**Билет 2**

Выполнил: Пищулин Игорь

Студент П1-17

25.06.2020

1. Ссылки в языках программирования [Приложение 1] (20 минут)

2. Разработать приложение для построения графика полинома n-й степени.

Коэффициенты полинома считываются из StringGrid или аналога.

[Приложение 2] (3 часа)

**Приложение 1**

**Ссылки в языках программирования**

Ссылка – это определенный объект, который указывает на определенные данные, но не хранит их. По сути это другое имя объекта. Это переменная, которая хранит ссылку на другую переменную. Операции над ссылками: копирование и разыменовыывание.

**Пример работы с ссылками в C++.**

Ссылки в C++ обязательно связаны с каким-либо объектом. Разыменовывание не требует дополнительных опреторов.

В данном примере ссылка помогает избежать копирования, присваивания и прочих операторов.

void sum(int x, int y, int &result)

{

result = x + y;

}

int main()

{

int res = 0;

sum(2, 10, res);

cout << "res = " << res << endl; // res = 12

return 0;

}

**Приложение 2**

Разработать приложение для построения графика полинома n-й степени.

Коэффициенты полинома считываются из StringGrid или аналога.

**Листинг 1. Главный модуль**

import matplotlib.pyplot as plt

from polynom import \*

from StringGrid import \*

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

root = None

interval = None

step = None

# получаем степень полинома

power = int(input('Polynomial degree = '))

# объект StringGrid

stringgrid = StringGrid()

# настройки таблицы

stringgrid.SetColCount(power)

stringgrid.SetRowCount(1)

# вносим данные

stringgrid.Add()

print(stringgrid.Cells)

# пока не получим верные данные запрашиваем их

while not root:

try:

# получаем корни уравнения

root = list(map(float, (\*stringgrid.Cells)))

except ValueError:

print("Incorrect value.")

root = None

stringgrid.Add()

print(stringgrid.Cells)

# пока не получим верные данные запрашиваем их

while not interval:

try:

# получаем интервал графика

interval = list(map(int, (input('Enter interval(example: -4 4): ')).split()))

except ValueError:

print("Incorrect value.")

interval = None

# пока не получим верные данные запрашиваем их

while not step:

try:

# получаем шаг

step = float(input('Enter step: '))

except ValueError:

print("Incorrect value.")

step = None

pol = Polynom(root) # создаем объект полинома

# устанавливаем заголовки окна графика

plt.title(f"График полинома y(x) = {pol}")

plt.xlabel("x") # ось абсцисс

plt.ylabel("y") # ось ординат

# plt.grid() # включение отображение сетки

# получаем списки координат для отрисовки

[x, y] = pol.draw(interval, step)

# выводим четность и нечетность полинома

plt.text (x[round(len(x) / 4) \* 3], y[-1], f"Odd: {pol.odd()}")

plt.text (x[round(len(x) / 4) \* 3], y[-1] - (((max(y) - min(y)) / 100) \* 7), f"Even: {pol.even()}")

# создаем график

plt.plot(x, y, label=f"y(x) = {pol}")

# создаем легенду

plt.legend(loc="upper left")

# отрисовка

plt.show()

**Листинг 2. Класс Polynom**

import numpy

# класс полинома

class Polynom:

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

self.root = root # корни уравнения

# строковое представление объекта

def \_\_repr\_\_(self):

l = len(self.root)

stra = []

for r in self.root:

l = l - 1

if r:

stra.append(''.join([str(r), f'x^{l}' if l else '']))

return ' + '.join(stra)

def f(self, x):

"""

получение f(x) для любого x

Args:

self: Объект класса.

x: Любой x.

Returns:

res: f(x).

Raises:

"""

l = len(self.root)

res = 0

for r in self.root:

l = l - 1

res += r \* pow(x, l)

return res

def even(self):

"""

четная ли функция

Args:

self: Объект класса.

Returns:

{bool}: Да или нет.

Raises:

"""

test = [i for i in range(1, 10)]

res = [self.f(i) == self.f(-i) for i in test]

if False in res:

return False

return True

def odd(self):

"""

четность выражения

Args:

self: Объект класса.

Returns:

{bool}: Да или нет.

Raises:

"""

test = [i for i in range(1, 10)]

res = [self.f(i) == -(self.f(-i)) for i in test]

if False in res:

return False

return True

def draw(self, interval, step):

"""

Нечетность выражения

Args:

self: Объект класса.

interval: интервал координат

step: шаг

Returns:

[x, y]: Список координат.

Raises:

"""

[intervalL, intervalR] = interval

y = []

yR = []

x = [i for i in numpy.arange(intervalL, intervalR + step, step)]

y = [self.f(i) for i in x]

return [x, y]

**Листинг 2. Класс StringGrid**

import msvcrt, sys

class StringGrid:

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

self.ColCount = 5

self.RowCount = 5

self.Row = ['' for i in range(self.ColCount)]

self.Cells = [self.Row for i in range(self.RowCount)]

def SetColCount(self, n):

"""

Установка количества столбцов

Args:

self: Объект класса.

n: количество

Returns:

Raises:

"""

self.ColCount = n

self.Row = ['' for i in range(self.ColCount)]

self.Cells = [self.Row for i in range(self.RowCount)]

def SetRowCount(self, n):

"""

Установка количества строк

Args:

self: Объект класса.

n: количество

Returns:

Raises:

"""

self.RowCount = n

self.Row = ['' for i in range(self.ColCount)]

self.Cells = [self.Row for i in range(self.RowCount)]

def Add(self):

"""

Добавление значений в таблицу

Args:

self: Объект класса.

Returns:

Raises:

"""

i = 0

j = 0

num = [str(i) for i in range(0, 10)]

arrow = False

c = ''

add = True

edit = False

print(f"StringGrid.Cells[0][0] = ",end="")

print('\r', end='')

while add:

# отслеживание нажатий клавиш

pressedKey = msvcrt.getch()

if pressedKey == b'\x1b':

add = False

print()

if c != '':

self.Cells[i][j] = c

c = ''

if pressedKey == b'\r':

if c != '':

self.Cells[i][j] = c

c = ''

elif pressedKey == b'\xe0':

arrow = True

if c != '':

self.Cells[i][j] = c

c = ''

elif pressedKey == b'-':

if c == '':

c += '-'

print(f"StringGrid.Cells[{i}][{j}] = {''.join([c, ''.join([' ' for i in range(20)])])}",end="")

print('\r', end='')

elif pressedKey == b'.' or pressedKey == b',':

if not '.' in c and c != '':

c += '.'

print(f"StringGrid.Cells[{i}][{j}] = {''.join([c, ''.join([' ' for i in range(20)])])}",end="")

print('\r', end='')

elif arrow and pressedKey == b'M':

if j < self.ColCount - 1:

j += 1

print(f"StringGrid.Cells[{i}][{j}] = {''.join([self.Cells[i][j], ''.join([' ' for i in range(20)])]) if self.Cells[i][j] != '' else ''.join([' ' for i in range(20)])}",end="")

print('\r', end='')

arrow = False

elif arrow and pressedKey == b'K':

if j > 0:

j -= 1

print(f"StringGrid.Cells[{i}][{j}] = {''.join([self.Cells[i][j], ''.join([' ' for i in range(20)])]) if self.Cells[i][j] != '' else ''.join([' ' for i in range(20)])}",end="")

print('\r', end='')

arrow = False

elif arrow and pressedKey == b'H':

if i > 0:

i -= 1

print(f"StringGrid.Cells[{i}][{j}] = {''.join([self.Cells[i][j], ''.join([' ' for i in range(20)])]) if self.Cells[i][j] != '' else ''.join([' ' for i in range(20)])}",end="")

print('\r', end='')

arrow = False

elif arrow and pressedKey == b'P':

if i < self.RowCount - 1:

i += 1

print(f"StringGrid.Cells[{i}][{j}] = {''.join([self.Cells[i][j], ''.join([' ' for i in range(20)])]) if self.Cells[i][j] != '' else ''.join([' ' for i in range(20)])}",end="")

print('\r', end='')

arrow = False

elif pressedKey == b'\x08':

c = c[:-1]

print(f"StringGrid.Cells[{i}][{j}] = {''.join([c, ''.join([' ' for i in range(20)])])}",end="")

print('\r', end='')

elif not arrow and pressedKey.decode() in num:

c += pressedKey.decode()

print(f"StringGrid.Cells[{i}][{j}] = {c}",end="")

print('\r', end='')

**Запуск программы.**

Требуется ввести максимальную степень уравнения, занести коэффициенты в таблицу, ввести интервал отрисовки графика и шаг

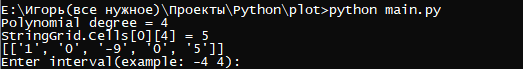


Рисунок 1. Запуск программы

Будет предложено ввести данные в таблицу. По таблице необходимо двигаться стрелочками на клавиатуре, можно вводить отрицательные и вещественные числа. Переход на другую ячейку сохраняет предыдущую, также нажатие на Enter сохраняет значение. Esc – сохранить значение и выйти из редактирования таблицы.

Запускается окно с отрисованным графиком полинома любой степени

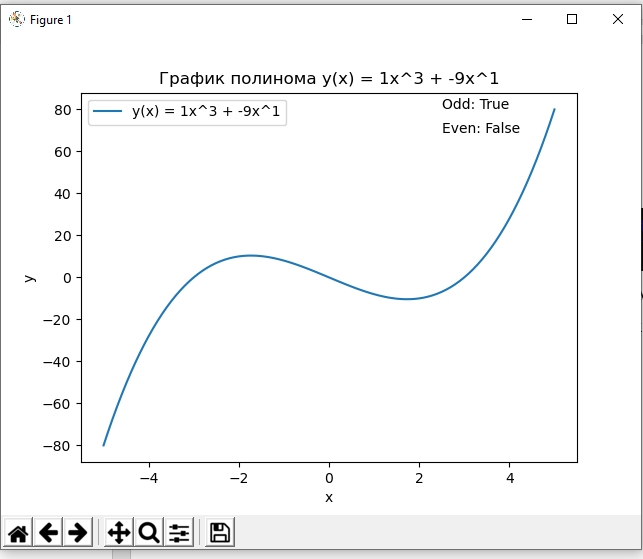


Рисунок 2. Отрисовка графика функции x^3 - 9x