Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

Факультет «Информатика и системы управления»
Кафедра «Системы обработки информации и управления»

Кузнецов Егор ИУ5-32Б

Парадигмы и конструкции языков программирования

ОТЧЁТ ПО
Домашнему заданию

Москва
2023

Задание

1. Выберите язык программирования (который Вы ранее не изучали) и (1) напишите по нему реферат с примерами кода или (2) реализуйте на нем небольшой проект (с детальным текстовым описанием).
2. Реферат (проект) может быть посвящен отдельному аспекту (аспектам) языка или содержать решение какой-либо задачи на этом языке.
3. Необходимо установить на свой компьютер компилятор (интерпретатор, транспилятор) этого языка и произвольную среду разработки.
4. В случае написания реферата необходимо разработать и откомпилировать примеры кода (или модифицировать стандартные примеры).
5. В случае создания проекта необходимо детально комментировать код.
6. При написании реферата (создании проекта) необходимо изучить и корректно использовать особенности парадигмы языка и основных конструкций данного языка.
7. Приветствуется написание черновика статьи по результатам выполнения ДЗ. Черновик статьи может быть подготовлен группой студентов, которые исследовали один и тот же аспект в нескольких языках или решили одинаковую задачу на нескольких языках.

Задача

Задачей является написание простейшей игры змейки на языке Python.

Структура программы

DZ.py – Основной файл, в котором выполнена вся работа.

Текст программы

```
from tkinter import *
from enum import Enum
import time
import random


class python_snake:
    def __init__(self, window, canv_x, canv_y, canv_width, canv_height):
        self.__started = 1
        self.__spped = 10
        self.__window = window
        self.__canv_x = canv_x
        self.__canv_y = canv_y
        self.canv_width = canv_width
        self.canv_height = canv_height
        self.__snake_x = self.canv_width // 2  # Координата старта змеи
        self.__snake_y = self.canv_height // 2  # Координата старта змеи
```

```
        self.canv = Canvas(self.__window, width=self.canv_width,
                           height=self.canv_height,
                           bg=self.CONST.CANVAS_BGCOLOR.value)
        self.canv.place(x=self.__canv_x, y=self.__canv_y)
        self.create_head_food()

        self.__window.bind('<d>', self.right)
        self.__window.bind('<D>', self.right)
        self.__window.bind('<Right>', self.right)
        self.__window.bind('<s>', self.down)
        self.__window.bind('<S>', self.down)
        self.__window.bind('<Down>', self.down)
        self.__window.bind('<a>', self.left)
        self.__window.bind('<A>', self.left)
        self.__window.bind('<Left>', self.left)
        self.__window.bind('<w>', self.up)
        self.__window.bind('<W>', self.up)
        self.__window.bind('<Up>', self.up)

        self.__window.bind('<e>', self.move)
        self.__window.bind('<q>', self.quit)
        self.__window.bind('<Destroy>', self.quit)
        self.__window.bind('<plus>', self.speed_key)
        self.__window.bind('<minus>', self.speed_key)
        self.__window.bind('<KP_Add>', self.speed_key)  # Клавиша + на
боковой клаве
        self.__window.bind('<KP_Subtract>', self.speed_key)  # Клавиша - на
боковой клаве

    class CONST(Enum):  # Список возможных направлений движения и других
констант
        RIGHT = 1
        DOWN = 2
        LEFT = 3
        UP = 4
        SNAKE_HCOLOR = 'red'  # Цвет головы змейки
        SNAKE_BCOLOR = 'green'  # Цвет тела змейки
        CANVAS_BGCOLOR = '#f75394'  # Цвет фона холста
        SNAKE_THICKNESS = 11  # Толщина тела змейки (нечётное число)
        FOOD_THICKNESS = 15  # Толщина еды (нечётное число)
        FOOD_COLOR = '#aced95'  # Цвет тела еды
        EXPLOSIVE = 15  # Диаметр взрыва при столкновении змеи с препятствием
(нечётное число)
        EXPLOSIVE_BORD = 10  # толщина контура взрыва при столкновении змеи с
препятствием
        EXPLOSIVE_BCOLOR = '#ff9999'  # Цвет тела взрыва
        EXPLOSIVE_CCOLOR = '#881a1a'  # Цвет контура взрыва

    # обработчики клавиш изменения направления движения:
    def right(self, event):
        self.__vector = self.CONST.RIGHT.value

    def down(self, event):
        self.__vector = self.CONST.DOWN.value

    def left(self, event):
        self.__vector = self.CONST.LEFT.value

    def up(self, event):
        self.__vector = self.CONST.UP.value

    def speed_key(self, event):
        # print(event.keysym)
        if event.keysym == 'KP_Add' or event.keysym == 'plus':
```

```
            self.speed('+')
        elif event.keysym == 'KP_Subtract' or event.keysym == 'minus':
            self.speed('-')

    def create_head_food(self):
        rand_vect = random.randint(1, 4)
        if rand_vect == 1:
            self.__vector = self.CONST.RIGHT.value
        elif rand_vect == 2:
            self.__vector = self.CONST.DOWN.value
        elif rand_vect == 3:
            self.__vector = self.CONST.LEFT.value
        else:
            self.__vector = self.CONST.UP.value
        self.head = self.element_square(self, self.__snake_x,
                                        self.__snake_y,
                                        self.CONST.SNAKE_THICKNESS.value,
                                        self.CONST.SNAKE_HCOLOR.value)
        self.food.add(self)
        self.body = []
        self.body.append({'id': self.head.draw(),
                          'x': self.__snake_x,
                          'y': self.__snake_y})
        self.step('add')
        self.step('add')
        self.step('add')
        self.step('add')

    def speed(self, way):
        if way == '+' and self.__spped > 1:
            self.__spped -= 1
        elif way == '-' and self.__spped < 20:
            self.__spped += 1

    def reload(self):
        self.quit = 'n'
        self.__started = 1
        self.__spped = 10
        self.canv.delete('all')
        del self.body
        self.body = []
        self.create_head_food()
        self.start()

    def quit(self, event):
        self.quit = 'y'

    def move(self, event):
        if self.quit != 'n':
            self.start()

    def start(self):  # Бесконечный цикл движения змейки
        if self.__started == 1:
            self.quit = 'n'
            i = 0
            add = 'del'
            while i == 0:
                self.step(add)
                if self.food.eat(self) == 1:
                    add = 'add'
                    self.speed('+')
                elif add == 'add':
                    add = 'del'
                if self.bump_wall() == 'the end':
```

```
                    break
                if self.bump_body() == 'the end':
                    break
                for x in range(1, (self.__spped + 1)):
                    time.sleep(0.05)
                    self.__window.update()
                    if self.quit == 'y':
                        i = 1
                        break

    def bump_wall(self):  # Проверка на столкновение со стеной
        __head_x = self.body[-1]['x']
        __head_y = self.body[-1]['y']
        if ((__head_x < ((self.CONST.SNAKE_THICKNESS.value // 2) + 1))
                or (__head_y < ((self.CONST.SNAKE_THICKNESS.value // 2) + 1))
                or (__head_x > (self.canv_width
                                - (self.CONST.SNAKE_THICKNESS.value // 2) +
1))
                or (__head_y > (self.canv_height
                                - (self.CONST.SNAKE_THICKNESS.value // 2) +
1))):
            self.explosive()
            return 'the end'
        else:
            return 0

    def bump_body(self):  # Проверка на столкновение с телом змеи
        __head_x = self.body[-1]['x']
        __head_y = self.body[-1]['y']
        bump = 0
        for i in range(0, (len(self.body) - 1)):
            if ((__head_x == self.body[i]['x'])
                    and (__head_y == self.body[i]['y'])):
                self.explosive()
                bump = 'the end'
        return bump

    def explosive(self):
        self.__started = 0
        self.canv.create_oval((self.body[-1]['x']
                               - self.CONST.EXPLOSIVE.value),
                              (self.body[-1]['y']
                               - self.CONST.EXPLOSIVE.value),
                              (self.body[-1]['x']
                               + self.CONST.EXPLOSIVE.value),
                              (self.body[-1]['y']
                               + self.CONST.EXPLOSIVE.value),
                              fill=self.CONST.EXPLOSIVE_BCOLOR.value,
                              outline=self.CONST.EXPLOSIVE_CCOLOR.value,
                              width=self.CONST.EXPLOSIVE_BORD.value)

    def step(self, add):  # Двигать тело змеюки в текущую сторону на 1 шаг
        # При этом тело может увеличиться (add='add') в размерах или нет
        if self.__vector == self.CONST.RIGHT.value:
            deltax = self.CONST.SNAKE_THICKNESS.value
            deltay = 0
        elif self.__vector == self.CONST.DOWN.value:
            deltax = 0
            deltay = self.CONST.SNAKE_THICKNESS.value
        elif self.__vector == self.CONST.LEFT.value:
            deltax = -self.CONST.SNAKE_THICKNESS.value
            deltay = 0
        elif self.__vector == self.CONST.UP.value:
            deltax = 0
```

```
            deltay = -self.CONST.SNAKE_THICKNESS.value
        self.head.x += deltax
        self.head.y += deltay
        self.head = self.element_square(self, self.head.x, self.head.y,
                                         self.CONST.SNAKE_THICKNESS.value,
                                         self.CONST.SNAKE_HCOLOR.value)
        self.body.append({'id': self.head.draw(), 'x': self.head.x,
                          'y': self.head.y})
        self.canv.itemconfig(self.body[-2]['id'],
                             fill=self.CONST.SNAKE_BCOLOR.value)
        if add != 'add':
            self.canv.delete(self.body[0]['id'])
            self.body.pop(0)

    class food:
        def add(self):
            self.food.x = random.randint(self.CONST.FOOD_THICKNESS.value
                                         // 2, self.canv_width
                                         - self.CONST.FOOD_THICKNESS.value //
2)
            self.food.y = random.randint(self.CONST.FOOD_THICKNESS.value
                                         // 2, self.canv_height
                                         - self.CONST.FOOD_THICKNESS.value //
2)
            self.food.body = self.element_square(self, self.food.x,
                                                 self.food.y,

self.CONST.FOOD_THICKNESS.value,
                                                 self.CONST.FOOD_COLOR.value)
            self.food.id = self.food.body.draw()

        def eat(self):
            head_x = self.body[-1]['x']
            head_y = self.body[-1]['y']
            eat = 0
            if ((head_x
                 + self.CONST.SNAKE_THICKNESS.value // 2 > (self.food.x
                                                            -
self.CONST.FOOD_THICKNESS.value // 2))
                    and (head_x
                         - self.CONST.SNAKE_THICKNESS.value // 2 <
(self.food.x
                                                                    +
self.CONST.FOOD_THICKNESS.value // 2))
                    and (head_y
                         + self.CONST.SNAKE_THICKNESS.value // 2 >
(self.food.y
                                                                    -
self.CONST.FOOD_THICKNESS.value // 2))
                    and (head_y
                         - self.CONST.SNAKE_THICKNESS.value // 2 <
(self.food.y
                                                                    +
self.CONST.FOOD_THICKNESS.value // 2))):
                self.canv.delete(self.food.id)
                self.food.add(self)
                eat = 1
            return eat

    class element_square:
        def __init__(self, self_glob, x, y, d, color):
            self.self_glob = self_glob
            self.x = x
            self.y = y
```

```
            self.d = d
            self.color = color
            if (self.d % 2) == 0:
                self.d += 1  # сторону квадрата делаю нечётной

        def draw(self):
            x = self.x - (self.d // 2)
            y = self.y - (self.d // 2)
            return self.self_glob.canv.create_rectangle(x, y, x + self.d,
                                                        y + self.d,
                                                        fill=self.color,
                                                        width=2)


def main():
    image1_data =
'''R0lGODlhSgBKAOf/AABBAxJBCgBLAgdKBQRPABVKCwBTAAVRBwBYBRBXCQBfAQxcAAdeDAVhAA
BnAABoCwpmCQZnFQRsBQBtEwBvABtnFhNqEQByAAByCwJ1AglxFg9xCwB6AAB6CAt4BwB9ACVxJQe
AAgCDBRh6GwCEACB3JCV3HQCGAAeCFACHCwWJ'''
    image2_data =
'''R0lGODlhSgBKAOf/ABBHBwBOBARPAABTAABXBABbAAdZAAtXDgBeAA9dAQleDQJiAxheGgdlCA
BoAA1lAA5lEwRtBQJuEwBxAAB0AQlyDBhtExptHAl3BwB7AAB7CRB4AAB9ACNwJi1sMRp3EwiAAwG
BEgCDBSB3HgCEAACGAACHAACGCy12JQKIABiAFyt6MAGKEQeKADt3OCd/JQCPBxiHECp/LBKNBiKG
IQ+PFxiQCwaWEh+OITGJJjGJLj2FMzSINBiTHB2TEDqIPUKGQR2VHySUL0mHSR2YKiGYISmVKFGJU
j6RQD2TMCOcLiicJjeWO356hTKbLkqRTIN8eC+ '''

    def button_press(a):
        reload_button['image'] = reload_button_img2
        snake.reload()

    def button_unpress(a):
        reload_button['image'] = reload_button_img1

    root = Tk()
    root.title('Программа Змейка на питоне')
    root.geometry('800x600+150+150')

    frame = Frame(root, width=740, height=90, bg='#67E300')
    frame.place(x=30, y=5)
    text = Label(root, text='''Игра Змейка Правила: Змейка должна кушать
зелёные плоды. При съедании плода, скорость змейки возрастает. Скорость можно
отрегулировать вручную клавишами "+" и "-". Нельзя выползать за границы поля
и есть себя.''',
                 bg='#67E300', width=79)
    text.place(x=30, y=10)
    reload_button_img1 = PhotoImage(data=image1_data)
    reload_button = Label(image=reload_button_img1, bg='#f2ffe0')
    reload_button.place(x=675, y=13)
    reload_button_img2 = PhotoImage(data=image2_data)
    reload_button.bind('<Button-1>', button_press)
    reload_button.bind('<ButtonRelease-1>', button_unpress)

    snake = python_snake(root, 30, 100, 740, 470)
    snake.start()

    root.mainloop()


if __name__ == '__main__':
    main()
```

Экранные формы