

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

Выполнил:

Полегкий А.С. (фамилия имя)

К3142 (номер группы)

12 (вариант)

Проверила:

Артамонова В.Е.

Санкт-Петербург

2023 г.

Содержание отчета

Содержание отчета 2

Задачи по варианту 3

Задание №1 Сортировка вставкой [N баллов] 3

Задание №3. Сортировка вставкой по убыванию [N баллов] 4

Задание №9. Сложение двоичных чисел [N баллов] 6

Вывод 8

Задачи по варианту

1 задача. Сортировка вставкой

Листинг кода:

```
with open("input.txt", "r") as f:
    n = int(f.readline())
    mas = [int(x) for x in f.readline().split()]

for i in range(1, len(mas)):
    for j in range(i - 1, -1, -1):
        if mas[j] > mas[j + 1]:
            mas[j], mas[j + 1] = mas[j + 1], mas[j]

with open("output.txt", "w") as f:
    f.write(" ".join([str(x) for x in mas]))
```

Текстовое объяснение решения:

Сначала мы считываем массив из входного файла с помощью функции `open()`, а после в переменную `n` записываем число с первой строки файла "input.txt", что будет определять количество чисел в массиве `mas`. Следующая строка в файле будет определять числа, которые будут находиться в массиве `mas`. Затем делаем цикл, в котором будет происходить сортировка чисел 2 раза. В первом проходе мы сравниваем каждый элемент с предыдущим элементом. Если предыдущий элемент больше текущего, то мы меняем их местами. Во втором проходе мы повторяем тот же алгоритм, но начинаем с конца массива. В конце отсортированный массив записываем в файл "output.txt".

Результаты работы кода на примерах из текста задачи:

Task1.py		input.txt		output.txt	
1	5			1	26 31 41 41 58 59
2	31 41 59 26 41				

Вывод по задаче: код позволяет считывать числа с файла и сортировать их по возрастанию.

3 Задача. Сортировка вставкой по убыванию

Листинг кода:

```
with open("input3.txt", "r") as f:
    n = int(f.readline())
    mas = [int(x) for x in f.readline().split()]
def insertion_sort(mas):
    for i in range(1, len(mas)):
        key = mas[i]
        j = i - 1

        while j >= 0 and mas[j] < key:
            swap(mas, j, j + 1)
            j -= 1

        if j >= 0:
            mas[j + 1] = key

def swap(mas, i, j):
    temp = mas[i]
    mas[i] = mas[j]
    mas[j] = temp

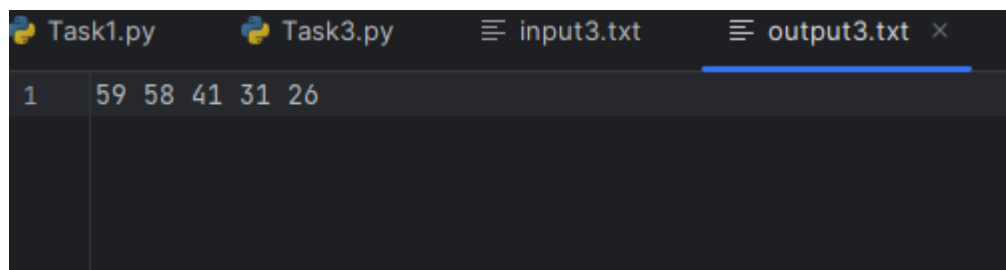
insertion_sort(mas)
with open("output3.txt", "w") as f:
    f.write(" ".join([str(x) for x in mas]))
```

Текстовое объяснение решения:

Сначала мы считываем массив из входного файла с помощью функции `open()`, а после в переменную `n` записываем число с первой строки файла "input.txt", что будет определять количество чисел в массиве `mas`. Следующая строка в файле будет определять числа, которые будут находиться в массиве `mas`. В данном случае мы сортируем массив в невозрастающем порядке. Это означает, что в отсортированном массиве элементы расположены в порядке возрастания. Чтобы сортировать массив в невозрастающем порядке, мы должны изменить направление сравнения в цикле `while`. В данном случае мы сравниваем элемент `key` с элементом `mas[j]` в обратном порядке. Мы также должны добавить проверку на равенство, чтобы избежать бесконечного цикла, если элементы отсортированного подмассива равны. В конце отсортированный массив записываем в файл "output.txt".

Результаты работы кода на примерах из текста задачи:

Task1.py × Task3.py input3.txt × output3.txt	
1	5
2	31 41 59 26 58



The image shows a code editor window with four tabs: Task1.py, Task3.py, input3.txt, and output3.txt. The output3.txt tab is active and shows the text "1 59 58 41 31 26" on the first line. The editor has a dark theme and a light blue underline under the active tab.

1	59	58	41	31	26
---	----	----	----	----	----

Вывод по задаче: сортировку массива также можно делать с помощью процедуры Swap.

9 задача. Сложение двоичных чисел

Листинг кода:

```
def sum(a, b):
    n = len(a)
    c = [0] * (n + 1)
    k = 0
    for i in range(n - 1, -1, -1):
        sum = a[i] + b[i] + k
        c[i + 1] = sum % 2
        k = sum // 2

    c[0] = k
    return c

def main():
    with open("input9.txt", "r") as f:
        a, b = f.readline().split()

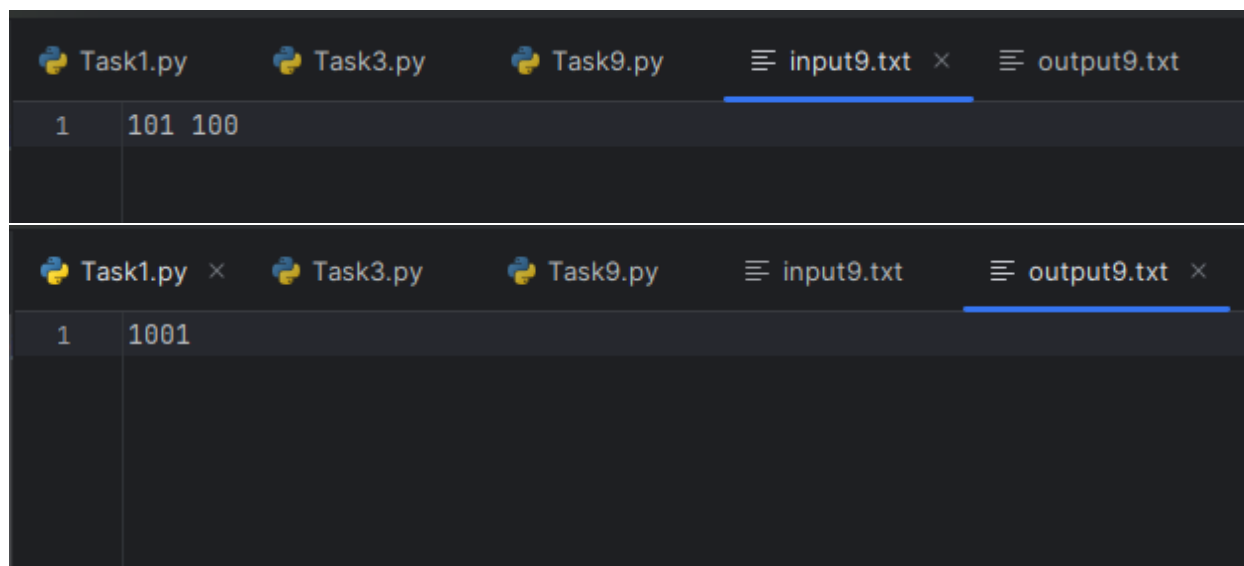
    c = sum(list(map(int, a)), list(map(int, b)))
    with open("output9.txt", "w") as f:
        f.write("".join(map(str, c)))

if __name__ == "__main__":
    main()
```

Текстовое пояснение к задаче:

Функция `sum` складывает два двоичных числа, хранящихся в массивах `a` и `b`. Она возвращает массив `c`, содержащий сумму двух чисел. Создаём новый массив «с» длины `n + 1`, заполненный нулями. Цикл `for` перебирает элементы массивов `a` и `b` в обратном порядке, от младшего разряда к старшему. «Sum» складывает элементы массивов `a` и `b` на текущей позиции, а также перенос из младшего разряда. «`c[i + 1] = sum % 2`» сохраняет результат сложения в массиве «с» на следующей позиции. После того, как цикл `for` завершится, массив «с» будет содержать сумму двух чисел. Далее в функции `main` считываем файл и в переменной «с» складываем 2 числа из файла с помощью функции “`sum`”. Записываем результат сложения в “`output9`”. Последний `if` предотвращает выполнение кода, предназначенного только для использования внутри модуля, при импорте модуля в другой модуль.

Результаты работы на собственных примерах:



Вывод по задаче: код может считывать 2 двоичных числа из файла и считать их сумму.

Вывод.

В ходе данной лабораторной работы я ознакомился с сортировкой вставками, методом swar, а также со сложением двоичных чисел с помощью массивов.