Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по лабораторной работе № 4

Выполнил:
студент группы ИУ5-34Б:
Суслов Дмитрий Сергеевич
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю. Е.
Подпись и дата:

Москва, 2023 г.

Задание:
Напишем консольное приложение популярной игры “Сапёр”

```
from random import choice
import pyAesCrypt
import os


def continue_or_new():
    """Спрашиваем у пользователя, желает он продолжить старую
игру или начать новую"""
    s = input('Введите 0, если хотите начать новую игру, и 1,
если хотите продолжить старую: ')
    while not s.isdigit() or len(s) != 1:
        s = input('Ввод некорректен. \nВведите 0, если хотите
начать новую игру, и 1, если хотите продолжить старую: ')
    return bool(int(s))


def encryption(file, password):
    buffer_size = 512 * 1024
    pyAesCrypt.encryptFile(
        str(file),
        str(file) + ".crp",
        password,
        buffer_size
    )
    os.remove(file)


def decryption(file, password):
    buffer_size = 512 * 1024
    pyAesCrypt.decryptFile(
        str(file),
        str(os.path.splitext(file)[0]),
        password,
        buffer_size
    )


class Sapper:
    def __init__(self):
        self.user_lost = False
        self.user_win = False
        self.s_x = None
        self.s_y = None
        self.real_bombs = []
        self.field = []
        self.user_field = []
        self.users_bombs = []

    def __str__(self):
        s = ' ' + ' ' * (len(str(self.s_x)) + 1)
        for i in range(1, self.s_y + 1):
```

```
            s += ('  ' + str(i) + '  ' if not i // 10 else '  '
+ str(i) + ' ')
            s += ' '
        s += '\n'
        for i in range(2 * self.s_x + 1):
            for j in range(2 * self.s_y + 2):
                if j == 0:
                    if i % 2:
                        s += str(i // 2 + 1) + ' ' + ' ' *
((len(str(self.s_x))) - len(str(i // 2 + 1)))
                    else:
                        s += ' ' + ' ' * (len(str(self.s_x)))
                    continue
                if not i % 2:
                    if not j % 2:
                        s += '-----'
                    else:
                        s += ' '
                else:
                    if j % 2:
                        s += '|'
                    else:
                        x, y = i // 2, j // 2 - 1
                        if self.get_state():
                            if (x, y) in self.users_bombs:
                                s += '  ?  '
                            else:
                                user_value =
self.user_field[x][y]
                                if user_value > 0:
                                    value = self.field[x][y]
                                    if value:
                                        s += f'
{self.field[x][y]}  '
                                    else:
                                        s += '     '
                                else:
                                    s += '  #  '
                        else:
                            value = self.field[x][y]
                            if value > 0:
                                s += f'  {self.field[x][y]}  '
                            if value == 0:
                                s += '     '
                            if value == -1:
                                s += ' BUM '
            s += '\n'

        return s

    def save(self, file='save.txt'):
        """Сохранение игры"""
        try:
```

```
            decryption(str(file),
                       str(file) + '.crp')

        except Exception:
            pass

        finally:
            with open(file, 'w+') as f:
                f.write(str(int(self.get_state())))
                f.write('$')
                f.write(str(self.s_x))
                f.write('&')
                f.write(str(self.s_y))
                f.write('$')
                for i in range(self.s_x):
                    for j in range(self.s_y):
                        f.write(str(self.field[i][j]))
                        f.write(' ')
                    f.write('\n')

                f.write('$')

                for i in range(self.s_x):
                    for j in range(self.s_y):
                        f.write(str(self.user_field[i][j]))
                        f.write(' ')
                    f.write('\n')

                f.write('$')

                for i in range(len(self.real_bombs)):
                    f.write(str(self.real_bombs[i][0]) + '&' +
str(self.real_bombs[i][1]) + '\n')

                f.write('$')
                for i in range(len(self.users_bombs)):
                    f.write(str(self.users_bombs[i][0]) + '&' +
str(self.users_bombs[i][1]) + '\n')

            encryption(file, '')

    def load(self, file='save.txt.crp'):
        """Загрузка сохранённой игры"""
        try:
            decryption(file, '')
            with open(file[:-4], 'r+') as f:
                data = f.read().split('$')
                f.close()
                encryption(file[:-4], '')
                f = int(data[0])
                if not f:
                    raise FileNotFoundError
                self.s_x, self.s_y = map(int,
```

```
data[1].split('&'))
                field = data[2].split()
                field = list(map(int, field))
                field = [[field[self.s_y * j + i - 1] for i in
range(1, self.s_y + 1)] for j in range(self.s_x)]
                self.field = field

                field = data[3].split()
                field = list(map(int, field))
                field = [[field[self.s_y * j + i - 1] for i in
range(1, self.s_y + 1)] for j in range(self.s_x)]
                self.user_field = field
                real_bombs = data[4].split()
                self.real_bombs = [tuple(map(lambda x: int(x),
real_bombs[i].split('&'))) for i in range(len(real_bombs))]
                users_bombs = data[5].split()
                self.users_bombs = [tuple(map(lambda x: int(x),
users_bombs[i].split('&'))) for i in range(len(users_bombs))]
                print(self)
                return

        except Exception:
            print('Сохранение не найдено, начинаем новую
игру...')
            self.setup()

    def get_state(self):
        """Проверка на завершение партии"""
        return not(self.user_lost or self.user_win)

    def check_win(self):
        """Проверка на победу"""
        return sorted(self.users_bombs) ==
sorted(self.real_bombs) \
               and (self.s_x * self.s_y) - sum([sum(row) for row
in self.user_field]) == len(self.real_bombs)

    def generate_field(self, x, y, n_bombs):
        """Генерация поля в зависимости от количества бомб и
первого хода"""
        all_cords = [[(i, j) for j in range(self.s_y)] for i in
range(self.s_x)]
        possible_cords = []
        margin = 1
        for cords in all_cords:
            possible_cords.extend(list(filter(lambda t:
(abs(t[0] - x) > margin) or (abs(t[1] - y) > margin), cords)))
        for i in range(n_bombs):
            bomb = choice(possible_cords)
            bomb_index = possible_cords.index(bomb)
            possible_cords = possible_cords[:bomb_index] +
possible_cords[bomb_index + 1:]
            self.real_bombs.append(bomb)
```

```
            self.field[bomb[0]][bomb[1]] = -1

        for i in range(self.s_x):
            for j in range(self.s_y):
                if self.field[i][j] == -1:
                    cords = [(i + 1, j), (i - 1, j), (i, j + 1),
(i, j - 1), (i - 1, j - 1), (i - 1, j + 1),
                             (i + 1, j + 1), (i + 1, j - 1)]
                    for c in cords:
                        try:
                            if c[0] >= 0 and c[1] >= 0:
                                if self.field[c[0]][c[1]] != -1:
                                    self.field[c[0]][c[1]] += 1
                        except IndexError:
                            continue

    def get_action(self):
        """Получение команды от пользователя"""
        s = input('Введите координаты клетки и действие (Open,
Flag) через пробел (Пример: 2 4 Open),'
                  ' либо "Save", чтобы сохранить игру:
').split()

        while True:
            if s[0] == 'Save':
                self.save()
                return 0, 0, ''
            if len(s) == 3:
                x, y, action = s
                if (x + y).isdigit() and action in ('Open',
'Flag'):
                    if (0 < int(x) <= self.s_x) and (0 < int(y)
<= self.s_y):
                        return int(x) - 1, int(y) - 1, action
                    else:
                        s = input('Данное действие невозможно.
Проверьте правильность ввода и попробуйте ещё: ').split()
                else:
                    s = input('Данное действие невозможно.
Проверьте правильность ввода и попробуйте ещё: ').split()
            else:
                s = input('Данное действие невозможно. Проверьте
правильность ввода и попробуйте ещё: ').split()

    def set_params(self, x, y):
        """Установка параметров"""
        self.field = [[0 for i in range(y)] for j in range(x)]
        self.user_field = [[0 for i in range(y)] for j in
range(x)]
        self.s_x = x
        self.s_y = y

    def setup(self):
```

```
        """Получение настроек"""
        s = input('Введите желаемый размер поля, а также
желаемое количество мин (Пример: 5 5 2): ').split()
        while True:
            if len(s) == 3:
                if (s[0] + s[1] + s[2]).isdigit():
                    if 20 >= min(int(s[0]), int(s[1])) >= 3 and
20 >= max(int(s[0]), int(s[1])) >= 4 and int(s[2]) > 0:
                        if int(s[2]) <= int(s[0]) * int(s[1]) -
9:
                            x, y, n_bombs = map(int, s)
                            break

                        else:
                            s = input(
                                f'При введённом размере поля
максимальное чсило мин - {int(s[0]) * int(s[1]) - 9}\n'
                                f'Попробуйте снова: ').split()

                    else:
                        s = input(
                            'Минимальный размер поля - 4 * 3.\n'
                            'Максимальный размер поля - 20 *
20\n'
                            'Минимальное количество мин - 1\n'
                            'Проверьте правильность ввода и
попробуйте ещё: ').split()
                else:
                    s = input('Введённые данные некорректны.
Проверьте правильность ввода и попробуйте ещё: ').split()
            else:
                s = input('Введённые данные некорректны.
Проверьте правильность ввода и попробуйте ещё: ').split()

        self.set_params(x, y)
        print(self)
        s = input('Введите через пробел координаты клетки,
которую хотите открыть первой (Пример: 2 2): ').split()
        while True:
            if len(s) == 2 and (s[0] + s[1]).isdigit():
                if 0 < int(s[0]) <= self.s_x and 0 < int(s[1])
<= self.s_y:
                    x, y = int(s[0]) - 1, int(s[1]) - 1
                    break
                else:
                    s = input('Введённые данные некорректны.
Проверьте правильность ввода и попробуйте ещё: ').split()
            else:
                s = input('Введённые данные некорректны.
Проверьте правильность ввода и попробуйте ещё: ').split()
        self.generate_field(x, y, n_bombs)
        self.open(x, y)
        print(self)
```

```
    def open(self, x, y):
        """Открытие ячейки"""
        if x < 0 or x >= self.s_x or y < 0 or y >= self.s_y:
            return

        if self.user_field[x][y] == 1:
            return

        for k in range(len(self.users_bombs)):
            if self.users_bombs[k] == (x, y):
                self.users_bombs = self.users_bombs[:k] +
self.users_bombs[k + 1:]
                break

        if self.field[x][y] == -1:
            self.user_lost = True
            return

        if self.field[x][y] == 0:
            cords = [(x + 1, y), (x - 1, y), (x, y + 1), (x, y -
1), (x - 1, y - 1), (x - 1, y + 1),
                           (x + 1, y + 1), (x + 1, y - 1)]
            self.user_field[x][y] = 1
            for c in cords:
                self.open(c[0], c[1])
            return

        if self.field[x][y] > 0:
            self.user_field[x][y] = 1
            cords = [(x + 1, y), (x - 1, y), (x, y + 1), (x, y -
1), (x - 1, y - 1), (x - 1, y + 1),
                     (x + 1, y + 1), (x + 1, y - 1)]
            cnt = 0
            t_cords = []
            for c in cords:
                if c in self.users_bombs:
                    cnt += 1
                    t_cords.append(c)

            if cnt == self.field[x][y]:
                for c in set(cords) - set(t_cords):
                    self.open(c[0], c[1])

            return

    def update(self):
        """Совершение хода"""
        x, y, action = self.get_action()
        if action == '':
            print('Игра успешно сохранена')
            print(self)
            return
```

```
        if action == 'Open':
            self.open(x, y)
        if action == 'Flag':
            self.set_flag(x, y)
        if self.check_win():
            self.user_win = True
        print(self)
        return

    def set_flag(self, x, y):
        """Установка флага"""
        if self.user_field[x][y] == 0 and (x, y) not in
self.users_bombs:
            self.users_bombs.append((x, y))
        return


while True:
    game = Sapper()
    answer = continue_or_new()

    if not answer:
        game.setup()
    else:
        game.load()

    while game.get_state():
        game.update()

    game.save()
    if game.user_win:
        print('Win!')
    else:
        print('Бууууууум!\nGame Over')
```

Код:

```
from math import exp, cosh
from abc import ABC


class ActivationFunction(ABC):
    def __init__(self):
        self.t = None

    def forward(self, x):
        pass

    def backward(self):
        pass


class ReLU(ActivationFunction):
```

```
    def __init__(self):
        super().__init__()

    def forward(self, x):
        self.t = x
        return max(0, x)

    def backward(self):
        return 1 if self.t >= 0 else 0


class Sigmoid(ActivationFunction):
    def __init__(self):
        super().__init__()

    def forward(self, x):
        if self.t is None:
            self.t = x
        return 1 / (1 + exp(-x))

    def backward(self):
        return self.forward(self.t) * (1 - self.forward(self.t))


class Tanh(ActivationFunction):
    def __int__(self):
        super().__init__()

    def forward(self, x):
        self.t = x
        return (exp(x) - exp(-x)) / (exp(x) + exp(-x))

    def backward(self):
        return 1 / cosh(self.t) ** 2


a = 10

sigm = Sigmoid()
relu = ReLU()
tanh = Tanh()

print(f'Sigmoid: Forward: {sigm.forward(a)} Backward:
{sigm.backward()}')
print(f'ReLU: Forward: {relu.forward(a)} Backward:
{relu.backward()}')
print(f'Tanh: Forward: {tanh.forward(a)} Backward:
{tanh.backward()}')
```

Не забудем собрать standalone приложение

```
Введите 0, если хотите начать новую игру, и 1, если хотите продолжить старую: 0
Введите желаемый размер поля, а также желаемое количество мин (Пример: 5 5 2): 8 8 10
    1     2     3     4     5     6     7     8
  ----- ----- ----- ----- ----- ----- ----- -----
1 |  #  |  #  |  #  |  #  |  #  |  #  |  #  |  #  |
  ----- ----- ----- ----- ----- ----- ----- -----
2 |  #  |  #  |  #  |  #  |  #  |  #  |  #  |  #  |
  ----- ----- ----- ----- ----- ----- ----- -----
3 |  #  |  #  |  #  |  #  |  #  |  #  |  #  |  #  |
  ----- ----- ----- ----- ----- ----- ----- -----
4 |  #  |  #  |  #  |  #  |  #  |  #  |  #  |  #  |
  ----- ----- ----- ----- ----- ----- ----- -----
5 |  #  |  #  |  #  |  #  |  #  |  #  |  #  |  #  |
  ----- ----- ----- ----- ----- ----- ----- -----
6 |  #  |  #  |  #  |  #  |  #  |  #  |  #  |  #  |
  ----- ----- ----- ----- ----- ----- ----- -----
7 |  #  |  #  |  #  |  #  |  #  |  #  |  #  |  #  |
  ----- ----- ----- ----- ----- ----- ----- -----
8 |  #  |  #  |  #  |  #  |  #  |  #  |  #  |  #  |
  ----- ----- ----- ----- ----- ----- ----- -----

Введите через пробел координаты клетки, которую хотите открыть первой (Пример: 2 2): 2 4
    1     2     3     4     5     6     7     8
  ----- ----- ----- ----- ----- ----- ----- -----
1 |     |     |     |     |     |     |     |     |
  ----- ----- ----- ----- ----- ----- ----- -----
2 |     |     |     |     |     |     |     |     |
  ----- ----- ----- ----- ----- ----- ----- -----
3 |  1  |  1  |     |     |     |     |     |     |
  ----- ----- ----- ----- ----- ----- ----- -----
4 |  #  |  2  |  1  |  1  |  1  |  1  |  1  |  1  |
  ----- ----- ----- ----- ----- ----- ----- -----
5 |  #  |  #  |  #  |  #  |  #  |  #  |  #  |  #  |
  ----- ----- ----- ----- ----- ----- ----- -----
6 |  #  |  #  |  #  |  #  |  #  |  #  |  #  |  #  |
  ----- ----- ----- ----- ----- ----- ----- -----
7 |  #  |  #  |  #  |  #  |  #  |  #  |  #  |  #  |
  ----- ----- ----- ----- ----- ----- ----- -----
8 |  #  |  #  |  #  |  #  |  #  |  #  |  #  |  #  |
  ----- ----- ----- ----- ----- ----- ----- -----

Введите координаты клетки и действие (Open, Flag) через пробел (Пример: 2 4 Open), либо "Save", чтобы сохранить игру: 4 1 Flag
     1     2     3     4     5     6     7     8
   ----- ----- ----- ----- ----- ----- ----- -----
1 |     |     |     |     |     |     |     |     |
   ----- ----- ----- ----- ----- ----- ----- -----
2 |     |     |     |     |     |     |     |     |
   ----- ----- ----- ----- ----- ----- ----- -----
3 |  1  |  1  |     |     |     |     |     |     |
   ----- ----- ----- ----- ----- ----- ----- -----
4 |  ?  |  2  |  1  |  1  |  1  |  1  |  1  |  1  |
   ----- ----- ----- ----- ----- ----- ----- -----
5 |  #  |  #  |  #  |  #  |  #  |  #  |  #  |  #  |
   ----- ----- ----- ----- ----- ----- ----- -----
6 |  #  |  #  |  #  |  #  |  #  |  #  |  #  |  #  |
   ----- ----- ----- ----- ----- ----- ----- -----
7 |  #  |  #  |  #  |  #  |  #  |  #  |  #  |  #  |
   ----- ----- ----- ----- ----- ----- ----- -----
8 |  #  |  #  |  #  |  #  |  #  |  #  |  #  |  #  |
   ----- ----- ----- ----- ----- ----- ----- -----
```