САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе № 1

по курсу «Алгоритмы и структуры данных»

Тема: Сортировка вставками, выбором, пузырьковая

Вариант 2

Выполнила:

Азизова Наида Элимизовна

К3140

Проверил(а):

Афанасьев А. В.

Санкт-Петербург

2024 г.

Содержание отчёта

[Задания по варианту 3](#_Toc180189099)

[Задача №1. Сортировка вставкой 3](#_Toc180189100)

[Задача №2. Сортировка вставкой + 5](#_Toc180189101)

[Задача №5. Сортировка выбором 7](#_Toc180189102)

[Дополнительные задания 9](#_Toc180189103)

[Задача №3. Сортировка вставкой по убыванию 9](#_Toc180189104)

[Задача №4. Линейный поиск 11](#_Toc180189105)

[Задача №6. Пузырьковая сортировка 13](#_Toc180189106)

[Вывод 15](#_Toc180189107)

# Задания по варианту

## Задача №1. Сортировка вставкой

Изображение выглядит как текст, снимок экрана, Шрифт, документ

Автоматически созданное описание

Решение:

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Текстовое объяснение решения:

1. Функция insertion\_sort принимает 2 параметра: n – число элементов в массиве и spis – массив элементов. Затем используем сортировку вставкой: если правый элемент меньше левого, то меняем их местами и точно также сo значениями переменных i и p (индексы данных элементов). После этого функция возвращает отсортированный список;
2. Импортируем модули time и tracemalloc;
3. Открываем файл input.txt, в котором содержатся входные данные. Считываем строку и приводим число n к целочисленному виду. Считываем следующую строку и преобразовываем ее в список целых чисел (mas). И закрываем файл;
4. Создаем переменную (itog\_spis), в которой выполняется функция insertion\_sort(n, mas);
5. Открываем файл output.txt и записываем в него полученный результат (отсортированный массив). И закрываем файл;
6. В конце выводим время (из начального времени вычитаем конечное) и затраченную память.

Результат работы кода на примерах из текста задачи:

Ввод:

Изображение выглядит как текст, Шрифт, снимок экрана, число

Автоматически созданное описание

Вывод:

Изображение выглядит как текст, Шрифт, снимок экрана, число

Автоматически созданное описание

Время выполнения и затраченная память:

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Вывод по задаче: Я научилась осуществлять сортировку вставкой массива целых чисел и узнала время ее выполнения и затраты памяти.

## Задача №2. Сортировка вставкой +

Изображение выглядит как текст, снимок экрана, Шрифт, документ

Автоматически созданное описание

Решение:

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Текстовое объяснение решения:

1. Функция insertion\_sort принимает 2 параметра: n – число элементов в массиве и spis – массив элементов. Затем используем сортировку вставкой: если правый элемент меньше левого, то меняем их местами. Также создаем список indexх (в него будем записывать новые индексы определенных элементов). В этом списке пока что только один элемент – это единица. После этого функция возвращает отсортированный список и список индексов;
2. Импортируем модули time и tracemalloc;
3. Открываем файл input.txt, в котором содержатся входные данные. Считываем строку и приводим число n к целочисленному виду. Считываем следующую строку и преобразовываем ее в список целых чисел (mas). И закрываем файл;
4. Создаем переменные (itog\_spis и itog\_ind), в которых выполняется функция insertion\_sort(n, mas);
5. Открываем файл output.txt и записываем в него полученный результат (элементы списка indexх в первой строчке, а отсортированный массив чисел — во второй). И закрываем файл;
6. В конце выводим время (из начального времени вычитаем конечное) и затраченную память.

Результат работы кода на примерах задачи:

Ввод:

Изображение выглядит как текст, Шрифт, снимок экрана, число

Автоматически созданное описание

Вывод:

Изображение выглядит как текст, Шрифт, снимок экрана, число

Автоматически созданное описание

Время выполнения и затраченная память:

Изображение выглядит как текст, Шрифт, снимок экрана, Графика

Автоматически созданное описание

Вывод по задаче: Я научилась осуществлять сортировку вставкой массива целых чисел и писать алгоритм по сохранению данных о новых индексах элементов.

## Задача №5. Сортировка выбором

Изображение выглядит как текст, Шрифт, снимок экрана, документ

Автоматически созданное описание

Решение:

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Текстовое объяснение решения:

1. Функция selection\_sort принимает 2 параметра: n – число элементов в массиве и spis – массив элементов. Затем используем сортировку выбором: сначала сравниваем элементы и сохраняем минимальный из них в переменную mini, а индекс минимального элемента сохраняем в переменной indexx. Метод pop удаляет и возвращает элемент из списка по его индексу в indexx, а метод insert добавляет новый элемент в любое место списка. Метод принимает два параметра: индекс (i), по которому будет вставлен элемент, и сам элемент (mini). После этого функция возвращает отсортированный список.
2. Импортируем модули time и tracemalloc;
3. Открываем файл input.txt, в котором содержатся входные данные. Считываем строку и приводим число n к целочисленному виду. Считываем следующую строку и преобразовываем ее в список целых чисел (mas). И закрываем файл;
4. Создаем переменную (itog\_spis), в которой выполняется функция selection\_sort(n, mas);
5. Открываем файл output.txt и записываем в него полученный результат (отсортированный массив чисел). И закрываем файл;
6. В конце выводим время (из начального времени вычитаем конечное) и затраченную память.

Результат работы кода на примерах задачи:

Ввод:

Изображение выглядит как текст, Шрифт, снимок экрана, число

Автоматически созданное описание

Вывод:

Изображение выглядит как текст, Шрифт, снимок экрана, число

Автоматически созданное описание

Время выполнения и затраченная память:

Изображение выглядит как текст, Шрифт, снимок экрана, Графика

Автоматически созданное описание

Вывод по задаче: Я научилась осуществлять сортировку выбором массива целых чисел и узнала время ее выполнения и затраты памяти.

# Дополнительные задания

## Задача №3. Сортировка вставкой по убыванию

Изображение выглядит как текст, Шрифт

Автоматически созданное описание

Решение:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Текстовое объяснение решения:

1. Функция swap принимает два параметра: а и b – целые числа. Вводим переменную temp, которая поможет нам поменять значения а и b местами. Затем меняем нашу функцию insertion\_sort так, что мы сдвигаем числа, которые больше, влево, используя функцию swap. После этого алгоритма функция возвращает отсортированный список.
2. Импортируем модули time и tracemalloc;
3. Открываем файл input.txt, в котором содержатся входные данные. Считываем строку и приводим число n к целочисленному виду. Считываем следующую строку и преобразовываем ее в список целых чисел (mas). И закрываем файл;
4. Создаем переменную (itog\_spis), в которой выполняется функция insertion\_sort(n, mas);
5. Открываем файл output.txt и записываем в него полученный результат (отсортированный массив чисел). И закрываем файл;
6. В конце выводим время (из начального времени вычитаем конечное) и затраченную память.

Результат работы кода на примерах задачи:

Ввод:

Изображение выглядит как текст, Шрифт, снимок экрана, число

Автоматически созданное описание

Вывод:

Изображение выглядит как текст, Шрифт, снимок экрана, число

Автоматически созданное описание

Время выполнения и затраченная память:

Изображение выглядит как текст, Шрифт, снимок экрана, типография

Автоматически созданное описание

Вывод по задаче: Я научилась осуществлять сортировку вставкой по убыванию массива целых чисел с помощью функции swap и узнала время ее выполнения и затраты памяти.

## Задача №4. Линейный поиск

Изображение выглядит как текст, Шрифт, снимок экрана, белый

Автоматически созданное описание

Решение:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Текстовое объяснение решения:

1. Функция linear\_search принимает два параметра: а – массив элементов, v – искомый элемент. Затем выполняем поиск v в массиве а: сравниваем каждый элемент массива с v. Если они равны, то добавляем i (индекс элемента, который мы нашли) в список res. После этого мы сравниваем длину списка res (количество элементов, равных v): если нашелся один элемент, равный v, то возвращаем его индекс; если таких элементов не нашлось, то возвращаем -1; если таких элементов несколько, то возвращаем их количество и список их индексов.
2. Импортируем модули time и tracemalloc;
3. Открываем файл input.txt, в котором содержатся входные данные. Считываем первую строку а. Считываем следующую строку (наш искомый элемент v. И закрываем файл;
4. Открываем файл output.txt и записываем в него полученный результат программы. И закрываем файл;
5. В конце выводим время (из начального времени вычитаем конечное) и затраченную память.

Результат работы кода на собственных примерах:

Ввод:

Изображение выглядит как текст, Шрифт, снимок экрана, число

Автоматически созданное описание

Вывод:

Изображение выглядит как текст, Шрифт, снимок экрана, число

Автоматически созданное описание

Время выполнения и затраченная память:

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Вывод по задаче: Я научилась осуществлять линейный поиск определенного элемента среди массива элементов и узнала время его выполнения и затраты памяти.

## Задача №6. Пузырьковая сортировка

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Решение:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Текстовое объяснение решения:

1. Опишем основную функцию - bubble\_sort. На вход она принимает два параметра: n - число элементов массива и spis - массив элементов. Затем производим пузырьковую сортировку данного массива: попарно сравниваем элементы и в случае, если правый меньше левого, меняем их местами. После выполнения этого алгоритма функция возвращает отсортированный список.
2. Для доказательства корректной работы алгоритма сортировки создадим функцию check, которая будет попарно сравнивать числа уже отсортированного массива. В случае, если число с меньшим индексом окажется больше числа с большим индексом, функция вернет False, иначе - True.
3. Открываем файл input.txt, содержащий входные данные. Читаем первую строку и приводим к целочисленному виду полученное число (n). Далее читаем вторую строку и преобразуем ее в список целых чисел. Закрываем файл.
4. Открываем файл output.txt и записываем в него полученный результат программы (отсортированный массив spis), после чего закрываем файл.
5. С помощью функции check проверяем, что массив отсортирован верно.

Результат работы кода на примерах из задачи:

Ввод:

Изображение выглядит как текст, Шрифт, снимок экрана, число

Автоматически созданное описание

Вывод:

Изображение выглядит как текст, Шрифт, снимок экрана, число

Автоматически созданное описание

Время выполнения и затраченная память:

Изображение выглядит как текст, Шрифт, снимок экрана, Графика

Автоматически созданное описание

Вывод по задаче: Я научилась осуществлять пузырьковую сортировку массива целых чисел и выяснила время его выполнения и затраты памяти.

# Вывод

В этой лабораторной работе я научилась осуществлять сортировку вставкой, сортировку вставкой по убыванию, сортировку выбором, пузырьковую сортировку и линейный поиск. Также я проверила время работы алгоритмов и память, которая тратится на их выполнение.