Министерство науки и высшего образования РФ

ФГАОУ ВО Пермский национальный исследовательский

политехнический университет

Кафедра «Вычислительная математика, механика и биомеханика»

Отчет по лабораторной работе № 3

тема «Игра+ Массив»

по дисциплине «Учебная Исследовательская Работа»

Выполнил: студент группы ИСТ-22-1б Афонина А.М

Проверил: Сахатбудинова Л.Р

Пермь, 2023

**Содержание**

[Задание 1 3](#_Toc149495170)

[1.1. Постановка задачи 3](#_Toc149495171)

[1.2. Решение задачи, код программы 3](#_Toc149495172)

[1.3. Тестирование работы программы 5](#_Toc149495173)

[Задание 2 5](#_Toc149495174)

[1.1.Постановка задачи 5](#_Toc149495175)

[1.2. Решение задачи, код программы 6](#_Toc149495176)

[1.3. Тестирование работы программы 8](#_Toc149495177)

# Задание 1

## 1.1. Постановка задачи

Составить игру «Камень – Ножницы – Бумага» на ЯП “Python”

Задачи

1) Пользователю предлагается сыграть в игру

2) Пользователь может вводить цифры от 1 до 3 , которые обознают камень , ножницы, бумагу соответственно

3) Пользователь играет с компьютером

4) На консоли долдно вывести, кто выйграл в игре

5) Программа также предлагает пользователю сыграть снова в эту игру

6) Программа должна учитывать также тот факт, если человек хочет поиграть неопределённое количество раз

## 1.2. Решение задачи, код программы

**import** **random**

**def** **vibor\_chel**():

**print**("Выбери, камень(1), ножницы(2), бумага(3)")

vibor = int(input())

**if** vibor == **1**:

**print**("вы выбрали камень")

**if** vibor == **2**:

**print**("вы выбрали ножницы")

**if** vibor == **3**:

**print**("вы выбрали бумага")

**while** vibor **not** **in** [**1**,**2**,**3**]:

vibor = input("Не та цифра, выбирай еще")

vibor = int(input())

**return** vibor

**def** **vibor\_computer**():

copik = random.randint(**1**,**3**)

**if** copik == **1**:

**print**("комп выбрал камень")

**if** copik == **2**:

**print**("комп выбрал ножницы")

**if** copik == **3**:

**print**("комп выбрал бумага")

**return** copik

**def** **IGRA**(chelik, copik):

**if** copik == chelik:

win=**0**

**if** chelik == **1** **and** copik == **2**:

win=**1**

**if** chelik == **1** **and** copik == **3**:

win=**2**

**if** chelik == **2** **and** copik == **3**:

win=**1**

**if** chelik == **3** **and** copik == **1**:

win=**1**

**if** chelik == **3** **and** copik == **2**:

win=**2**

**if** chelik == **2** **and** copik == **1**:

win=**2**

**if** win == **0**:

**return** ("Ничья")

**if** win == **1**:

**return**("Умничка, выйграл")

**if** win ==**2**:

**return**("Комп выйграл")

**def** **game**():

a = vibor\_chel()

b = vibor\_computer()

c = IGRA(a,b)

**return** c

**def** **ask**():

**print**("Хотите продолжить игру?, 1 - true, 2 - false")

t = int(input())

**while** t!=**1** **and** t!=**2**:

**print**("Введи другие данные")

t = int(input())

**if** t ==**1**:

**while** ( t == **1**):

k = game()

**print** (k)

**return** ask()

**if** t == **2**:

**print**("Poka")

**exit(0)**

**print**("Привет, хочешь поиграть в камень ножницы бумага true/false?")

answer = bool(input())

**if** answer == False:

**print**("ну ладненько")

**else**:

d = game()

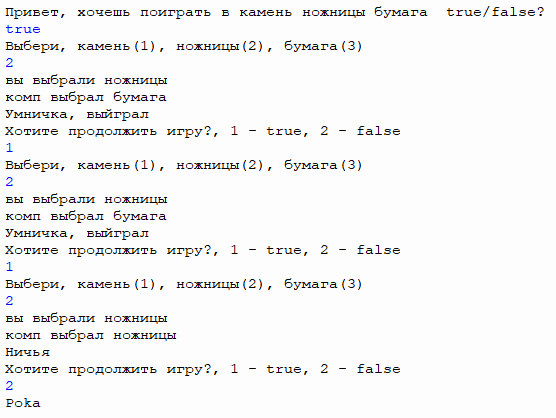
**print**(d)

w = ask()

**print**(w)

1. Функция **vibor\_chel()** отвечает за выбор пользователя. Она выводит на экран сообщение с вариантами выбора (камень, ножницы, бумага) и возвращает выбор пользователя.
2. Функция **vibor\_computer()** выбирает случайным образом вариант компьютера и выводит сообщение о его выборе. Она возвращает выбор компьютера.
3. Функция **IGRA(chelik, copik)** принимает выбор пользователя и выбор компьютера в качестве аргументов и определяет результат игры. Результат (победа пользова-теля, победа компьютера или ничья) возвращается в виде строки.
4. Функция **game()** выполняет одну игру, вызывая функции **vibor\_chel()** и **vibor\_computer()** для выбора пользователя и компьютера соответственно, а затем передает эти выборы в функцию **IGRA()** для определения результата. Результат иг-ры возвращается.
5. Функция **ask()** выводит сообщение с предложением продолжить игру и принима-ет ввод пользователя (1 для продолжения, 2 для выхода). Если пользователь вы-бирает продолжить, то вызывается функция **game()** для новой игры. Если пользо-ватель выбирает выход, то выводится сообщение "Poka" и происходит остановка программы.
6. В основном блоке кода выводится приветственное сообщение и принимается ввод пользователя **(true/false)** для начала игры. Если пользователь выбирает false, то выводится сообщение "ну ладненько". В противном случае, запускается одна игра с помощью функции **game(),** результат выводится на экран, затем вызывается функция **ask()** для предложения продолжить игру или выйти.

## 1.3. Тестирование работы программы



# Задание 2

## Постановка задачи

Необходимо заполнить матрицу А[n\*n] по одному из описанных алгоритов.

Затем отразить ее "зеркально" вправо, получив при этом матрицу размера B[n\*2n].Полученную матрицу B отразить "зеркально" вниз, полив матрицу C [2n\*2n]

## 1.2. Решение задачи, код программы

**def** **matrica**(n):

matrix = [[**0**] \* n **for** \_ **in** range(n)]

num = **1**

left = **0**

right = n - **1**

top = **0**

bottom = n - **1**

**while** left <= right **and** top <= bottom:

# Заполнение верхней горизонтальной строки

**for** i **in** range(left, right + **1**):

matrix[top][i] = num

num += **1**

top += **1**

# Заполнение правого вертикального столбца

**for** i **in** range(top, bottom + **1**):

matrix[i][right] = num

num += **1**

right -= **1**

# Заполнение нижней горизонтальной строки в обратном порядке

**if** top <= bottom:

**for** i **in** range(right, left - **1**, -**1**):

matrix[bottom][i] = num

num += **1**

bottom -= **1**

# Заполнение левого вертикального столбца в обратном порядке

**if** left <= right:

**for** i **in** range(bottom, top - **1**, -**1**):

matrix[i][left] = num

num += **1**

left += **1**

**return** matrix

**def** **matrica\_right**(matrix):

n = len(matrix)

matrix\_b = [[**0**] \* (**2** \* n) **for** \_ **in** range(n)]

**for** i **in** range(n):

**for** j **in** range(n):

matrix\_b[i][j] = matrix[i][j]

matrix\_b[i][j+n] = matrix[i][n-j-**1**]

**return** matrix\_b

**def** **matrica\_down**(matrix):

n = len(matrix)

matrix\_c = [[**0**] \* (**2** \* n) **for** \_ **in** range(**2** \* n)]

**for** i **in** range(n):

**for** j **in** range(**2** \* n):

matrix\_c[i][j] = matrix[i][j]

matrix\_c[**2** \* n - i - **1**][j] = matrix[i][j]

**return** matrix\_c

# Пример использования

n = int(input("Введите размер матрицы"))

matrix\_a = matrica(n)

matrix\_b = matrica\_right(matrix\_a)

matrix\_c = matrica\_down(matrix\_b)

# Вывод результатов

**print**("Матрица A:")

**for** row **in** matrix\_a:

**print**(row)

**print**("**\n**Матрица B:")

**for** row **in** matrix\_b:

**print**(row)

**print**("**\n**Матрица C:")

**for** row **in** matrix\_c:

**print**(row)

1. Функция **matrica** заполняет матрицу **размером n\*n** с помощью метода для заполнения спирали. Мы создаем матрицу с нулями, а затем с помощью цикла while последовательно заполняем ячейки матрицы числами от 1 до n\*n.
2. Затем у нас есть две вспомогательные функции: **matrica\_right**(matrix) и **matrica\_down**(matrix)
3. Функция **matrica\_right** принимает матрицу **matrix** и отражает ее "зеркально" вправо, создавая матрицу **matrix\_b** размером **n\*2n**. Мы создаем новую матрицу **matrix\_b** с двойной шириной, а затем копируем значения из matrix в соответствующие ячейки матрицы matrix\_b, а с другой стороны копируем значения из matrix в обратном порядке.
4. Функция **matrica\_down** принимает матрицу matrix и отражает ее "зеркально" вниз, создавая матрицу **matrix\_c** размером 2n\*2n. Мы создаем новую матрицу **matrix\_c** с двойной высотой, а затем копируем значения из matrix в соответствующие ячейки матрицы **matrix\_c**, а с другой стороны копируем значения из **matrix** в обратном порядке.
5. Наконец, ниже приведен пример использования этих функций:
6. Мы сначала заполняем матрицу A с помощью функции **matrica**, затем отражаем ее вправо, создавая матрицу B с помощью функции **matrica\_right**, а затем отражаем матрицу B вниз, создавая матрицу C с помощью функции **matrica\_down**. Затем мы выводим все три матрицы на печать.

## 1.3. Тестирование работы программы

