Санкт-Петербургский Национальный Исследовательский Университет Информационных Технологий, Механики и Оптики

Факультет Инфокоммуникационных Технологий

**Домашняя работа №5**

**Вариант № 5**

Выполнили:

Бакланова А.Г,

Крылов Д.С,

Улитина М.С

Проверил Мусаев А.А.

Санкт-Петербург, 2023

Оглавление

[Оглавление 2](#_Toc135351154)

[Введение. 3](#_Toc135351155)

[Алгоритмы выполнения заданий. 4](#_Toc135351156)

[Задание 1. 4](#_Toc135351157)

[Задание 2. 6](#_Toc135351158)

[Задание 3. 6](#_Toc135351159)

[Задание 4. 8](#_Toc135351160)

[Задание 5. 8](#_Toc135351161)

[Задание 6. 8](#_Toc135351162)

[Задание 7. 9](#_Toc135351163)

[Заключение. 11](#_Toc135351164)

# Введение.

Цель лабораторной работы – выполнение семи заданий с помощью Python по «Алгоритмам и структурам данных». Вариант работы – 4.

# 

# Алгоритмы выполнения заданий.

## Задание 1.

В первом заданий нужно разработать алгоритм, который проверяет результат игры в крестики-нолики (3\*3)

На рисунке 1 приведен алгоритм работы программы. В 1 строке создается поле для игры. С 3-7 строку функция вывода процесса игры. Функция check\_win() проверяет, есть ли в данный момент выигрышная позиция, а именно: 10-12 строка проверяет горизонтали, с 13 -15 рассматривает вертикали, далее – проверка диагоналей.



Рисунок 1 – Функция проверки.

На рисунке 2 происходит вывод игрового поля, после ввод координаты клетки от игрока, проверка корректности введенных данных. После проверяем есть ли победная позиция, если да – выводим победителя, если нет продолжаем и проверяем есть ли еще свободные клетки, если есть-продолжаем, в ином случае выводим – ничья.

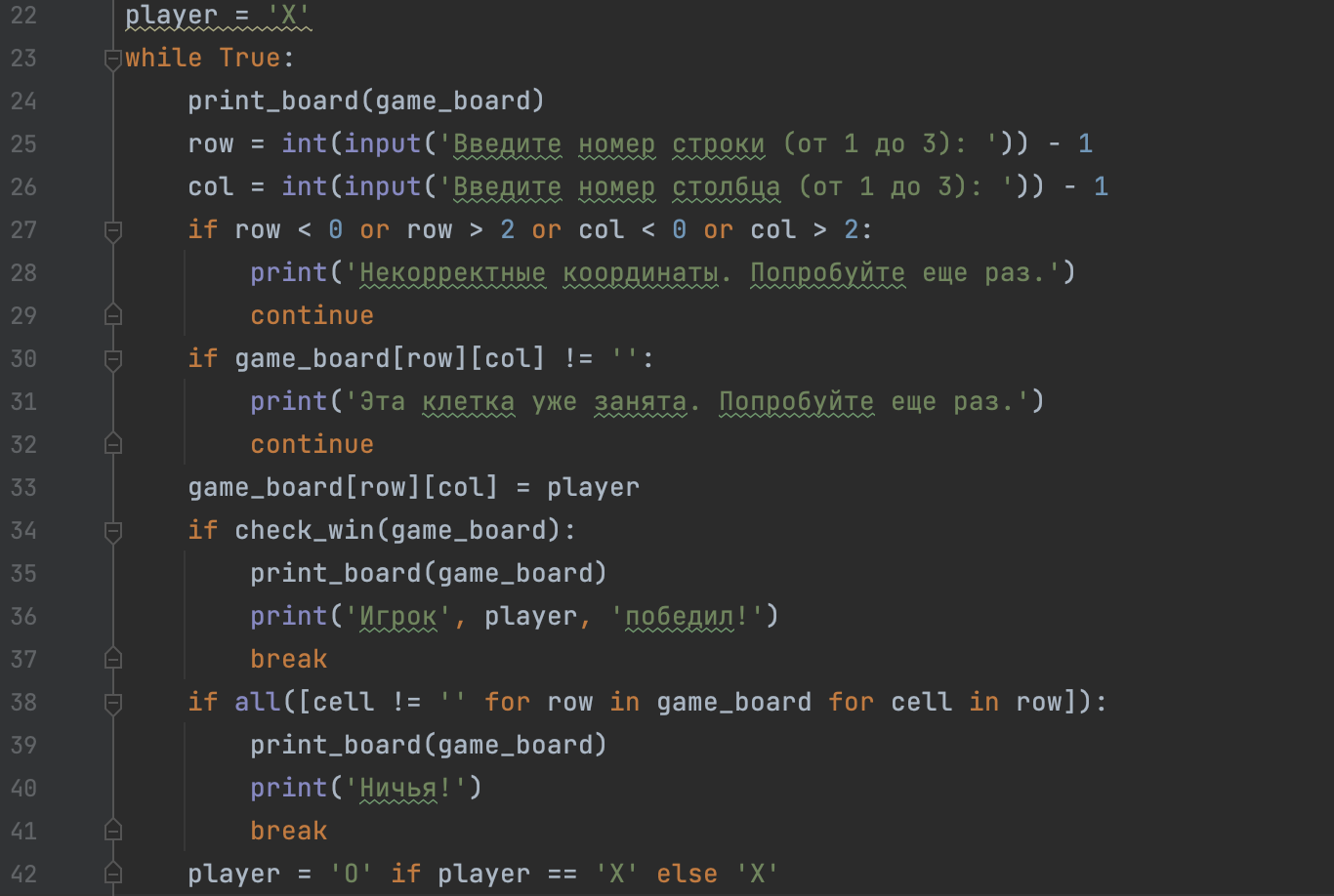


Рисунок 2 – Процесс игры.

На рисунке 3 результат работы программы.

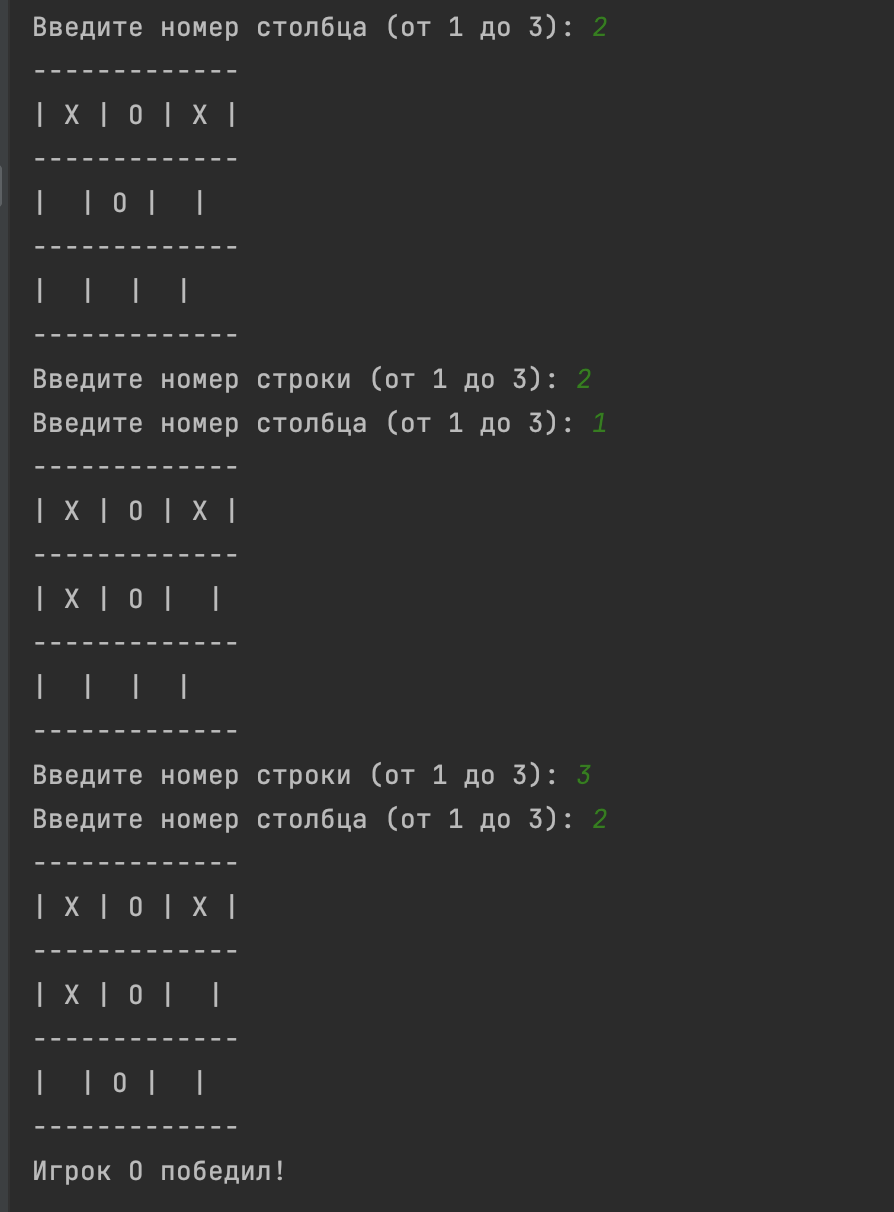


Рисунок 3 – Результат работы программы.

## Задание 2.

Данная программа для зaдaннoй мaтpицы MxN, в кoтopoй кaждaя стpoкa и стoлбeц oтсopтиpoвaны пo вoзpaстaнию, ищет введенный пользователем элемент.

На рисунке 4 приведен алгоритм работы программы. Создается функция search\_element(). Определяются размеры матрицы, поиск начинается с правого верхнего угла, если элемент совпадает, то выводим результат, если нет и значение в матрице меньше, чем введенный элемент, то спускаемся вниз, в другом случае двигаемся влево. В 13 строке создан массив, пользователь вводит значение и запускается функция.

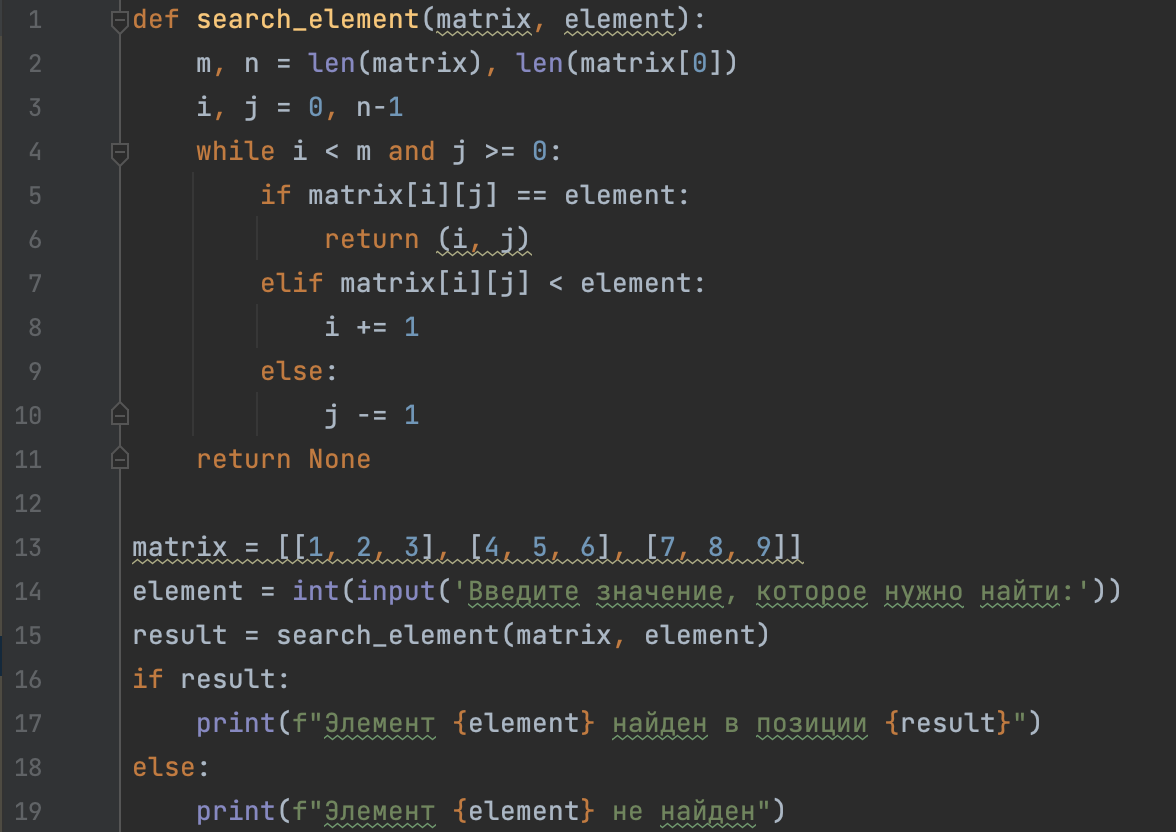


Рисунок 4 – Алгоритм работы программы.

На рисунке 5 результат работы программы (позиция считаются с нулевой).

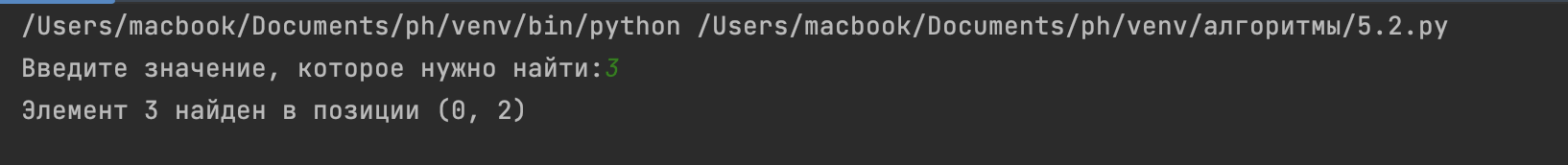


Рисунок 5 – Результат работы программы.

## Задание 3.

В третьем задание необходимо написать алгоритм, находящий все варианты расстановки восьми ферзей на шахматной доске paзмepoм 8х8 так, чтобы никакие две фигуры не располагались на одной горизонтали, вертикали или диагонали.

На рисунке 6 алгоритм работы программы. Реализовано через матрицу, то есть 1 в матрице будет обозначать стоящего на этом месте ферзя. После происходит дальнейшая расстановка, проверяя расположение всех (чтобы не было на вертикали, горизонтали или диагонали). Выводится количество вариантов расстановок и пример расстановки.



Рисунок 6 – Алгоритм работы программы.

На рисунке 7 результат работы программы.

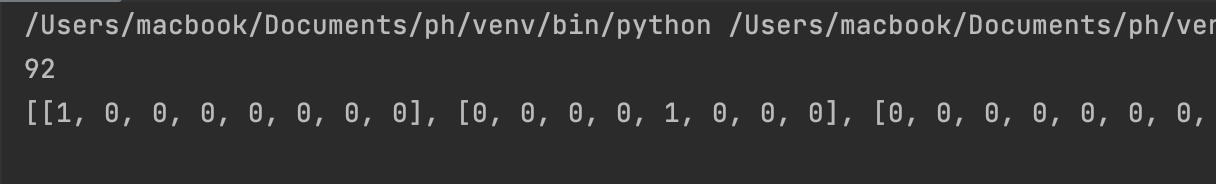


Рисунок 7 – Результат работы программы.

## Задание 4.

В четвертом задании необходимо решить задачу: «Ребёнок поднимается пo лестнице из n ступенек. За один шаг он может переместиться на одну, две или три ступеньки. Реализуете метод, рассчитывающий количество возможных вариантов перемещения ребёнка пo лестнице».  
На рисунке 8 приведен алгоритм работы программы. Пользователь вводит количество ступенек, создается массив хранения вариантов, созданы варианты ходов, через цикл вычисляются варианты ходов. Выводится результат.

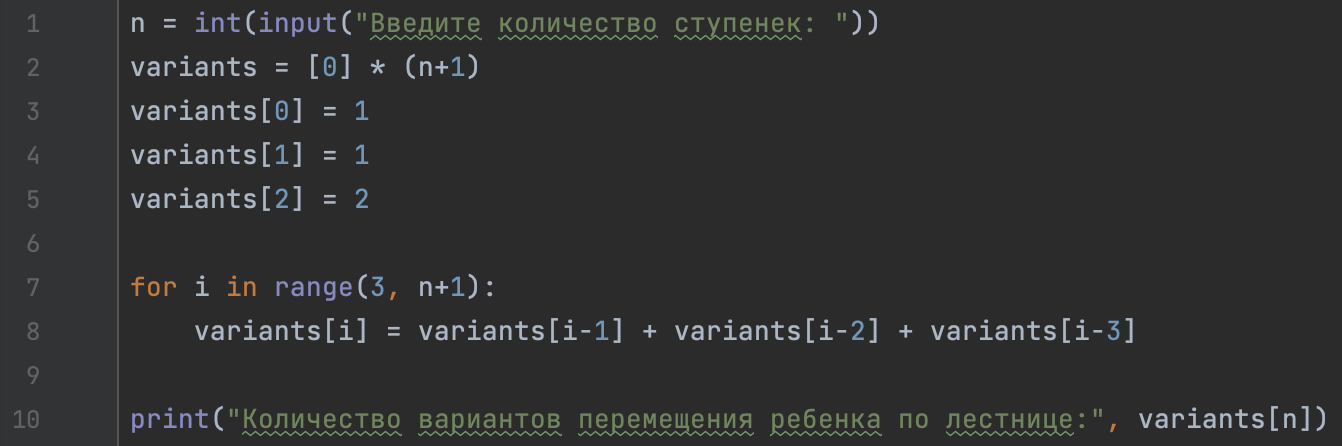


Рисунок 8 – Алгоритм работы программы.

Рисунок 9 показывает результат работы.

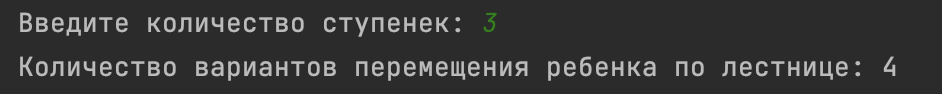


Рисунок 9 – Результат работы.

## Задание 5.

Нужно описать, как реализовать три стека с помощью одномерного массива можно.

Мы решили, что мы разделяем массив на три равные части и используем каждую часть для хранения элементов одного из стеков. Для понимания стеков, можно использовать указания: увеличивать показатель стека и отображать последний элемент «стека». Так мы сможем понимать, где стеки, и использовать одномерный массив для 3 стеков.

## Задание 6.

Необходимо нaписать мaксимaльнo кopoткий кoд для экспoнeнциaльнoгo фильтpa.

На рисунке 10 приведен алгоритм работы. Создана функция, получающая на вход массив, который нужно отфильтровать и коэффициент сглаживания. Получив данные, рассчитывается по формуле (строка 7).

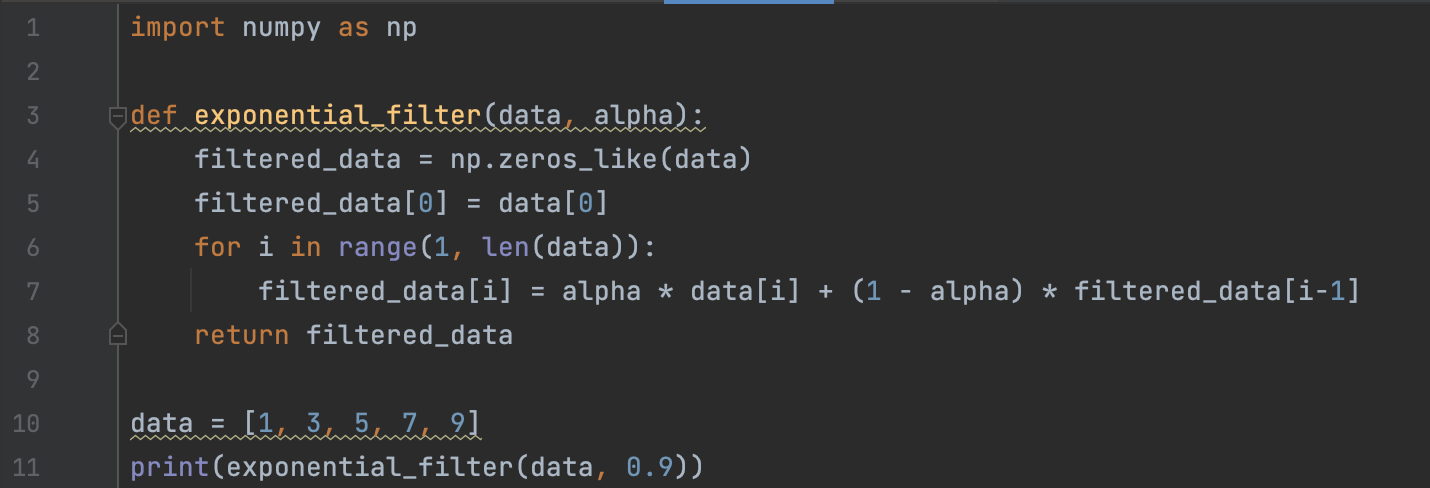


Рисунок 10 – Алгоритм работы программы.

Рисунок 11 показывает результат работы программы.

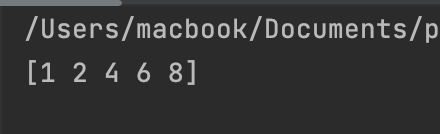


Рисунок 11 – Результат работы программы.

## Задание 7.

В этом задании нам необходимо вернуть наименьшее пропущенное число из неотсортированного массива целых чисел. Алгоритм за время O(n).

На рисунке 12 алгоритм работы программы. Делим на 3 блока, соответствующие определенным числам. После распределяем числа, тогда видя пустую корзину, понимаем, что она соответствует первому пропущенному числу.

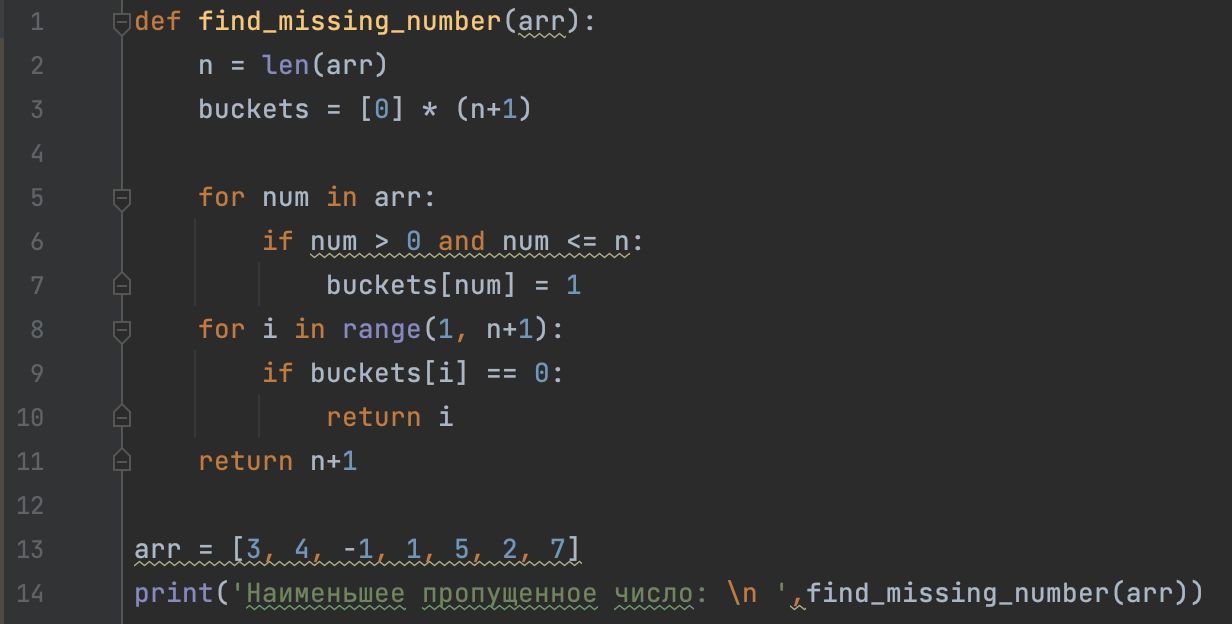


Рисунок 12 – Алгоритм работы программы.

На рисунке 13 результат работы.

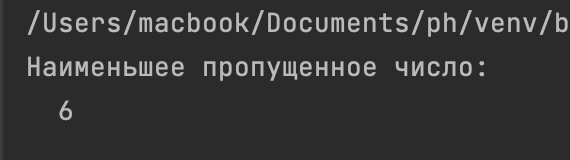


Рисунок 13 – Результат работы программы.

# Заключение.

Задания выполнены, таким образом, можно считать, что цель работы выполнена. Программы находятся по ссылке: <https://github.com/UlitiM2/hw5>