразно определенные интегралы вычислять приближенно. Существуют различные методы приближенного вычисления определенного интеграла (как говорят, численного интегрирования), наиболее распространенными из которых являются: метод прямоугольников (левых, средних, правых), метод трапеций и метод парабол (Симпсона).

Рассмотрим первый из них – метод средних прямоугольников. Решение данной задачи проанализируем сначала в самом общем виде.

Пусть требуется определить значение определенного интеграла  от функции  с заданными пределами интегрирования *a* и *b* и с заданной точностью *eps*. Исходим из того, что функция  непрерывна и неотрицательна на отрезке [a,b]. Тогда в общем случае график функции  можно представить в виде рис.1. Найти значение определенного интеграла – это значит найти площадь некоторой криволинейной фигуры *S* (см. заштрихованную область на рис.1). Площадь криволинейной фигуры, ограниченная графиком функции , отрезком [a,b] оси O*x* и соответствующими отрезками прямых: *x=a* и *x=b*, называемых пределами интегрирования, равна: .

Данный алгоритм основан на вычислении площадей прямоугольников, на которые разбивается область интегрирования *S*. В общем виде алгоритм выглядит следующим образом. Отрезок интегрирования  разбивается на *n* равных частей точками . Длина каждой части .











Рис.1

Обозначим . значения функции  в точках , т. е.

  …,  

Произведение  представляет собой площадь *i*-го прямоугольника с основанием  и высотой . Площадь всей криволинейной трапеции *S* приближенно равна площади ступенчатой фигуры, состоящей из прямоугольников. Поэтому получившееся выражение:



или

.

называется ***формулой прямоугольников***, на которой и основан методсредних прямоугольников при вычислении определенного интеграла некоторой реляционной функции.

Однако описанный выше общий алгоритм вычисления определенного интеграла не дает ответа на вопрос, как достигается заданная точность *eps* при его вычислении. Для этого рассмотрим подробнее, каким образом разбивается область интегрирования *S* на прямоугольники и последовательность увеличения числа прямоугольных площадей, т.к. именно от количества прямоугольников зависит достигаемая точность вычисления определенного интеграла.

Для вычисления первого приближенного значения интеграла разделим отрезок интегрирования  на *n=4* равных частей () и определим значение функции , соответственно, для четырех точек . При этом значения точек  рассчитываются исходя из выражений:

, где: .

Расчет точек *, i=1,…4* согласно этим выражениям обеспечивает расположение ординаты функции  в точке пересечения с серединой прямоугольника по ширине в самой верхней его части (рис. 2), рассчитанной для каждой из четырех точек *, i=1,…4*:





Рис.2































Найдя , вычисляется площадь каждого из полученных прямоугольников на первом шаге приближенного вычисления интеграла (*n=4*):

, где *i=1,2…,n=4*

Сумма всех четырех площадей этих прямоугольников – есть первое приближение значения интеграла:

, где *i=1,2,…,n=4*.

Однако одно приближение не позволяет оценить точность, с которой вычислено значение определенного интеграла – необходимо найти второе приближение значения определенного интеграла. Для этого увеличим значение *n*, описывающее количество равных участков, на которое разбивался отрезок интегрирования , в два раза, т.е. *n=8*. Увеличение количества прямоугольников повышает точность описания площади криволинейной трапеции, что наглядно подтверждает рис.3.

Аналогично сумме площадей *S1* находим сумму восьми срединных прямоугольников (рис.3):

, где *i=1,2,…,n=8*.





Рис.3



















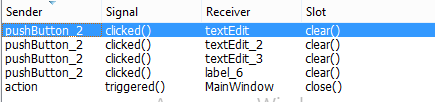
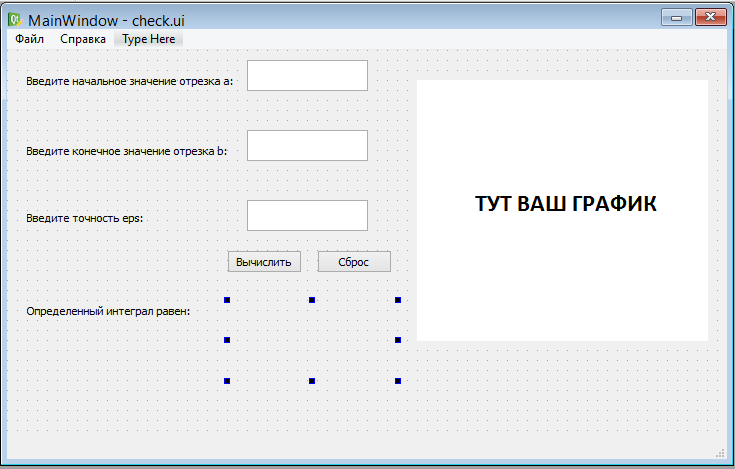
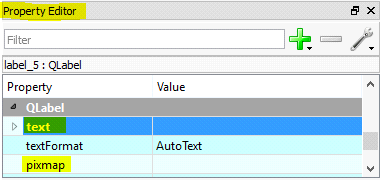
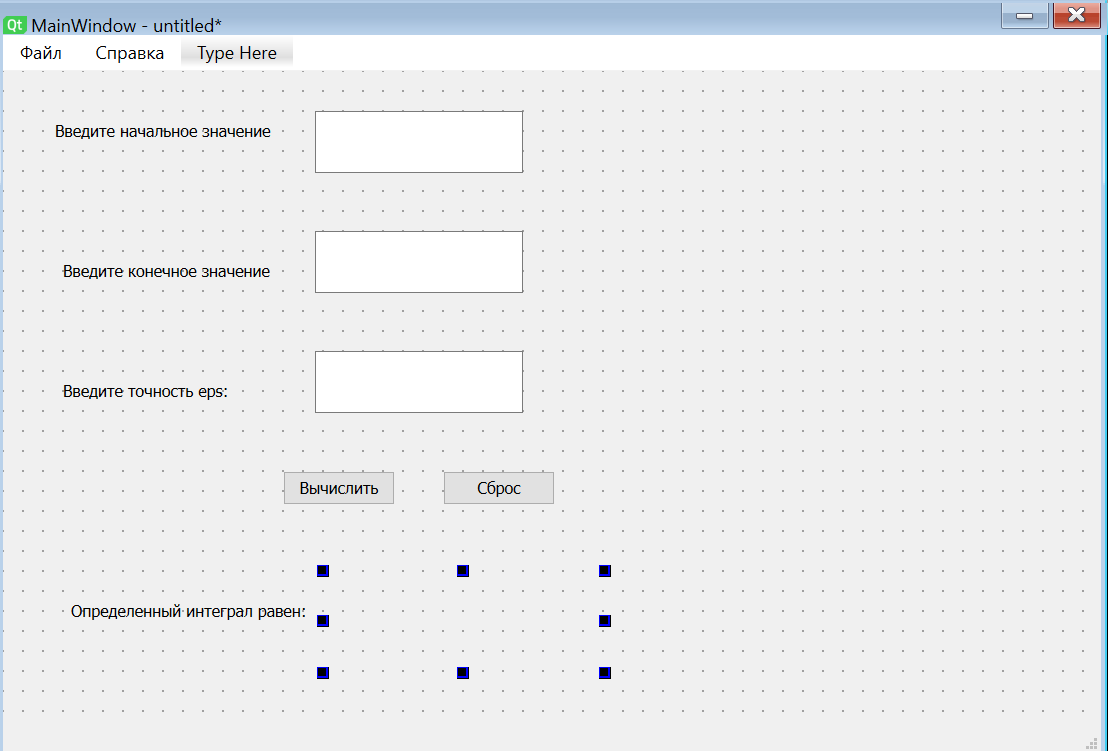
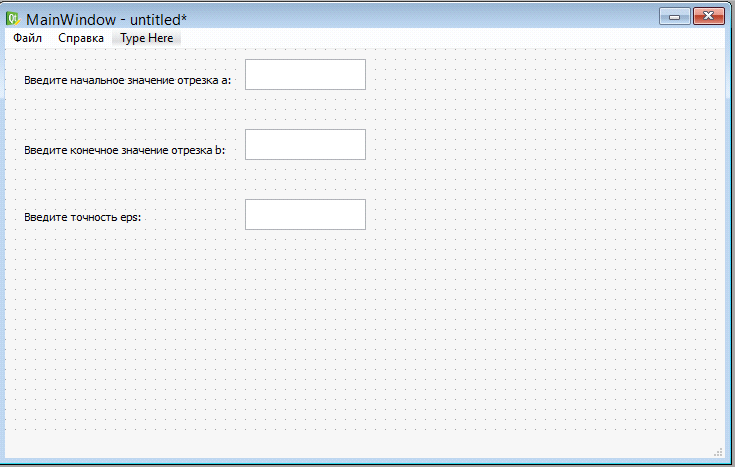
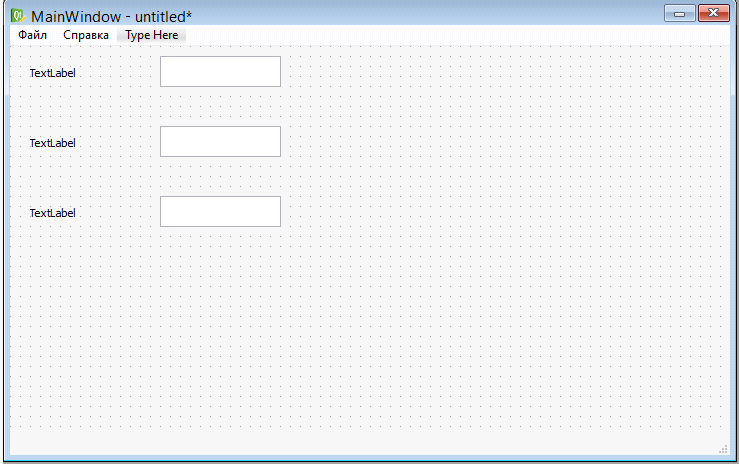
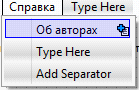
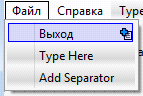
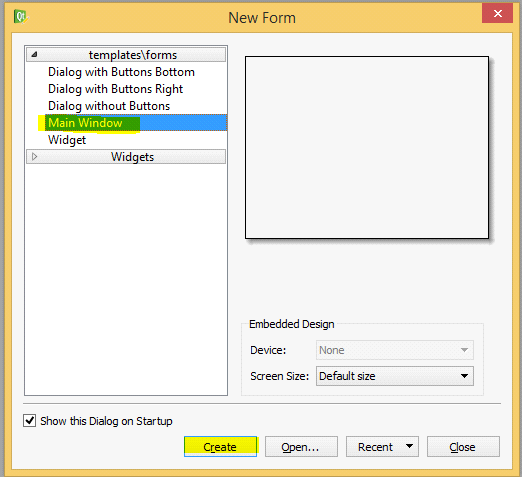


Зная значение первого и второго приближения значений определенного интеграла, соответственно, *S1* и *S2*, несложно оценить заданную точность *eps*. Для этого достаточно, чтобы выполнялось условие:



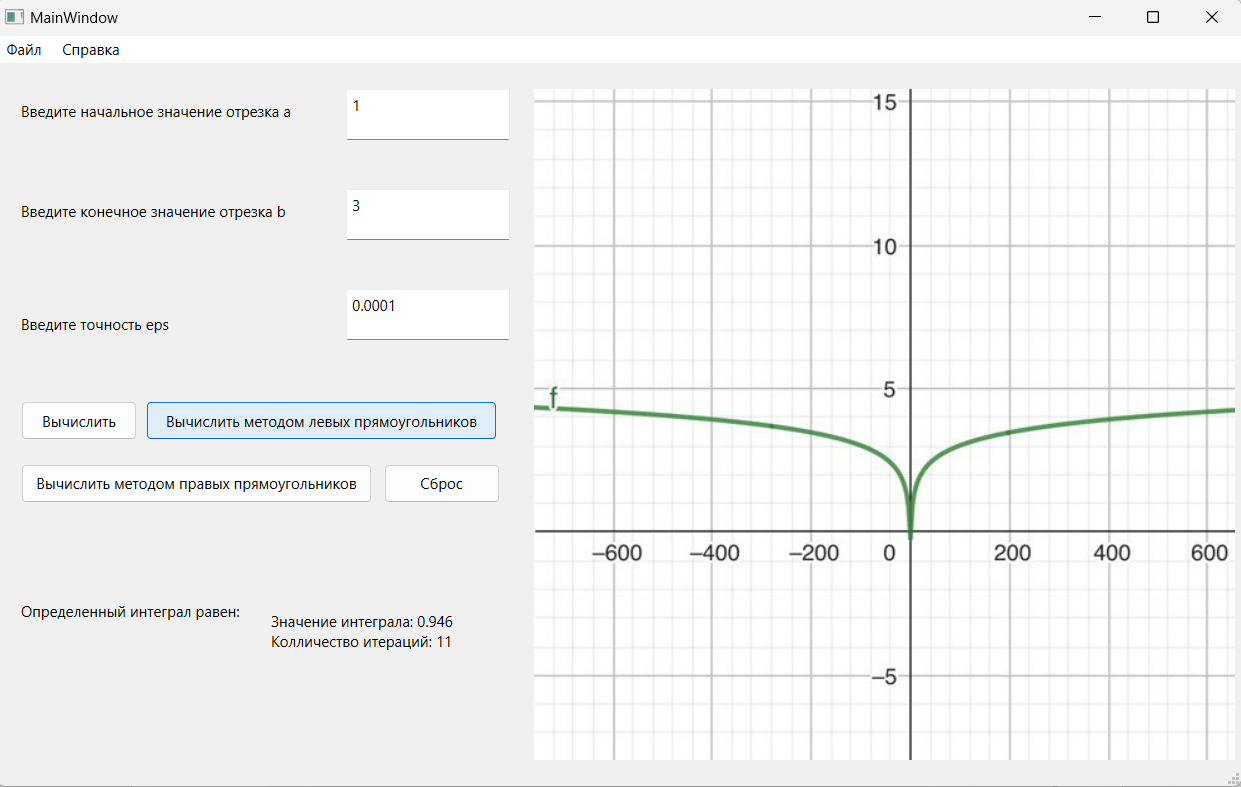
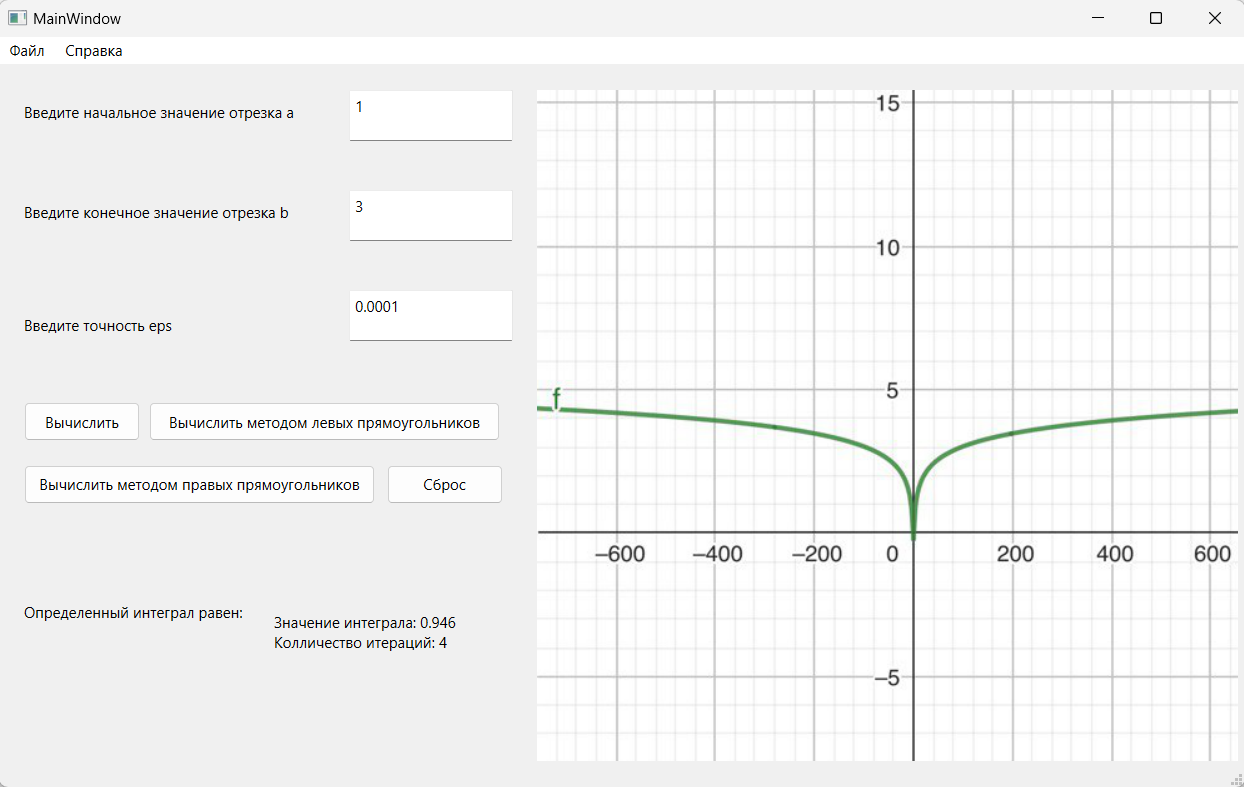
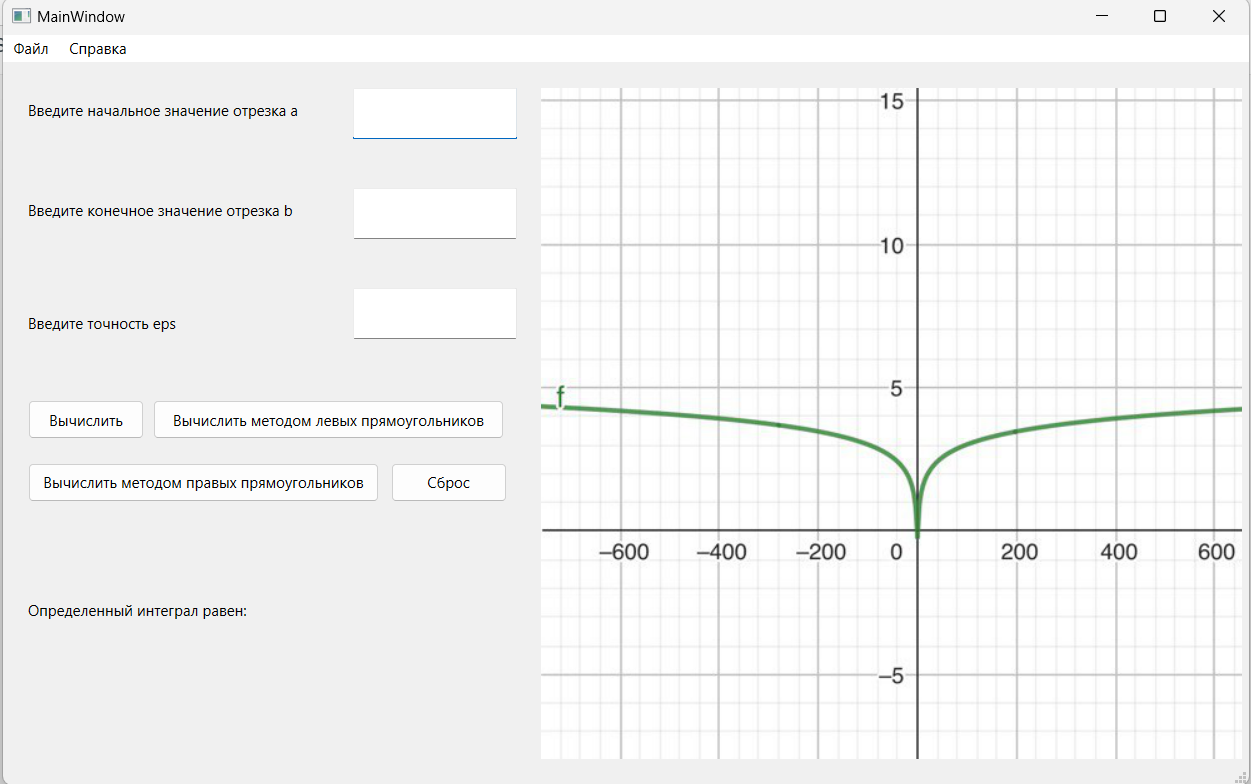
Если условие выполняется, то значение *S2* принимается за искомое значение интеграла. Если условие не выполняется, то искомое значение *S2* принимается за предыдущее значение интеграла, т.е. *S1=S2*. Число точек деления отрезка интегрирования  снова увеличиваем в два раза (*n=16*) и вычисляется новое значение *S2*. Такой процесс удвоения точек деления и вычисления *S2* будем продолжать до тех пор, пока модуль разности  не станет меньше заданной точности *eps*.

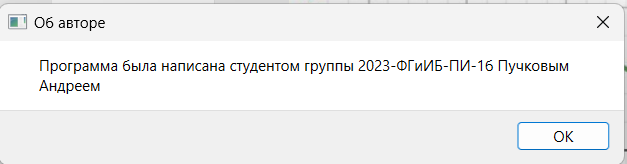
**3.2.2. Разработка пользовательского интерфейса программы с использованием виджетов PyQt**



**3.2.3. Используемые библиотеки в программе**

|  |  |
| --- | --- |
| **Название библиотеки** | **Назначение библиотеки** |
| sys | Предоставляет доступ к некоторым переменным и функциям, взаимодействующим со структурой Python. |
| os | Модуль os предоставляет функции для работы с операционной системой, такие как управление файлами и директориями. |
| PyQt6 | Набор Python-пакетов для создания графического интерфейса пользователя (GUI) на основе библиотеки Qt от Digia. Импортированные подмодули QtCore, QtGui, QtWidgets, QMessageBox предоставляют классы и методы для работы с элементами интерфейса и диалоговыми окнами. |
| QMainWindow | Функциональность элеметов GUI - окно |
| QLabel | Метка |
| QGridLayout | Сетка |
| QWidget | Виджет |
| QPushButton | Кнопка |
| QMessageBox | Диалоговое окно |
| math | Модуль math предоставляет математические функции, такие как sin(), cos(), sqrt() и другие. |

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

**

**3.2.5. Текст отлаженной программы**

**main.py**

import sys  
import os  
from check1 import \*  
from PyQt6 import QtCore, QtGui, QtWidgets  
from PyQt6.QtWidgets import QMainWindow, QLabel, QGridLayout, QWidget, QPushButton, QMessageBox  
from PyQt6.QtCore import QSize  
from math import \*  
  
  
class MyWin(QtWidgets.QMainWindow):  
 def \_\_init\_\_(self, parent=None):  
 QtWidgets.QWidget.\_\_init\_\_(self, parent)  
 self.ui = Ui\_MainWindow()  
 self.ui.setupUi(self)  
 self.ui.pushButton.clicked.connect(self.Integral) # функция кнопки вычислить  
 self.ui.action\_2.triggered.connect(self.Info) # Beшaем на "Об aвтоpe" функцию Inf  
 self.ui.pushButton\_3.clicked.connect(self.left\_Integral)  
 self.ui.pushButton\_4.clicked.connect(self.right\_Integral)  
  
 def Info(self):  
 QMessageBox.about(self, "Об авторe",  
 "Программа была написана студентом группы 2023-ФГиИБ-ПИ-1б Пучковым Андреем")  
  
 def Integral(self):  
 self.ui.label\_5.setText(" ")  
 s1 = 1  
 s2 = 0  
 lter = 0 # Счетчик итераций  
 a = int(self.ui.textEdit.toPlainText())  
 b = int(self.ui.textEdit\_2.toPlainText()) # "Считываем" нижнюю и верхнюю границы  
 n = 4 # Число площадок (разбиений)  
 i = 0  
 eps = float(self.ui.textEdit\_3.toPlainText()) # Считываем точность  
 while abs(s2 - s1) >= eps:  
 s1 = s2  
 s2 = 0  
 n = n \* 2  
 h = (b - a) / n  
 lter += 1  
 i = 0  
 while i < n:  
 x = a + h \* (i + 0.5)  
 k = sin(1 / 3) \* log(0.5 + x \*\* 2) # ТУТ ДОЛЖНА БЫТЬ ВАША ФУНКЦИЯ!!!!  
 s2 += k \* h  
 i += 1  
 Otvet = "\nЗначение интеграла: " + str("%.3f" % (s2)) + "\nКолличество итераций: " + str(lter)  
 self.ui.label\_5.setText(Otvet) # Выводим значение интеграла.  
  
 def left\_Integral(self):  
 self.ui.label\_5.setText(" ")  
 s1 = 1  
 s2 = 0  
 lter = 0 # Счетчик итераций  
 a = int(self.ui.textEdit.toPlainText())  
 b = int(self.ui.textEdit\_2.toPlainText()) # "Считываем" нижнюю и верхнюю границы  
 n = 4 # Число площадок (разбиений)  
 i = 0  
 eps = float(self.ui.textEdit\_3.toPlainText()) # Считываем точность  
 while abs(s2 - s1) >= eps:  
 s1 = s2  
 s2 = 0  
 n = n \* 2  
 h = (b - a) / n  
 lter += 1  
 i = 0  
 while i < n:  
 x = a + h \* i  
 k = sin(1 / 3) \* log(0.5 + x \*\* 2) # ТУТ ДОЛЖНА БЫТЬ ВАША ФУНКЦИЯ!!!!  
 s2 += k \* h  
 i += 1  
 Otvet = "\nЗначение интеграла: " + str("%.3f" % (s2)) + "\nКолличество итераций: " + str(lter)  
 self.ui.label\_5.setText(Otvet) # Выводим значение интеграла.  
  
 def right\_Integral(self):  
 self.ui.label\_5.setText(" ")  
 s1 = 1  
 s2 = 0  
 lter = 0 # Счетчик итераций  
 a = int(self.ui.textEdit.toPlainText())  
 b = int(self.ui.textEdit\_2.toPlainText()) # "Считываем" нижнюю и верхнюю границы  
 n = 4 # Число площадок (разбиений)  
 i = 0  
 eps = float(self.ui.textEdit\_3.toPlainText()) # Считываем точность  
 while abs(s2 - s1) >= eps:  
 s1 = s2  
 s2 = 0  
 n = n \* 2  
 h = (b - a) / n  
 lter += 1  
 i = 0  
 while i < n:  
 x = a + h \* (i + 1)  
 k = sin(1 / 3) \* log(0.5 + x \*\* 2) # ТУТ ДОЛЖНА БЫТЬ ВАША ФУНКЦИЯ!!!!  
 s2 += k \* h  
 i += 1  
 Otvet = "\nЗначение интеграла: " + str("%.3f" % (s2)) + "\nКолличество итераций: " + str(lter)  
 self.ui.label\_5.setText(Otvet) # Выводим значение интеграла.  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 app = QtWidgets.QApplication(sys.argv) # ОБЯЗАТЕЛЬНО К НАПИСАНИЮ, ДЛЯ КОРРЕКТНОЙ РАБОТЫ ПРОГРАММЫ!!!  
 myapp = MyWin()  
 myapp.show()  
 sys.exit(app.exec())

**check.py**

# Form implementation generated from reading ui file 'check.ui'  
#  
# Created by: PyQt6 UI code generator 6.6.1  
#  
# WARNING: Any manual changes made to this file will be lost when pyuic6 is  
# run again. Do not edit this file unless you know what you are doing.  
  
  
from PyQt6 import QtCore, QtGui, QtWidgets  
  
  
class Ui\_MainWindow(object):  
 def setupUi(self, MainWindow):  
 MainWindow.setObjectName("MainWindow")  
 MainWindow.resize(997, 600)  
 self.centralwidget = QtWidgets.QWidget(parent=MainWindow)  
 self.centralwidget.setObjectName("centralwidget")  
 self.textEdit = QtWidgets.QTextEdit(parent=self.centralwidget)  
 self.textEdit.setGeometry(QtCore.QRect(280, 20, 131, 41))  
 self.textEdit.setObjectName("textEdit")  
 self.label = QtWidgets.QLabel(parent=self.centralwidget)  
 self.label.setGeometry(QtCore.QRect(20, 30, 241, 16))  
 self.label.setObjectName("label")  
 self.textEdit\_2 = QtWidgets.QTextEdit(parent=self.centralwidget)  
 self.textEdit\_2.setGeometry(QtCore.QRect(280, 100, 131, 41))  
 self.textEdit\_2.setObjectName("textEdit\_2")  
 self.label\_2 = QtWidgets.QLabel(parent=self.centralwidget)  
 self.label\_2.setGeometry(QtCore.QRect(20, 110, 241, 16))  
 self.label\_2.setObjectName("label\_2")  
 self.textEdit\_3 = QtWidgets.QTextEdit(parent=self.centralwidget)  
 self.textEdit\_3.setGeometry(QtCore.QRect(280, 180, 131, 41))  
 self.textEdit\_3.setObjectName("textEdit\_3")  
 self.label\_3 = QtWidgets.QLabel(parent=self.centralwidget)  
 self.label\_3.setGeometry(QtCore.QRect(20, 200, 141, 16))  
 self.label\_3.setObjectName("label\_3")  
 self.pushButton = QtWidgets.QPushButton(parent=self.centralwidget)  
 self.pushButton.setGeometry(QtCore.QRect(20, 270, 93, 28))  
 self.pushButton.setObjectName("pushButton")  
 self.pushButton\_2 = QtWidgets.QPushButton(parent=self.centralwidget)  
 self.pushButton\_2.setGeometry(QtCore.QRect(230, 270, 93, 28))  
 self.pushButton\_2.setObjectName("pushButton\_2")  
 self.label\_4 = QtWidgets.QLabel(parent=self.centralwidget)  
 self.label\_4.setGeometry(QtCore.QRect(20, 340, 191, 16))  
 self.label\_4.setObjectName("label\_4")  
 self.label\_5 = QtWidgets.QLabel(parent=self.centralwidget)  
 self.label\_5.setGeometry(QtCore.QRect(220, 330, 191, 111))  
 self.label\_5.setText("")  
 self.label\_5.setObjectName("label\_5")  
 self.label\_6 = QtWidgets.QLabel(parent=self.centralwidget)  
 self.label\_6.setGeometry(QtCore.QRect(470, 20, 511, 371))  
 self.label\_6.setText("")  
 self.label\_6.setPixmap(QtGui.QPixmap("Graphics/graphic.png"))  
 self.label\_6.setObjectName("label\_6")  
 MainWindow.setCentralWidget(self.centralwidget)  
 self.menubar = QtWidgets.QMenuBar(parent=MainWindow)  
 self.menubar.setGeometry(QtCore.QRect(0, 0, 997, 26))  
 self.menubar.setObjectName("menubar")  
 self.menu = QtWidgets.QMenu(parent=self.menubar)  
 self.menu.setObjectName("menu")  
 self.menu\_2 = QtWidgets.QMenu(parent=self.menubar)  
 self.menu\_2.setObjectName("menu\_2")  
 MainWindow.setMenuBar(self.menubar)  
 self.statusbar = QtWidgets.QStatusBar(parent=MainWindow)  
 self.statusbar.setObjectName("statusbar")  
 MainWindow.setStatusBar(self.statusbar)  
 self.action = QtGui.QAction(parent=MainWindow)  
 self.action.setObjectName("action")  
 self.action\_2 = QtGui.QAction(parent=MainWindow)  
 self.action\_2.setObjectName("action\_2")  
 self.menu.addAction(self.action)  
 self.menu\_2.addAction(self.action\_2)  
 self.menubar.addAction(self.menu.menuAction())  
 self.menubar.addAction(self.menu\_2.menuAction())  
  
 self.retranslateUi(MainWindow)  
 self.action.triggered.connect(MainWindow.close) # type: ignore  
 self.pushButton\_2.clicked.connect(self.textEdit.clear) # type: ignore  
 self.pushButton\_2.clicked.connect(self.textEdit\_2.clear) # type: ignore  
 self.pushButton\_2.clicked.connect(self.textEdit\_3.clear) # type: ignore  
 self.pushButton\_2.clicked.connect(self.label\_6.clear) # type: ignore  
 QtCore.QMetaObject.connectSlotsByName(MainWindow)  
  
 def retranslateUi(self, MainWindow):  
 \_translate = QtCore.QCoreApplication.translate  
 MainWindow.setWindowTitle(\_translate("MainWindow", "MainWindow"))  
 self.label.setText(\_translate("MainWindow", "Введите начальное значение отрезка a"))  
 self.label\_2.setText(\_translate("MainWindow", "Введите конечное значение отрезка b"))  
 self.label\_3.setText(\_translate("MainWindow", "Введите точность eps"))  
 self.pushButton.setText(\_translate("MainWindow", "Вычислить"))  
 self.pushButton\_2.setText(\_translate("MainWindow", "Сброс"))  
 self.label\_4.setText(\_translate("MainWindow", "Определенный интеграл равен:"))  
 self.menu.setTitle(\_translate("MainWindow", "Файл"))  
 self.menu\_2.setTitle(\_translate("MainWindow", "Справка"))  
 self.action.setText(\_translate("MainWindow", "Выход"))  
 self.action\_2.setText(\_translate("MainWindow", "Об авторах"))

**ВЫВОДЫ**

1. Мной были выполнены практические работы. В них я использовал императивное, функциональное, объектно-ориентированное программирование. Я работал над матрицей, табулированием функции и пользовательским интерфейсом.

2. В ООП экземпляр класса не получал входные атрибуты, потому что неправильно прописал конструктор класса, вследствие чего программа выдала ошибку.

3. Каждый вид хорош по-своему. Но ООП нужен во многих сферах программирования. К тому же, в нем много деталей и вариаций реализации, из-за чего он интереснее.

**СПИСОК ИСПОЛЬЗОВАННЫХ ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ**

1.

2.

3.