

**МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»**

**(МИИТ)**

|  |
| --- |
| **ИНСТИТУТ ТРАНСПОРТНОЙ ТЕХНИКИ И СИСТЕМ УПРАВЛЕНИЯ**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Кафедра «Управление и защита информации»

**Лабораторная работа №9**

**«Быстрая сортировка»**

**по дисциплине**

**«Методы программирования»**

**Выполнил:** студент группы ТКИ-311

Куминов В. П.

**Проверил:** к.т.н., доц. Логинова Л. Н.,

к.т.н., доц. Сафронов А. И.

**Москва – 2022 г.**

**1. Цель**

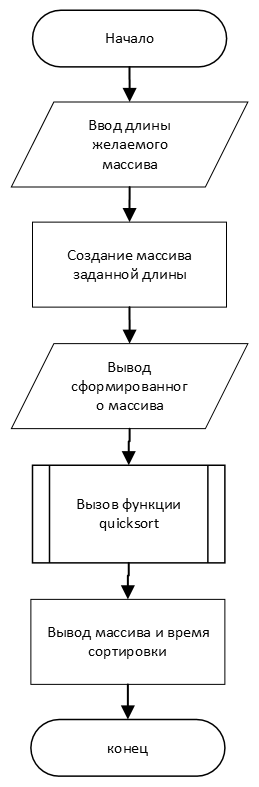
Изучить работу алгоритма быстрой сортировки.

**2. Постановка задачи**

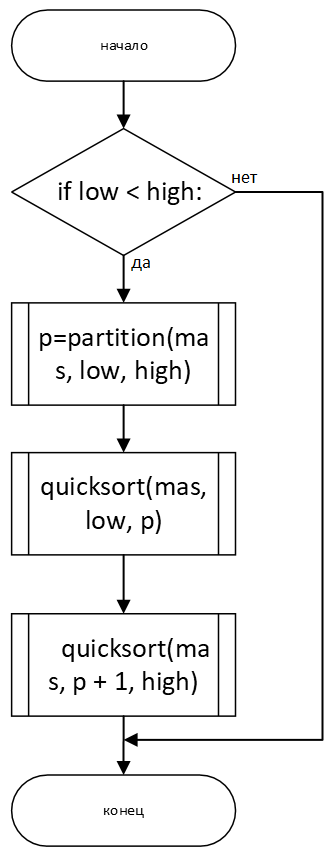
Реализовать на выбранном языке программирования алгоритм быстрой сортировки.

3. Блок-схема алгоритма быстрой сортировки

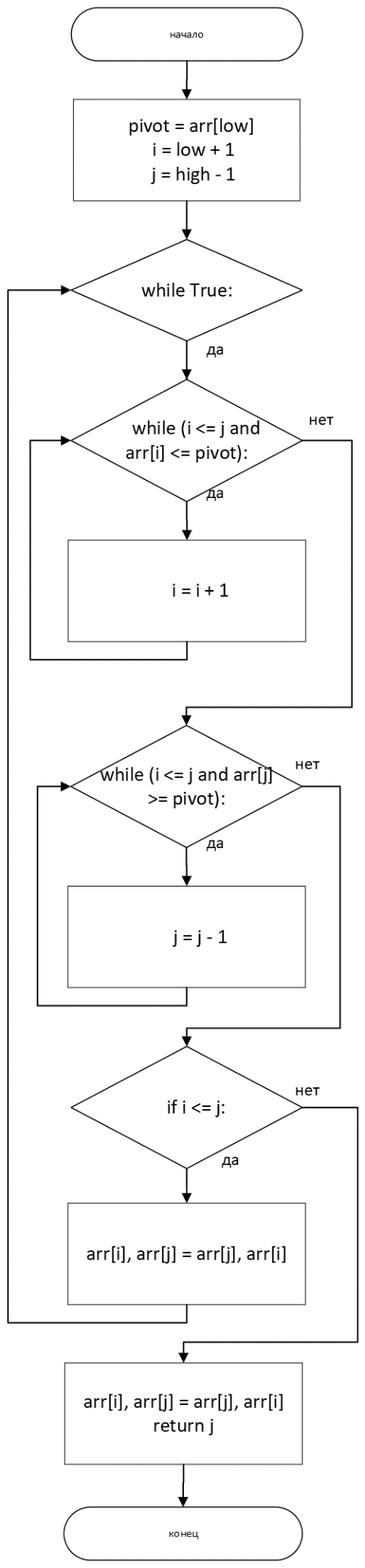
3.1. Блок-схема основной программы



3.2. Блок-схема функции quicksort



3.3. Блок-схема функции partition



**4. Подбор тестовых примеров**

Входные данные: 44, 55, 12, 42, 94, 18, 6, 67

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **44** | 55 | 12 | 42 | 94 | 18 | **6** | 67 |
| i = 1 (44 > 42) |  |  | опорный |  |  | j = 7 (6 < 42) | j = 8 (67 < 42) |
| Первый обмен | | | | | | | |
| 6 | **55** | 12 | 42 | 94 | **18** | 44 | 67 |
|  | i = 2 (55 > 42) |  | опорный |  | j = 6 (18 < 42) |  |  |
| Второй обмен | | | | | | | |
| 6 | 18 | 12 | 42 | 94 | 55 | 44 | 67 |
|  |  | i = 3 (12 < 42) | опорный | j = 5 (94 > 42) |  |  |  |
| Массив разделён относительно опорного элемента | | | | | | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 6 | 18 | 12 |  | 42 | 94 | 55 | 44 | 67 |
| i = 1 (6 < 18) | i = 2 (18 = 18) | j = 3 (12 <18) | i = 1 (42 < 55) | i = 2 (94 > 55) | опорный | j = 5 (67 > 55) | j = 5 (67 > 55) |
| Третий обмен | | | | | | | | |
| 6 | 12 | 18 |  | 42 | 44 | 55 | 94 | 67 |
|  |  |  |  |  |  | i = 1 (94 = 94) | j = 1 (67 < 94) |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 6 | 12 | 18 | 42 | 44 | 55 | 67 | 94 |

**5. Код программы**

import random

import time

def quicksort(mas, low, high):

if low < high:

p = partition(mas, low, high)

quicksort(mas, low, p)

quicksort(mas, p + 1, high)

def partition(arr, low, high):

pivot = arr[low]

i = low + 1

j = high - 1

while True:

while (i <= j and arr[i] <= pivot):

i = i + 1

while (i <= j and arr[j] >= pivot):

j = j - 1

if i <= j:

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

else:

arr[low], arr[j] = arr[j], arr[low]

return j

n = int(input('Введите длину массива: '))

Array = [random.randint(10, 99) for i in range(n)]

print('Исходный массив: ', Array)

quicksort(Array, 0, len(Array))

print("Время сортировки: ", time.process\_time(),"\n")

print(Array)

**6. Проверка тестовых примеров на ПК**

Изображение выглядит как текст

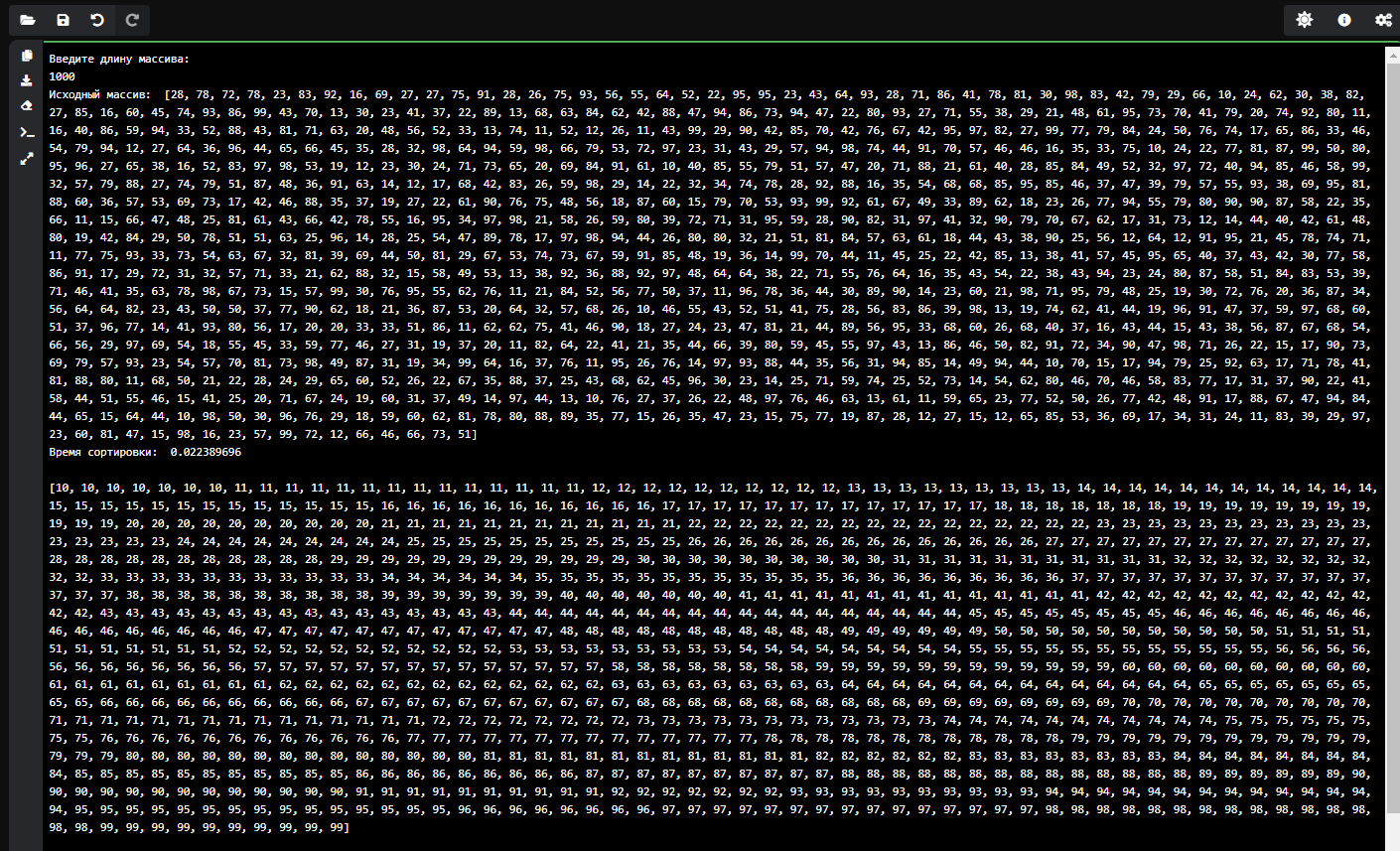
Автоматически созданное описание

10 элементов

Изображение выглядит как текст, внутренний, экран, снимок экрана

Автоматически созданное описание

100 элементов



1000 элементов

Таблица 1 – Анализ всех реализованных сортировок

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Количество сортируемых элем. | Быстрая сортировка  (t) мс | Пирамидальная  Сортировка (t) мс | Сортировка вставкой (t) мс | Сортировка  Слиянием (t) мс |
| 10 | 0,0311369 | 0,017059155 | 0,0251427 | 0,02973772 |
| 100 | 0,0188327 | 0,037608229 | 0,0228232 | 0,02937872 |
| 1000 | 0,0223889 | 0,051505519 | 0,1173644 | 0,03523212 |

**7. Вывод**

В ходе выполнения работы - изучил методы работы алгоритма быстрой сортировки. Алгоритм быстрой сортировки имеет самое быстрое время выполнения, данный вывод можно сделать исходя из таблицы 1. Данные о времени работы алгоритмов сортировки «вставкой», сортировки «слиянием», пирамидальной сортировки были получены из предыдущих лабораторных работ.