Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования «Ульяновский государственный технический университет»

Кафедра «Вычислительная техника»

**Отчет по лабораторной работе №1**

Дисциплина: «Разработка профессиональных приложений»

Обработка списков

Вариант № 22

Выполнил:

студент группы ИВТАПбд-21

Старостин К. А.

Проверил:

преподаватель кафедры

«Вычислительная техника»

Исхаков Ильнур Ирекович

Ульяновск, 2023

# Задание по варианту

Необходимо на языке программирования Python реализовать быструю сортировку.

# Требования к программе

1. Ввод элементов списка должен быть доступен с клавиатуры и путем автоматической генерации
2. Программа должна корректно обрабатывать некорректный ввод.
3. Исходный код должен быть откомментирован
4. Необходимо реализовать правильную декомпозицию программы на методы.
5. Необходимо реализовать предложенный алгоритм без использования стандартных функций и с использованием (необходимо реализовать два отдельных метода).

# Описание реализации

Для реализации быстрой сортировки были использованы следующие методы: quick\_sort() – непосредственно сама реализация быстрой сортировки, исходя из требований был создан метод input\_list() – пользовательский ввод списка, generate\_list() – автоматическая реализация списка. Во избежание пересечения методов input\_list и generate\_list была использована следующая конструкция:

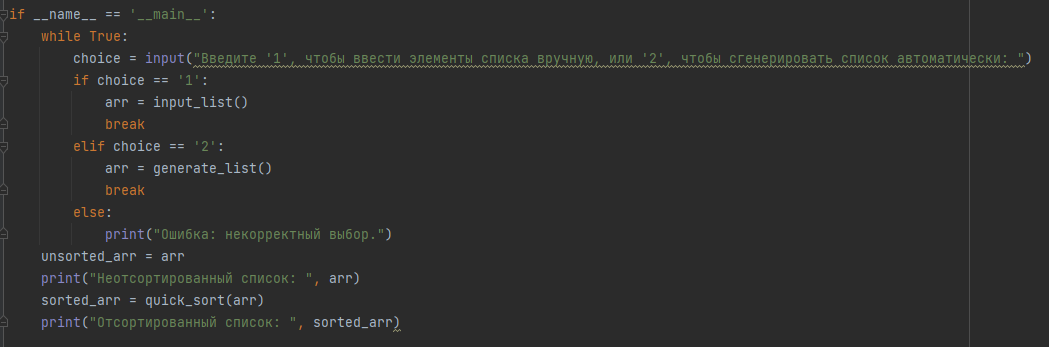


Рисунок 1 Конструкция "Точка входа"

Из-за того, что Python считает точкой входа начало кода, появилась необходимость задать начало самостоятельно, поэтому и была написана вышеприведённая конструкция. Конструкция была необходима для создания выбора ввода массива, а также для того, чтобы методы не пересекались между собой, т.е. не выводились последовательно.

Метод quick\_sort принимает массив, после следует условию: Если длинна массива меньше, либо равна одному, то возвращает массива (массив, состоящий из одного элемента является отсортированным). В противном случае начальный элемент принимается за опорное значение и происходит разделение на два массива: всё, что меньше опорного и всё, что больше опорного. После рекурсивным методом проверяются эти два массива.

Метод input\_list создан для ручного ввода элементов в массив. Для соблюдения требований код метода был обёрнут в try/except. Пользователь вводит размер массива, а после каждый элемент массива отдельно. Ввод элементов реализован с помощью цикла, поэтому после каждого элемента необходимо наживать ENTER. Если введено больше элементов, чем пользователь указал, то выводит ошибку.

Метод generate\_list создан для автоматической генерации рандомного списка. Пользователь вводит размер массива и с помощью конструкции

arr = [random.randint(0, 100) for i in range(n)]

генерируется список из n элементов и в свою очередь элемент должен быть в диапазоне от 0 до 100. Данный код также завёрнут в конструкцию try/except и выводит ошибку некорректного ввода.

С помощью конструкции

if \_\_name\_\_ == '\_\_main\_\_':

задана точка входа. Аналог main() в языке C++. Добавлен выбор, либо ввод ручной, либо автоматический. После выбора вызываются соответствующие методы. После ввода выводится введённый массив, неотсортированный, вызывается метод с быстрой сортировкой и выводится уже отсортированный массив.

Также добавлена библиотечная функция sort(), которая также сортирует массив.

# Описание возникших затруднений

При написании кода затруднений не возникло.

# Описание альтернативных способов решения

Генерацию случайного списка можно было реализовать следующей конструкцией:

# создание списка от 0 до 20

data = [i for i in range(0, 20)]

# перемешивание списка

random.shuffle(data)

В этом случае используется метод shuffle, перемешивание списка.

# Результат работы программы

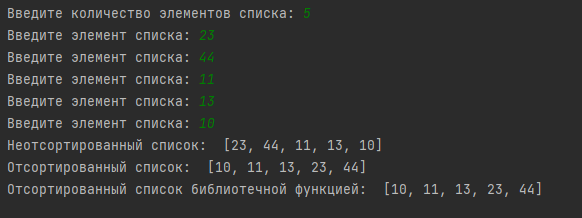


Рисунок 2 Ручной ввод массива

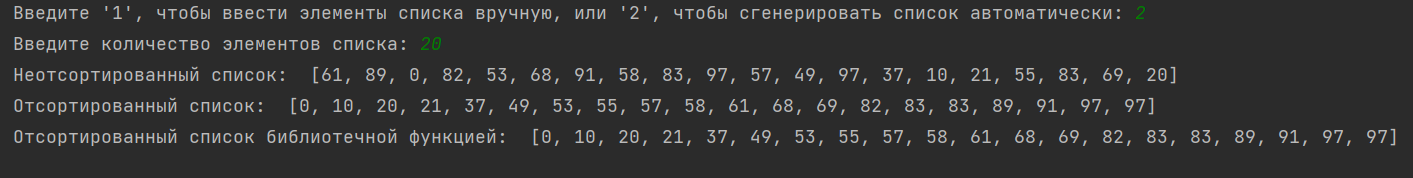


Рисунок 3 Рандомный массив

# Код

import random  
  
# быстрая сортировка списка  
  
  
def quick\_sort(arr):  
 if len(arr) <= 1:  
 return arr  
 else:  
 pivot = arr[0]  
 less = [x for x in arr[1:] if x <= pivot]  
 greater = [x for x in arr[1:] if x > pivot]  
 return quick\_sort(less) + [pivot] + quick\_sort(greater)  
  
  
# ручной ввод списка  
  
  
def input\_list():  
  
 try:  
 n = int(input("Введите количество элементов списка: "))  
 arr = [int(input("Введите элемент списка: ")) for i in range(n)]  
 return arr  
 except ValueError:  
 print("Ошибка: некорректный ввод элементов списка.")  
  
# автоматическая генерация списка  
  
  
def generate\_list():  
 try:  
 n = int(input("Введите количество элементов списка: "))  
 arr = [random.randint(0, 100) for i in range(n)]  
 return arr  
 except ValueError:  
 print("Ошибка: некорректный ввод количества элементов списка.")  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 while True:  
 choice = input("Введите '1', чтобы ввести элементы списка вручную, или '2', чтобы сгенерировать список автоматически: ")  
 if choice == '1':  
 arr = input\_list()  
 break  
 elif choice == '2':  
 arr = generate\_list()  
 break  
 else:  
 print("Ошибка: некорректный выбор.")  
 unsorted\_arr = arr  
 print("Неотсортированный список: ", arr)  
 sorted\_arr = quick\_sort(arr)  
 print("Отсортированный список: ", sorted\_arr)  
 arr.sort()  
 print("Отсортированный список библиотечной функцией: ", arr)