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 8

Выполнил:
Журбина Марина Андреевна
Группа К3139

Проверил:
Афанасьев А. В.

Санкт-Петербург
2024 г.

Содержание отчета

Оглавление

Задачи по варианту 8

Задание № 1. Сортировка вставкой.

Текст задачи:

1 задача. Сортировка вставкой

Используя код процедуры Insertion-sort, напишите программу и проверьте сортировку массива $A = \{31, 41, 59, 26, 41, 58\}$.

- **Формат входного файла (input.txt).** В первой строке входного файла содержится число n ($1 \leq n \leq 10^3$) — число элементов в массиве. Во второй строке находятся n различных целых чисел, по модулю не превосходящих 10^9.
- **Формат выходного файла (output.txt).** Одна строка выходного файла с отсортированным массивом. Между любыми двумя числами должен стоять ровно один пробел.
- Ограничение по времени. 2сек.
- Ограничение по памяти. 256 мб.

Выберите любой набор данных, подходящих по формату, и протестируйте алгоритм.

Код программы:

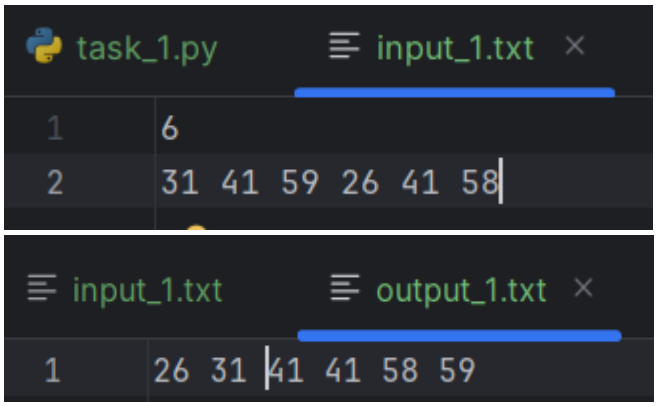
```
import time

t_start = time.perf_counter()
_min = - 10 ** 9
_max = 10 ** 9
n_min = 1
n_max = 10 ** 3
f = open('input_1.txt')
n = int(f.readline())
if n_min <= n <= n_max:
    s = list(map(int, f.readline().split()))
    flag = True
    for el in s:
        if not (_min < el < _max):
            flag = False
    if flag:
        for i in range(1, len(s)):
            a = s[i]
            j = i - 1
            while j >= 0 and a < s[j]:
                s[j + 1] = s[j]
                j = j - 1
            s[j + 1] = a
        f_output = open('output_1.txt', 'w')
        f_output.write(' '.join(map(str, s)))
        t_stop = time.perf_counter()
        if t_stop - t_start <= 2:
            print("Время выполнения:", t_stop - t_start, 'секунд')
        else:
            print("Превышено время выполнения:", t_stop - t_start,
'секунд')
    else:
        f_output = open('output_1.txt', 'w')
        f_output.write('Введены некоректные данные')
else:
    f_output = open('output_1.txt', 'w')
```

```
    f_output.write('Введены некоректные данные')
f.close()
f_output.close()
```

Текстовое объяснение решения:

1) Импорт модуля time для отслеживания времени работы программы.
2) Обьявление переменной счетчика времени.
3) Заданы левая и правая границы интервала допустимых значений для n и для чисел массива.
4) Из файла считывается n и проверяется на соответствие интервалу.
5) Из файла считывается массив и каждый элемент проверяется на соответствие интервалу.
6) Создается цикл for, который проходит со второго элемента и до конца. Слева от элемента находится отсортированная часть массива. Элемент сравнивается с элементами слева до тех пор, пока не встанет на свое место.
7) Отсортированный массив записывается в файл.
8) Выводится время работы алгоритма.

Примеры работы кода:



	Время выполнения
Нижняя граница диапазона значений входных данных из текста задачи	Время выполнения: 0.0013404000019363593 секунд
Пример из задачи	Время выполнения: 0.0013739999994868413 секунд

Верхняя граница диапазона значений входных данных из текста задачи	Время выполнения: 0.040074799999274546 секунд

Вывод по задаче:
В решении задачи использованы переменные, операторы сравнения и if-else конструкции, цикл for, работа с файлами. Код выполняет поставленную задачу.

Задание №3. Сортировка вставкой по убыванию.

Текст задачи:

3 задача. Сортировка вставкой по убыванию

Перепишите процедуру Insertion-sort для сортировки в невозрастающем порядке вместо неубывающего с использованием процедуры `Swap`.

Формат входного и выходного файла и ограничения - как в задаче 1.

Подумайте, можно ли переписать алгоритм сортировки вставкой с использованием рекурсии?

Код:

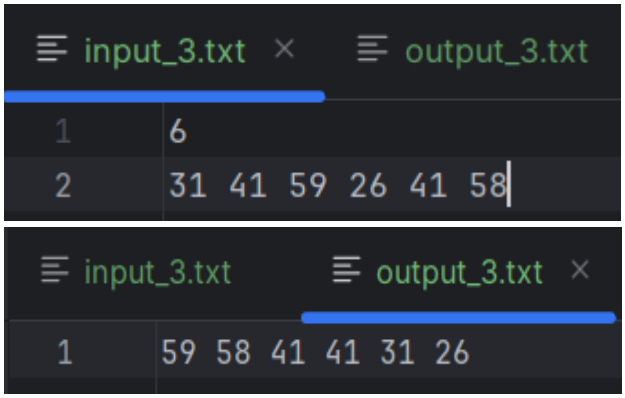
```
import time

t_start = time.perf_counter()
_min = - 10 ** 9
_max = 10 ** 9
n_min = 1
n_max = 10 ** 3
f = open('input_3.txt')
n = int(f.readline())
if n_min <= n <= n_max:
    s = list(map(int, f.readline().split()))
    flag = True
    for el in s:
        if not (_min < el < _max):
            flag = False
    if flag:
        for i in range(1, len(s)):
            a = s[i]
            j = i - 1
            while j >= 0 and a > s[j]:
                s[j + 1], s[j] = s[j], s[j + 1]
                j = j - 1
        f_output = open('output_3.txt', 'w')
        f_output.write(' '.join(map(str, s)))
        t_stop = time.perf_counter()
        if t_stop - t_start <= 2:
            print("Время выполнения:", t_stop - t_start, 'секунд')
        else:
            print("Превышено время выполнения:", t_stop - t_start,
'секунд')
    else:
        f_output = open('output_3.txt', 'w')
        f_output.write('Введены некоректные данные')
else:
    f_output = open('output_3.txt', 'w')
```

```
    f_output.write('Введены некоректные данные')
f.close()
f_output.close()
```

Текстовое объяснение решения:

1) Импорт модуля time для отслеживания времени работы программы.
2) Обьявление переменной счетчика времени.
3) Заданы левая и правая границы интервала допустимых значений для n и для чисел массива.
4) Из файла считывается n и проверяется на соответствие интервалу.
5) Из файла считывается массив и каждый элемент проверяется на соответствие интервалу.
6) Создается цикл for, который проходит со второго элемента и до конца. Слева от элемента находится отсортированная часть массива. Элемент сравнивается с элементами слева до тех пор, пока не встанет на свое место. Каждый раз два сравниваемых элемента, при положительном результате, меняются местами процедурой swap.
7) Отсортированный массив записывается в файл.
8) Выводится время работы алгоритма.

Примеры работы кода:



Нижняя граница диапазона значений входных данных из текста задачи	Время выполнения: 0.0011163999988639262 секунд
Пример из задачи	Время выполнения: 0.0010108000024047215 секунд

Верхняя граница диапазона значений входных данных из текста задачи	Время выполнения: 0.0651082000003953 секунд

Вывод по задаче:

В решении задачи использованы переменные, операторы сравнения и if-else конструкции, цикл for, работа с файлами. Код выполняет поставленную задачу. В отличие от первой задачи в третьей используется процедура swap и массив сортируется в порядке убывания.

Задание №5. Сортировка выбором.

Текст задачи:

5 задача. Сортировка выбором.

Рассмотрим сортировку элементов массива , которая выполняется следующим образом. Сначала определяется наименьший элемент массива , который ставится на место элемента $A[1]$. Затем производится поиск второго наименьшего элемента массива A, который ставится на место элемента $A[2]$. Этот процесс продолжается для первых $n-1$ элементов массива A.

Напишите код этого алгоритма, также известного как сортировка выбором (selection sort). Определите время сортировки выбором в наихудшем случае и в среднем случае и сравните его со временем сортировки вставкой.

Формат входного и выходного файла и ограничения - как в задаче 1.

Код:

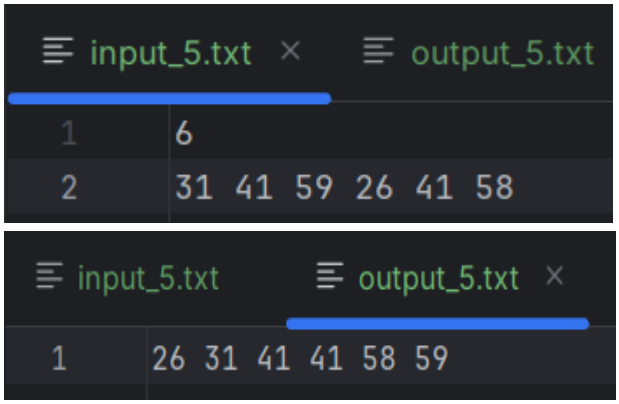
```
import time

t_start = time.perf_counter()
_min = - 10 ** 9
_max = 10 ** 9
n_min = 1
n_max = 10 ** 3
f = open('input_5.txt')
n = int(f.readline())
if n_min <= n <= n_max:
    s = list(map(int, f.readline().split()))
    flag = True
    for el in s:
        if not (_min < el < _max):
            flag = False
    if flag:
        for i in range(0, len(s) - 1):
            ind = i
            for j in range(i + 1, len(s)):
                if s[j] < s[ind]:
                    ind = j
            s[i], s[ind] = s[ind], s[i]
        f_output = open('output_5.txt', 'w')
        f_output.write(' '.join(map(str, s)))
```

```
        t_stop = time.perf_counter()
        if t_stop - t_start <= 2:
            print("Время выполнения:", t_stop - t_start, 'секунд')
        else:
            print("Превышено время выполнения:", t_stop - t_start,
'секунд')
    else:
        f_output = open('output_5.txt', 'w')
        f_output.write('Введены некоректные данные')
else:
    f_output = open('output_5.txt', 'w')
    f_output.write('Введены некоректные данные')
f.close()
f_output.close()
```

Текстовое объяснение решения:

1) Импорт модуля time для отслеживания времени работы программы.
2) Обьявление переменной счетчика времени.
3) Заданы левая и правая границы интервала допустимых значений для n и для чисел массива.
4) Из файла считывается n и проверяется на соответствие интервалу.
5) Из файла считывается массив и каждый элемент проверяется на соответствие интервалу.
6) Создается цикл for, который проходит с первого элемента и до предпоследнего. Создается еще один вложенный цикл for, в котором проходят по всем элементам массива справа от нашего элемента и выбирается наименьший. Слева от элемента находится отсортированная часть массива. Наименьший элемент выбранный в правой части меняется с элементом местами.
7) Отсортированный массив записывается в файл.
8) Выводится время работы алгоритма.

Пример работы кода:



Нижняя граница диапазона значений входных данных из текста задачи	Время выполнения: 0.0010892999998759478 секунд
Пример из задачи	Время выполнения: 0.0011743999966711272 секунд
Верхняя граница диапазона значений входных данных из текста задачи	Время выполнения: 0.038219099998968886 секунд

Вывод по задаче:

В решении задачи использованы переменные, операторы сравнения и if-else конструкции, цикл for, работа с файлами. Код выполняет поставленную задачу. В отличие от первой задачи в пятой используется процедура swap, а также сначала справа ищется минимальный элемент и становится в конец отсортированной части массива.

Дополнительные задачи

Задание №2. Сортировка вставкой +.

Текст задачи:

2 задача. Сортировка вставкой +

Измените процедуру Insertion-sort для сортировки таким образом, чтобы в выходном файле отображалось в первой строке n чисел, которые обозначают новый индекс элемента массива после обработки.

- **Формат выходного файла (input.txt).**В первой строке выходного файла выведите n чисел. При этом i-ое число равно индексу, на который, в момент обработки его сортировкой вставками, был перемещен i-ый элемент исходного массива. Индексы нумеруются, начиная с единицы. Между любыми двумя числами должен стоять ровно один пробел.

Пример.

input.txt	output.txt
10 1 8 4 2 3 7 5 6 9 0	1 2 2 2 3 5 5 6 9 1 0 1 2 3 4 5 6 7 8 9

В примере сортировка вставками работает следующим образом:

- Первый элемент остается на своем месте, поэтому первое число в ответе — единица. Отсортированная часть массива: [1]
- Второй элемент больше первого, поэтому он тоже остается на своем месте, и второе число в ответе — двойка. [1 8]
- Четверка меньше восьмерки, поэтому занимает второе место. [1 4 8]
- Двойка занимает второе место. [1 2 4 8]
- Тройка занимает третье место. [1 2 3 4 8]

Код:

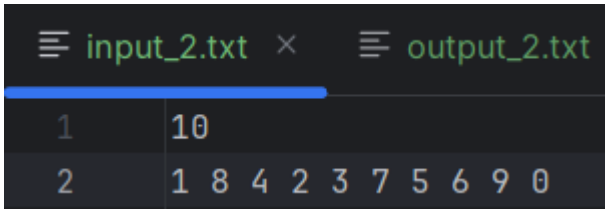
```
import time

t_start = time.perf_counter()
_min = - 10 ** 9
_max = 10 ** 9
n_min = 1
n_max = 10 ** 3
f = open('input_2.txt')
n = int(f.readline())
if n_min <= n <= n_max:
    s = list(map(int, f.readline().split()))
    flag = True
    for el in s:
        if not (_min < el < _max):
            flag = False
    if flag:
        indices = [1] + [0] * (len(s) - 1)
        for i in range(1, len(s)):
            a = s[i]
            j = i - 1
            while j >= 0 and a < s[j]:
                s[j + 1] = s[j]
                j = j - 1
            indices[i] = j + 2
```

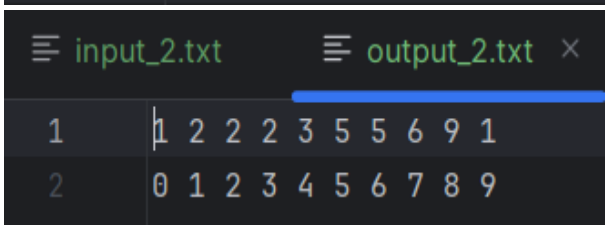
```
            s[j + 1] = a
        f_output = open('output_2.txt', 'w')
        f_output.write(' '.join(map(str, indices)) + '\n')
        f_output.write(' '.join(map(str, s)))
        t_stop = time.perf_counter()
        if t_stop - t_start <= 2:
            print("Время выполнения:", t_stop - t_start, 'секунд')
        else:
            print("Превышено время выполнения:", t_stop - t_start,
'секунд')
    else:
        f_output = open('output_2.txt', 'w')
        f_output.write('Введены некоректные данные')
else:
    f_output = open('output_2.txt', 'w')
    f_output.write('Введены некоректные данные')
f.close()
f_output.close()
```

Текстовое объяснение решения:

1) Импорт модуля time для отслеживания времени работы программы.
2) Обьявление переменной счетчика времени.
3) Заданы левая и правая границы интервала допустимых значений для n и для чисел массива.
4) Из файла считывается n и проверяется на соответствие интервалу.
5) Из файла считывается массив и каждый элемент проверяется на соответствие интервалу.
6) Создается цикл for, который проходит со второго элемента и до конца. Слева от элемента находится отсортированная часть массива. Элемент сравнивается с элементами слева до тех пор, пока не встанет на свое место.
7) Также по условию задачи нужно помимо отсортированного массива вывести массив с местами куда встали в отсортированном массиве элементы из неотсортированного массива. Это выполняется строчками:
   ```
   indices = [1] + [0] * (len(s) - 1)
   indices[i] = j + 2
   ```
   На место элемента, который мы передвигали, записывается место на которое этот элемент передвинули (j + 1 это индекс, на который поместили элемент, чтобы записать место записывается индекс + 1).
8) Отсортированный массив записывается в файл.
9) Выводится время работы алгоритма.

Пример работы кода:





Нижняя граница диапазона значений входных данных из текста задачи	Время выполнения: 0.0011599999997997656 секунд
Пример из задачи	Время выполнения: 0.0010712000002968125 секунд
Верхняя граница диапазона значений входных данных из текста задачи	Время выполнения: 0.04242549999980838 секунд

Вывод по задаче:
В решении задачи использованы переменные, операторы сравнения и if-else конструкции, цикл for, работа с файлами. Код выполняет поставленную задачу. В отличие от первой задачи во второй нужно помимо отсортированного массива вывести массив с местами куда встали в отсортированном массиве элементы из неотсортированного массива.

Задание №8. Секретарь Своп.

Текст задачи:

8 задача. Секретарь Своп

Дан массив, состоящий из n целых чисел. Вам необходимо его отсортировать по неубыванию. Но делать это нужно так же, как это делает мистер Своп — то есть, каждое действие должно быть взаимной перестановкой пары элементов. Вам также придется записать все, что Вы делали, в файл, чтобы мистер Своп смог проверить Вашу работу.

- **Формат входного файла (input.txt).** В первой строке входного файла содержится число n ($3 \leq n \leq 5000$) — число элементов в массиве. Во второй строке находятся n целых чисел, по модулю не превосходящих 10^9. Числа могут совпадать друг с другом.

- **Формат выходного файла (output.txt).** В первых нескольких строках выведите осуществленные Вами операции перестановки элементов. Каждая строка должна иметь следующий формат:

  `Swap elements at indices X and Y.`

  Здесь X и Y — различные индексы массива, элементы на которых нужно переставить ($1 \leq X, Y \leq n$). Мистер Своп любит порядок, поэтому сделайте так, чтобы $X < Y$.

  После того, как все нужные перестановки выведены, выведите следующую фразу:

  `No more swaps needed.`

- Пример:

input.txt	output.txt
5 3 1 4 2 2	Swap elements at indices 1 and 2. Swap elements at indices 2 and 4. Swap elements at indices 3 and 5. No more swaps needed.

Код:

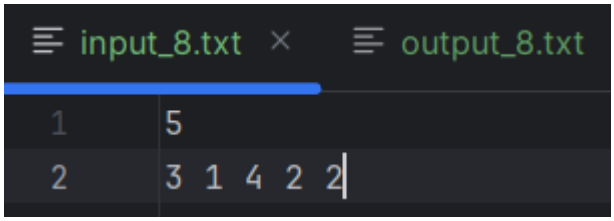
```
import time

t_start = time.perf_counter()
_min = - 10 ** 9
_max = 10 ** 9
n_min = 3
n_max = 5000
f = open('input_8.txt')
n = int(f.readline())
if n_min <= n <= n_max:
    s = list(map(int, f.readline().split()))
    flag = True
    for el in s:
        if not (_min < el < _max):
            flag = False
    if flag:
        f_output = open('output_8.txt', 'w')
        for i in range(0, len(s) - 1):
```

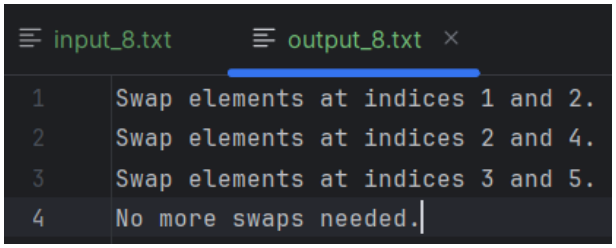
```
            ind = i
            for j in range(i + 1, len(s)):
                if s[j] < s[ind]:
                    ind = j
            if i != ind:
                s[i], s[ind] = s[ind], s[i]
                f_output.write(f'Swap elements at indices {i + 1} and {ind
+ 1}.\n')
        f_output.write('No more swaps needed.')
        t_stop = time.perf_counter()
        if t_stop - t_start <= 1:
            print("Время выполнения:", t_stop - t_start, 'секунд')
        else:
            print("Превышено время выполнения:", t_stop - t_start,
'секунд')
    else:
        f_output = open('output_8.txt', 'w')
        f_output.write('Введены некоректные данные')
else:
    f_output = open('output_8.txt', 'w')
    f_output.write('Введены некоректные данные')
f.close()
f_output.close()
```

Текстовое объяснение решения:

1) Импорт модуля time для отслеживания времени работы программы.
2) Обьявление переменной счетчика времени.
3) Заданы левая и правая границы интервала допустимых значений для n и для чисел массива.
4) Из файла считывается n и проверяется на соответствие интервалу.
5) Из файла считывается массив и каждый элемент проверяется на соответствие интервалу.
6) Создается цикл for, который проходит с первого элемента и до предпоследнего. Создается еще один вложенный цикл for, в котором проходят по всем элементам массива справа от нашего элемента и выбирается наименьший. Слева от элемента находится отсортированная часть массива. Наименьший элемент выбранный в правой части меняется с элементом местами. В файл записывается строчка с индексами элементов, которые поменялись.
7) Выводится время работы алгоритма.

Пример работы кода:





	Время выполнения
Нижняя граница диапазона значений входных данных из текста задачи	Время работы: 0. 0.0018050999970000703 секунд
Пример из задачи	Время работы: 0.0015028999987407587 секунд
Верхняя граница диапазона значений входных данных из текста задачи	Время работы: 0.9659905000007711 секунд

Вывод по задаче:
В решении задачи использованы переменные, операторы сравнения и if-else конструкции, цикл for, работа с файлами. Код выполняет поставленную задачу. В отличие от пятой задачи в восьмой записывается в файл не отсортированный массив, а строчки о каждой операции перестановки элементов.

Задание № 10. Палиндром.

Текст задачи:

10 задача★. Палиндром

Палиндром - это строка, которая читается одинаково как справа налево, так и слева направо.

На вход программы поступает набор больших латинских букв (не обязательно различных). Разрешается переставлять буквы, а также удалять некоторые буквы. Требуется из данных букв по указанным правилам составить палиндром наибольшей длины, а если таких палиндромов несколько, то выбрать первый из них в алфавитном порядке.

- **Формат входного файла (input.txt).** В первой строке входных данных содержится число n ($1 \leq n \leq 100000$). Во второй строке задается последовательность из n больших латинских букв (буквы записаны без пробелов).
- **Формат выходного файла (output.txt).** В единственной строке выходных данных выдайте искомый палиндром.

Код:

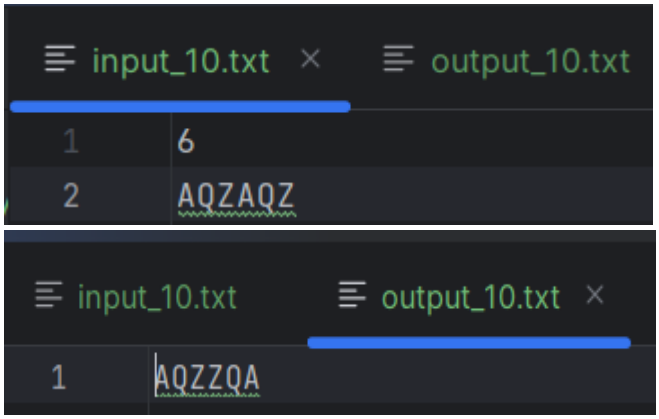
```
import time

t_start = time.perf_counter()
n_min = 1
n_max = 10 ** 5
f = open('input_10.txt')
n = int(f.readline())
if n_min <= n <= n_max:
    s = f.readline()
    ord_A = ord("A")
    a = [0] * (ord("Z") - ord_A + 1)
    for elem in s:
        a[ord(elem) - ord_A] += 1
    left_side = ''
    center = ''
    for i in range(len(a)):
        ch, cnt = chr(i + ord_A), a[i]
        h = ch * (cnt // 2)
        left_side += h
        if center == "" and cnt % 2 == 1:
            center = ch
    f_output = open('output_10.txt', 'w')
    f_output.write(left_side + center + left_side[::-1])
    t_stop = time.perf_counter()
    if t_stop - t_start <= 1:
        print("Время выполнения:", t_stop - t_start, 'секунд')
    else:
        print("Превышено время выполнения:", t_stop - t_start, 'секунд')
else:
    f_output = open('output_10.txt', 'w')
    f_output.write('Введены некоректные данные')
f.close()
f_output.close()
```

Текстовое объяснение решения:

1) Импорт модуля time для отслеживания времени работы программы.

2) Обьявление переменной счетчика времени.
3) Заданы левая и правая границы интервала допустимых значений для n.
4) Из файла считывается n и проверяется на соответствие интервалу.
5) Из файла считывается строка состоящая из заглавных букв.
6) Циклом for проходимся по всем элементам строки и в массив записываем количество каждой буквы в строке.
7) Проходимся по элементам массива, в котором записано количество каждой буквы, и добавляем к левой части палиндрома количество поделенное на два (т.к. в правую часть столько же букв должно уйти).
8) Находим центр палиндрома, т.к. если какой-то буквы будет нечетное количество, то для достижения максимальной длинны палиндрома нужно вставить эту букву в середину.
9) В файл записывается палиндром состоящий из левой части + центра + отраженной левой части.
10) Выводится время работы алгоритма.

Пример работы кода:



	Время выполнения
Нижняя граница диапазона значений входных данных из текста задачи	Время работы: 0.0009924000005412381 секунд
Пример из задачи	Время работы: 0.002334000000701053 секунд

Верхняя граница диапазона значений входных данных из текста задачи	Время работы: 0.01982560000033118 секунд

Вывод по задаче:
В решении задачи использованы переменные, операторы сравнения и if-else конструкции, цикл for, работа с файлами, функции ord и chr. Код выполняет поставленную задачу.

Вывод:
В данной лабораторной работе я практиковалась в работе с файлами, с логическими конструкциями, с модулем time, для оценки скорости работы алгоритмов. Научилась писать различные сортировки массивов.