



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Московский государственный технический университет имени
Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчет по лабораторной работе №3 по курсу "Анализ алгоритмов"

Тема Трудоёмкость сортировок

Студент Козлова И.В.

Группа ИУ7-52Б

Оценка (баллы) _____

Преподаватель Волкова Л.Л.

Оглавление

Введение	3
1 Аналитическая часть	5
1.1 Сортировка перемешиванием	5
1.2 Сортировка вставками	5
1.3 Сортировка выбором	6
2 Конструкторская часть	7
2.1 Разработка алгоритмов	7
2.2 Модель вычислений	10
2.3 Трудоёмкость алгоритм	10
2.3.1 Алгоритм сортировки перемешиванием	10
2.3.2 Алгоритм сортировки вставками	10
2.3.3 Алгоритм сортировки выбором	10
3 Технологическая часть	11
3.1 Требования к ПО	11
3.2 Средства реализации	11
3.3 Сведения о модулях программы	11
3.4 Листинг кода	11
3.5 Функциональные тесты	13
4 Исследовательская часть	14
4.1 Технические характеристики	14
4.2 Демонстрация работы программы	14
4.3 Время выполнения алгоритмов	14
4.4 Использование памяти	15
Заключение	16
Литература	17

Введение

Одной из важнейших процедур обработки структурированной информации является сортировка.

Сортировка - это процесс перегруппировки заданной последовательности (кортежа) объектов в некотором определенном порядке. Такой определенный порядок позволяет, в некоторых случаях, эффективнее и удобнее работать с заданной последовательностью. В частности, одной из целей сортировки является облегчение задачи поиска элемента в отсортированном множестве.

Алгоритм сортировки используются практически в любой программной системе. Целью алгоритмов сортировки является упорядочение последовательности элементов данных. Поиск элемента в последовательности отсортированных данных занимает время, пропорциональное логарифму количеству элементов в последовательности, а поиск элемента в последовательности не отсортированных данных занимает время, пропорциональное количеству элементов в последовательности, то есть намного больше. Существует множество различных методов сортировки данных. Однако любой алгоритм сортировки можно разбить на три основные части:

- сравнение, определяющее упорядочность пары элементов;
- перестановка, меняющая местами пару элементов;
- собственно сортирующий алгоритм, который осуществляет сравнение и перестановку элементов данных до тех пор, пока все эти элементы не будут упорядочены.

Одной из важнейшей характеристик любого алгоритма сортировки является скорость его работы, которая определяется функциональной зависимостью среднего времени сортировки последовательностей элементов данных, определенной длины, от этой длины.

Задачами данной лабораторной:

1. изучить и реализовать три алгоритма сортировки: шейкер, вставками, выбором;

2. провести сравнительный анализ трудоемкости алгоритмов на основе теоретических расчетов и выбранной модели вычислений;
3. провести сравнительный анализ алгоритмов на основе экспериментальных данных, а именно по времени;
4. описать и обосновать полученные результаты в отчете о выполненной лабораторной работе, выполненного как расчётно-пояснительная записка к работе.

1 Аналитическая часть

В этом разделе будут представлены описания алгоритмов сортировки перемешиванием, вставками и выбором.

1.1 Сортировка перемешиванием

Сортировка перемешиванием [1] — это разновидность сортировки пузырьком. Отличие в том, что данная сортировка в рамках одной итерации проходит по массиву в обоих направлениях (слева направо и справа налево), тогда как сортировка пузырьком — только в одном направлении (слева направо).

Общие идеи алгоритма:

- обход массива слева направо, аналогично пузырьковой — сравнение соседних элементов, меняя их местами, если левое значение больше правого;
- обход массива в обратном направлении (справа налево), начиная с элемента, который находится перед последним отсортированным, то есть на этом этапе элементы также сравниваются между собой и меняются местами, чтобы наименьшее значение всегда было слева.

1.2 Сортировка вставками

Сортировка вставками [2] — алгоритм сортировки, которым элементы входной последовательности просматриваются по одному, и каждый новый поступивший элемент размещается в подходящее место среди ранее упорядоченных элементов.

В начальный момент отсортированная последовательность пуста. На каждом шаге алгоритма выбирается один из элементов входных данных и помещается на нужную позицию в уже отсортированной последовательности до тех пор, пока набор входных данных не будет исчерпан. В любой

момент времени в отсортированной последовательности элементы удовлетворяют требованиям к входным данным алгоритма.

1.3 Сортировка выбором

Сортировка выбором [3] - алгоритм, основанный на сравнение каждого элемента с каждым, и в случае необходимости производя обмен.

Шаги алгоритма:

1. находим номер минимального значения в текущем массиве;
2. производим обмен этого значения со значением первой неотсортированной позиции (обмен не нужен, если минимальный элемент уже находится на данной позиции);
3. теперь сортируем “хвост” массива, исключив из рассмотрения уже отсортированные элементы.

Для реализации устойчивости алгоритма необходимо в пункте 2 минимальный элемент непосредственно вставлять в первую неотсортированную позицию, не меняя порядок остальных элементов, что может привести к резкому увеличению числа обменов.

Вывод

В данной работе стоит задача реализации 3 алгоритмов сортировки, а именно: перемешиванием, вставками и выбором. Необходимо оценить теоретическую оценку алгоритмов и проверить ее экспериментально.

2 Конструкторская часть

В этом разделе будут приведены схемы алгоритмов и вычисления трудоемкости данных алгоритмов.

2.1 Разработка алгоритмов

На рисунках 2.1, 2.2 и 2.3 представлены схемы алгоритмов сортировки пузырьком, выбором и вставками соответственно.

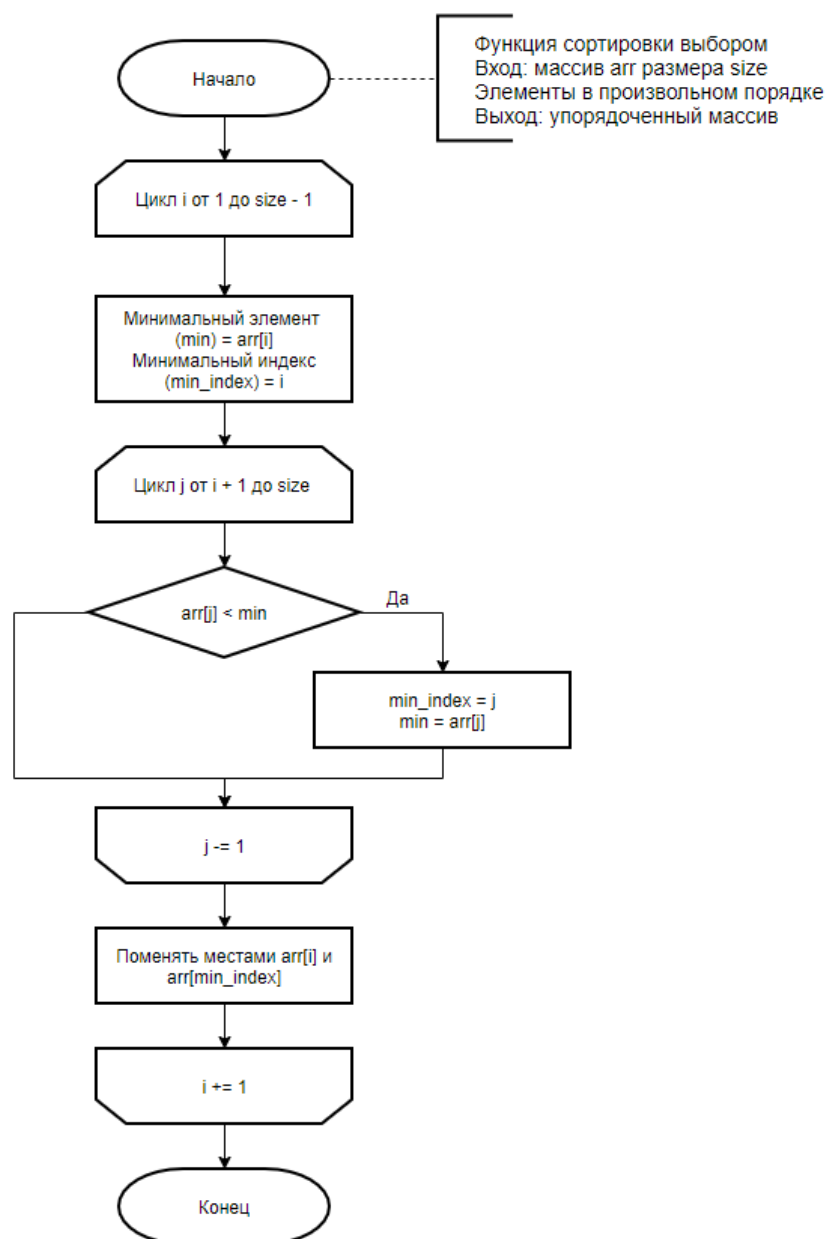


Рисунок 2.1 – Схема алгоритма сортировки выбором

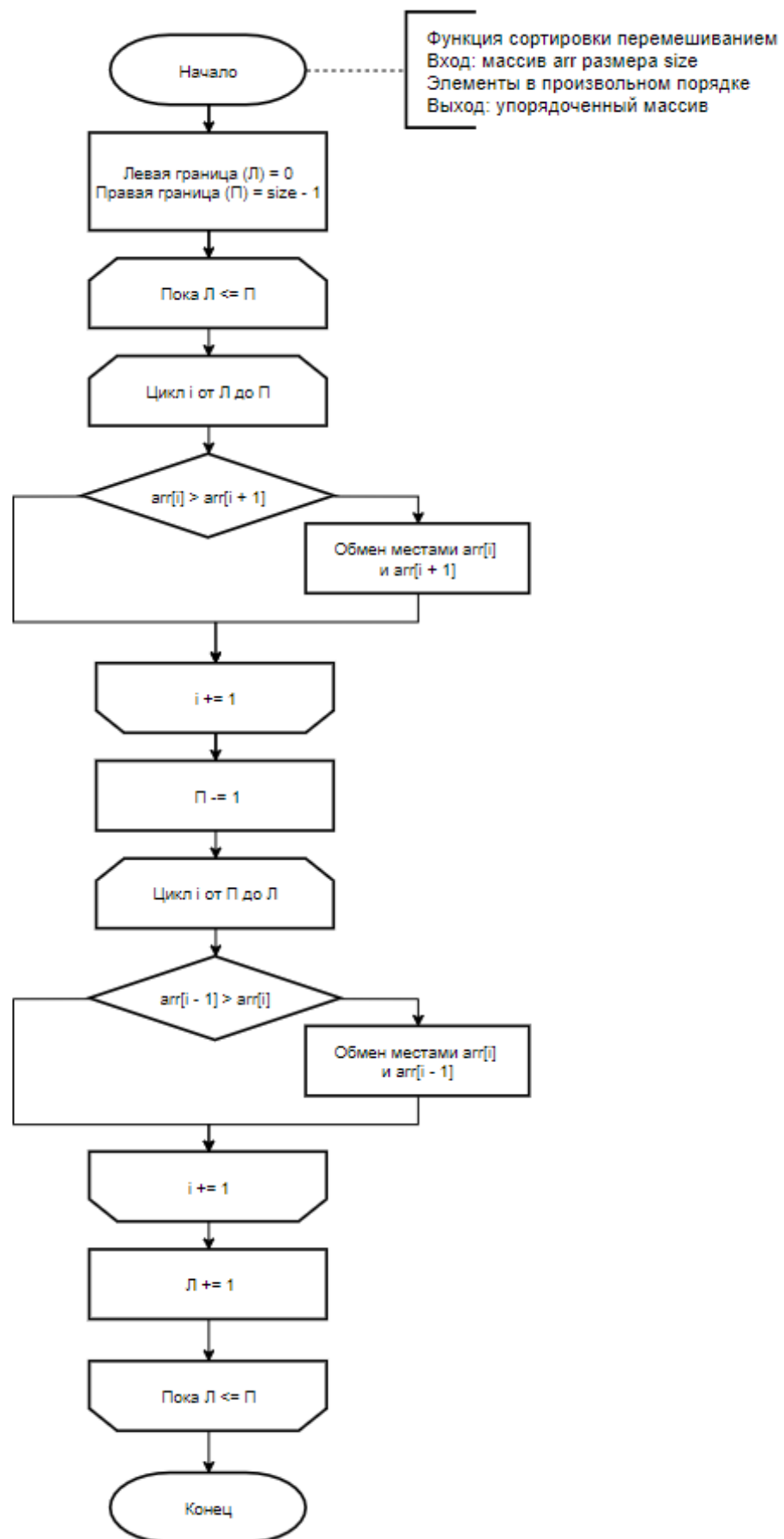


Рисунок 2.2 – Схема алгоритма сортировки перемешиванием

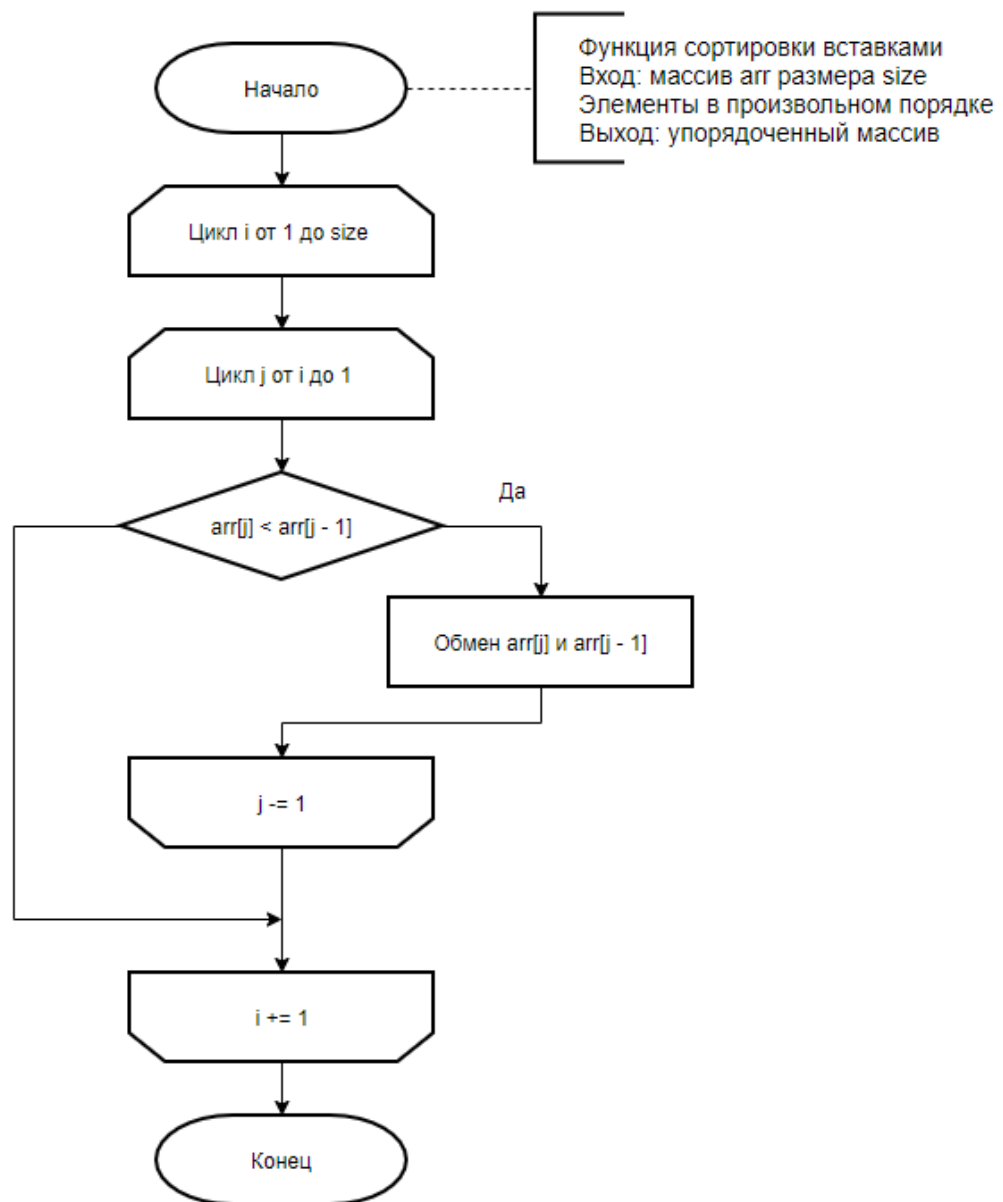


Рисунок 2.3 – Схема алгоритма сортировки вставками

2.2 Модель вычислений

2.3 Трудоёмкость алгоритм

2.3.1 Алгоритм сортировки перемешиванием

2.3.2 Алгоритм сортировки вставками

2.3.3 Алгоритм сортировки выбором

Вывод

3 Технологическая часть

В данном разделе будут приведены требования к программному обеспечению, средства реализации и листинги кода.

3.1 Требования к ПО

К программе предъявляется ряд требований:

- на вход подаётся массив сравнимых элементов (целые числа);
- на выходе — тот же массив, но в отсортированном порядке.

3.2 Средства реализации

В качестве языка программирования для реализации данной лабораторной работы был выбран ЯП Python [4].

Данный язык достаточно удобен и гибок в использовании.

Время работы алгоритмов было замерено с помощью функции `process_time()` из библиотеки `time` [5]

3.3 Сведения о модулях программы

Программа состоит из двух модулей:

1. `main.py` - главный файл программы, в котором располагаются коды всех алгоритмов и меню;
2. `test.py` - файл с замерами времени для графического изображения результата.

3.4 Листинг кода

Листинг 3.1 – Алгоритм сортировки выбором

```
1 def selection_sort(arr):
2     for i in range(0, len(arr) - 1):
3         min = i
4         for j in range(i + 1, len(arr)):
5             if arr[j] < arr[min]:
6                 min = j
7         arr[i], arr[min] = arr[min], arr[i]
8     return arr
```

Листинг 3.2 – Алгоритм сортировки вставками

```
1 def insertion_sort(arr):
2     for i in range(1, len(arr)):
3         j = i - 1
4         key = arr[i]
5
6         while j >= 0 and arr[j] > key:
7             arr[j + 1] = arr[j]
8             j -= 1
9         arr[j + 1] = key
10    return arr
```

Листинг 3.3 – Алгоритм сортировки перемешиванием (или сортировка шейкер)

```
1 def shaker_sort(arr):
2     length = len(arr)
3     swapped = True
4     start_index = 0
5     end_index = length - 1
6
7     while (swapped == True):
8         swapped = False
9
10        for i in range(start_index, end_index):
11            if (arr[i] > arr[i + 1]):
12                arr[i], arr[i + 1] = arr[i + 1], arr[i]
13                swapped = True
14
15        if (not (swapped)):
16            break
17
```

```

18         swapped = False
19         end_index = end_index - 1
20
21         for i in range(end_index - 1, start_index - 1, -1):
22             if (arr[i] > arr[i + 1]):
23                 arr[i], arr[i + 1] = arr[i + 1], arr[i]
24                 swapped = True
25
26         start_index = start_index + 1
27     return arr

```

3.5 Функциональные тесты

В таблице 3.1 приведены тесты для функций, реализующих алгоритмы сортировки. Тесты пройдены успешно.

Таблица 3.1 – Функциональные тесты

Входной массив	Ожидаемый результат	Результат
[1, 2, 3, 4]	[1, 2, 3, 4]	[1, 2, 3, 4]
[5, 4, 3, 2, 1]	[1, 2, 3, 4, 5]	[1, 2, 3, 4, 5]
[3, 2, -5, 0, 1]	[-5, 0, 0, 2, 3]	[-5, 0, 0, 2, 3]
[4]	[4]	[4]
[]	[]	[]

Вывод

Были разработаны схемы всех трех алгоритмов сортировки. Для каждого из них были рассчитаны и оценены лучшие и худшие случаи.

4 Исследовательская часть

В данном разделе будут приведены примеры работы программ, постановка эксперимента и сравнительный анализ алгоритмов на основе полученных данных.

4.1 Технические характеристики

Технические характеристики устройства, на котором выполнялось тестирование:

- Операционная система: Ubuntu 20.04.3 [6] Linux [7] x86_64.
- Память: 8 GiB.
- Процессор: 11th Gen Intel® Core™ i5-1135G7 @ 2.40GHz [8].

Тестирование проводилось на ноутбуке, включенном в сеть электропитания. Во время тестирования ноутбук был нагружен только встроенными приложениями окружения, а также непосредственно системой тестирования.

4.2 Демонстрация работы программы

На рисунке 4.1 представлен результат работы программы.

4.3 Время выполнения алгоритмов

Алгоритмы тестировались при помощи функции `process_time()` из библиотеки `time` языка Python. Данная функция всегда возвращает значения времени, а именно сумму системного и пользовательского процессорного времени текущего процессора, типа `float` в секундах.

Контрольная точка возвращаемого значения не определена, поэтому допустима только разница между результатами последовательных вызовов.

МЕНЮ:

- 0. Выход
 - 1. Сортировка Шейкер
 - 2. Сортировка вставками
 - 3. Сортировка выбором
 - 4. Замер времени (длина слов от 1 до 10)
- Выбор: 4

Введите массив: 4 5 9 0 23 2 -4 1

Массив отсортированный сортировкой-Шейкер [-4, 0, 1, 2, 4, 5, 9, 23]

Время в мкс (сортировкой-Шейкер) 6.71875

Массив отсортированный сортировкой вставками [-4, 0, 1, 2, 4, 5, 9, 23]

Время в мкс (сортировкой вставками) 5.15625

Массив отсортированный сортировкой выбором [-4, 0, 1, 2, 4, 5, 9, 23]

Время в мкс (сортировкой выбором) 5.9375

Рисунок 4.1 – Пример работы программы

Таблица 4.1 – Результаты замеров времени.

Вывод

Заключение

В ходе выполнения лабораторной работы были решены следующие задачи:

- изучены и реализованы 3 алгоритма сортировки: перемешиванием, вставками, выбором;
- проведен сравнительный анализ трудоёмкости алгоритмов на основе теоретических расчетов и выбранной модели вычислений;
- проведен сравнительный анализ алгоритмов на основе экспериментальных данных;
- подготовлен отчет о лабораторной работе.

Литература

- [1] Шейкерная сортировка (перемешиванием) [Электронный ресурс]. Режим доступа: <https://kvodo.ru/shaker-sort.html> (дата обращения: 12.09.2021).
- [2] Сортировка вставками [Электронный ресурс]. Режим доступа: <https://kvodo.ru/sortirovka-vstavkami-2.html> (дата обращения: 12.09.2021).
- [3] Сортировка выбором [Электронный ресурс]. Режим доступа: <https://kvodo.ru/sortirovka-vyiborom-2.html> (дата обращения: 12.09.2021).
- [4] Welcome to Python [Электронный ресурс]. Режим доступа: <https://www.python.org> (дата обращения: 04.09.2021).
- [5] time — Time access and conversions [Электронный ресурс]. Режим доступа: <https://docs.python.org/3/library/time.html#functions> (дата обращения: 04.09.2021).
- [6] Ubuntu 20.04.3 LTS (Focal Fossa) [Электронный ресурс]. Режим доступа: <https://releases.ubuntu.com/20.04/> (дата обращения: 04.09.2021).
- [7] Linux – Википедия [Электронный ресурс]. Режим доступа: <https://ru.wikipedia.org/wiki/Linux> (дата обращения: 04.09.2021).
- [8] Процессор Intel® Core™ i5-1135G7 [Электронный ресурс]. Режим доступа: <https://www.intel.ru/content/www/ru/ru/products/sku/208658/intel-core-i51135g7-processor-8m-cache-up-to-4-20-ghz/specifications.html> (дата обращения: 04.09.2021).