Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Ульяновский государственный технический университет»

Кафедра: «Вычислительная техника»

Лабораторная работа №1

Дисциплина: «Разработка профессиональных приложений»

«Обработка списков»

Вариант 22

Выполнил

Студент группы ИВТАСбд-22

Хабибуллин Х. Р.

Проверил

преподаватель кафедры

«Вычислительная техника»

Исхаков И. И.

Ульяновск, 2023

**Задание по варианту**

Реализовать быструю сортировку.

**Описание реализации**

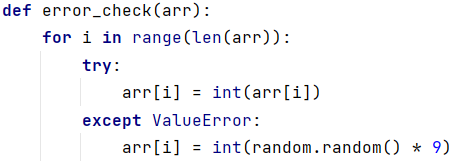
Для ввода элементов с клавиатуры используем *input* и его метод *split*, который разделит строку на список.

*Листинг 1. Ввод списка с клавиатуры*



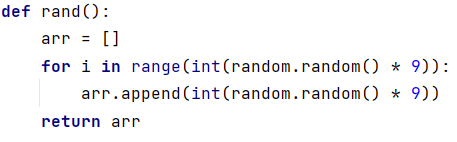
Для проверки корректности ввода, создадим функцию, которая, с помощью *try* и *except*, будет поэлементно приводить список к *int*, а при возникновении ошибки, если элемент не окажется числом, заменять его на случайное число.

*Листинг 2. Проверка корректности ввода*



Для автоматической генерации списка создадим функцию, которая, с помощью библиотеки *random*, будет генерировать список со случайной длиной и со случайными элементами.

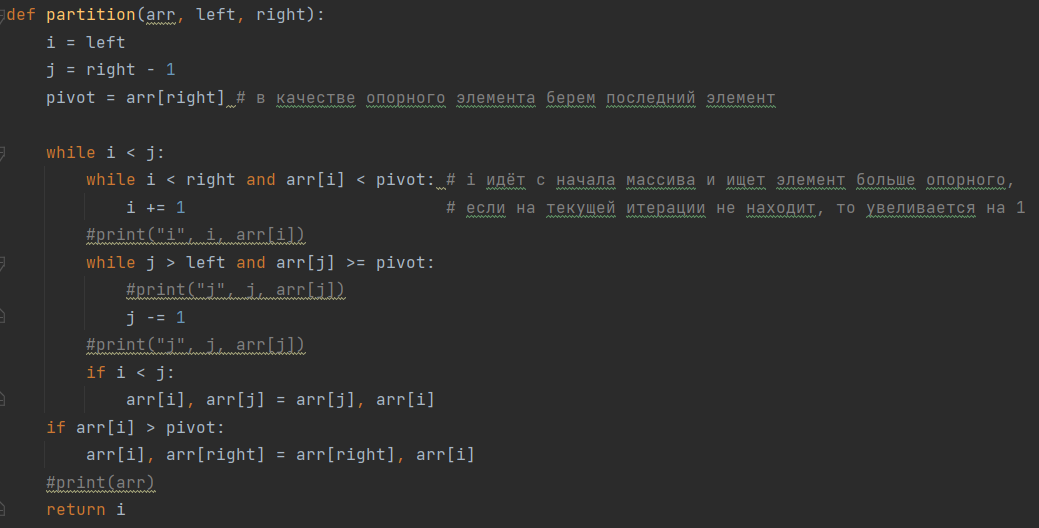
*Листинг 3. Генерация списка*



Выбор опорного элемента: Выбирается опорный элемент из массива. Этот элемент будет использоваться для разделения массива на две части. Массив перераспределяется таким образом, что элементы меньше опорного элемента помещаются перед ним, а элементы больше опорного - после него. Опорный элемент становится на свое место.

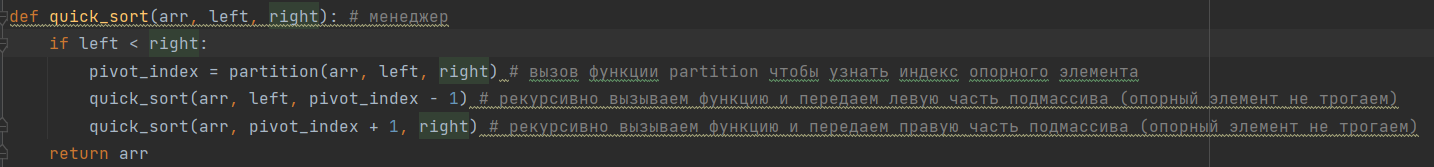
Массив перераспределяется таким образом, что элементы меньше опорного элемента помещаются перед ним, а элементы больше опорного - после него. Опорный элемент становится на свое место.

*Листинг 4. Разделение массива на 2 части*



Рекурсивно повторяется процесс для двух подмассивов слева и справа от опорного элемента.

*Листинг 5. Рекурсивный вызов*



**Описание возникших затруднений**

При выполнении работы затруднений не возникло.

**Тесты**

Тест 1. Генерация случайных списков и их обработка



Тест 2. Ввод корректных данных и их обработка



**Выводы о проделанной работе**

Выполняя данную работу, мы научились работать со списками: считывать их, генерировать, изменять.

**Приложение**

Полный код программы:

**import** random

import math

**def** error\_check(arr): *# функция проверки корректности ввода* **for** i **in** range(len(arr)):  
 **try**:  
 arr[i] = int(arr[i])  
 **except** ValueError:  
 arr[i] = int(random.randint(0, 9))  
  
  
**def** rand(): *# функция генерации списка случайной длины со случайными числами* arr = []  
 **for** i **in** range(random.randint(1, 9)):  
 arr.append(random.randint(0, 9))  
 **return** arr  
  
#n = int(input("Введите количество элементов массива: ")) # ввод количество элементов

#my\_arr = []; # объявление массива

#for i in range (n): # цикл для того чтобы заполнить массив

# my\_arr.append(int(input("Введите элемент массива: "))) # функция append для того чтобы заполнить массив (заполняется с конца)

#print(my\_arr)

arr = [] # объявление массива

arr = input("Введите элементы массива через пробел: ").split()

error\_check(arr1)  
arr = rand()

def quick\_sort(arr, left, right): # менеджер

if left < right:

pivot\_index = partition(arr, left, right) # вызов функции partition чтобы узнать индекс опорного элемента

quick\_sort(arr, left, pivot\_index - 1) # рекурсивно вызываем функцию и передаем левую часть подмассива (опорный элемент не трогаем)

quick\_sort(arr, pivot\_index + 1, right) # рекурсивно вызываем функцию и передаем правую часть подмассива (опорный элемент не трогаем)

Return arr

def partition(arr, left, right):

i = left

j = right - 1

pivot = arr[right] # в качестве опорного элемента берем последний элемент

while i < j:

while i < right and arr[i] < pivot: # i идёт с начала массива и ищет элемент больше опорного,

i += 1 # если на текущей итерации не находит, то увеливается на 1

print("i", i, arr[i])

while j > left and arr[j] >= pivot:

#print("j", j, arr[j])

j -= 1

print("j", j, arr[j])

if i < j:

arr[i], arr[j] = arr[j], arr[i]

if arr[i] > pivot:

arr[i], arr[right] = arr[right], arr[i]

print(arr)

return I

sorted\_arr = quick\_sort(arr, 0, len(arr) - 1)

print("Отсортированный массив:", sorted\_arr)