

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:  
Марченко В.А.  
К3141

Проверил:  
Афанасьев А.В.

Санкт-Петербург  
2024 г.

## Содержание отчета

Содержание отчета.....	2
Задания .....	3
Задание 1. Сортировка вставкой .....	3
Задание 2. Сортировка вставкой +.....	5
Задание 3. Сортировка вставкой по убыванию .....	7
Задание 4. Линейный поиск .....	9
Задание 5. Сортировка выбором .....	11
Задание 10. Палиндром.....	13
Вывод.....	16

## Задания

### Задание 1. Сортировка вставкой

В данной задаче требуется отсортировать массив чисел из файла с помощью сортировки вставкой (Insertion-sort).

Листинг кода:

```
def insertionsort(input_name, output_name):
    input_file = open(input_name, 'r')
    output_file = open(output_name, 'w')

    n = int(input_file.readline())
    arr = list(map(int, input_file.readline().split()))

    for i in range(1, n):
        key = arr[i]
        j = i - 1

        while j >= 0 and key < arr[j]:
            arr[j + 1] = arr[j]
            j -= 1
        arr[j + 1] = key

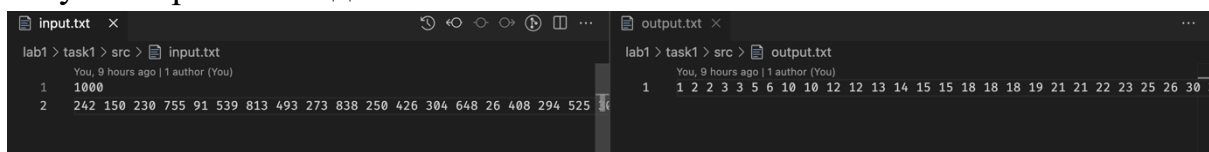
    output_file.write(' '.join(map(str, arr)))

    input_file.close()
    output_file.close()
```

Текстовое объяснение решения:

Сначала открываются два файла: один для чтения исходных данных, другой – для записи отсортированных результатов. Из входного файла считывается первое число, которое указывает количество элементов в массиве, затем сам массив чисел. После этого выполняется сортировка методом вставки: для каждого элемента массива, начиная со второго, находится его место среди уже отсортированных элементов слева. Внутренний цикл сдвигает элементы вправо, пока не найдётся позиция для текущего элемента. После завершения сортировки, отсортированный массив преобразуется в строку и записывается в выходной файл. В конце файлы закрываются.

Результат работы кода:



```
lab1 > task1 > src > input.txt
You, 9 hours ago | 1 author (You)
1 1000
2 242 150 230 755 91 539 813 493 273 838 250 426 304 648 26 408 294 525 1000

lab1 > task1 > src > output.txt
You, 9 hours ago | 1 author (You)
1 1 2 2 3 3 5 6 10 10 12 12 13 14 15 15 18 18 18 19 21 21 22 23 25 26 30 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000
```

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.00036 ms	0.01805 MB
Верхняя граница диапазона значений входных данных из текста задачи	0.18421 ms	0.11822 MB

Вывод по задаче:

Таким образом, в этой задаче мы изучили и реализовали сортировку вставкой. Алгоритм работает эффективно для небольших наборов данных.

## Задание 2. Сортировка вставкой +

В данной задаче требуется отредактировать код из прошлого задания так, чтобы в первой строчке файла от так же выводил для каждого числа новый индекс, соответствующего ниже элемента.

Листинг кода:

```
def insertionsort_indexes(input_name, output_name):
    input_file = open(input_name, 'r')
    output_file = open(output_name, 'w')

    n = int(input_file.readline())
    arr = list(map(int, input_file.readline().split()))
    indexes = list(range(1, n + 1))

    for i in range(1, n):
        key = arr[i]
        j = i - 1

        while j >= 0 and key < arr[j]:
            arr[j + 1] = arr[j]
            j -= 1
        arr[j + 1] = key
        indexes[i] = j + 2

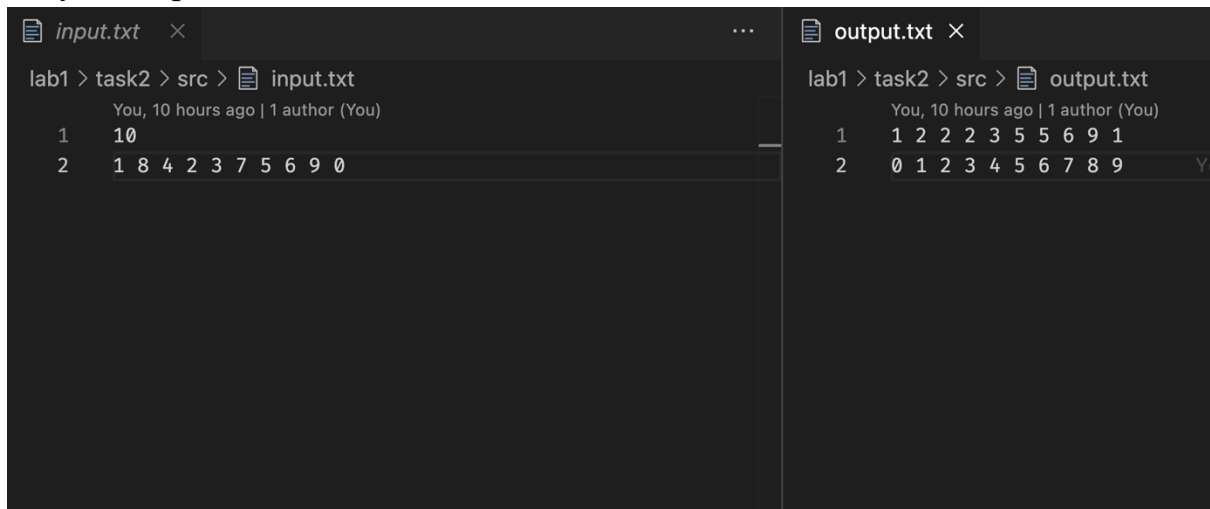
    output_file.write(' '.join(map(str, indexes)) + "\n")
    output_file.write(' '.join(map(str, arr)))

    input_file.close()
    output_file.close()
```

Текстовое объяснение решения:

Сначала открываются входной и выходной файлы: первый для чтения исходных данных, второй для записи результатов. Из входного файла считывается количество элементов массива и сам массив, после чего создаётся массив `indexes`, содержащий начальные индексы каждого элемента. Затем начинается сортировка методом вставки, при которой каждый элемент вставляется на своё место в отсортированной части массива. При этом одновременно обновляются соответствующие индексы в массиве `indexes`, чтобы отслеживать новые позиции элементов. Индекс обновляется таким образом, чтобы он отражал положение элемента в отсортированном массиве с учётом того, что индексация начинается с 1. После завершения сортировки в выходной файл записываются два ряда данных: в первой строке – новые индексы элементов, а во второй – сам отсортированный массив.

## Результат работы кода:



The screenshot shows a code editor with two tabs: `input.txt` and `output.txt`. Both tabs show the same directory path: `lab1 > task2 > src >`. The `input.txt` tab displays the following content:

```
You, 10 hours ago | 1 author (You)
1 10
2 1 8 4 2 3 7 5 6 9 0
```

The `output.txt` tab displays the following content:

```
You, 10 hours ago | 1 author (You)
1 1 2 2 2 3 5 5 6 9 1
2 0 1 2 3 4 5 6 7 8 9
```

## Вывод по задаче:

В этой задаче мы успешно реализовали сортировку вставки с отслеживанием индексов элементов, что помогает проследить, как менялись индексы элементов в ходе сортировки.

### Задание 3. Сортировка вставкой по убыванию

В данной задаче требуется отредактировать код из задания 1 так, чтобы сортировка происходила в невозрастающем порядке. Так же необходимо использовать операцию Swap.

Листинг кода:

```
def insertionsort_reversed(input_name, output_name):
    input_file = open(input_name, 'r')
    output_file = open(output_name, 'w')

    n = int(input_file.readline())
    arr = list(map(int, input_file.readline().split()))

    for i in range(1, n):
        j = i
        while j > 0 and arr[j - 1] < arr[j]:
            arr[j], arr[j - 1] = arr[j - 1], arr[j]
            j -= 1

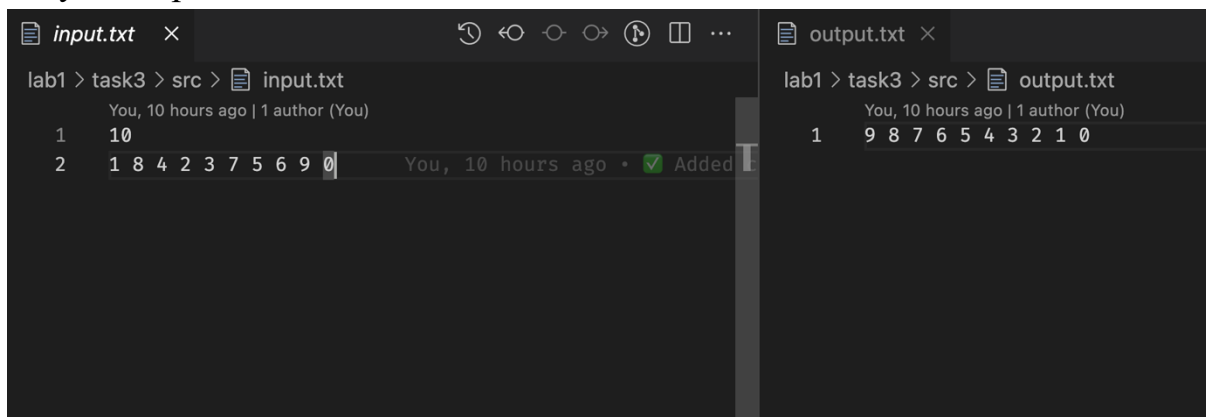
    output_file.write(' '.join(map(str, arr)))

    input_file.close()
    output_file.close()
```

Текстовое объяснение решения:

Сначала открываются входной и выходной файлы, считывается количество элементов и сам массив чисел. Затем начинается сортировка: для каждого элемента начиная со второго, выполняется цикл, который перемещает элемент на правильное место среди уже отсортированных элементов слева, если он больше предыдущего элемента. Для перемещения используется операция Swap, которая меняет местами текущий элемент и предыдущий, если первый больше. После завершения сортировки отсортированный массив записывается в выходной файл.

Результат работы кода:



Вывод по задаче:

В этой задаче мы успешно поменяли порядок сортировки вставкой, а также узнали о том, как мы можем в Python менять местами элементы с помощью операции Swap.



#### Задание 4. Линейный поиск

Необходимо написать код линейного поиска, которые ищет заданный элемент  $V$  в последовательности. Если такого элемента нет, то выводится  $-1$ . Если один такой элемент есть, выводится его индекс в последовательности. Если их несколько, то сначала выводится сколько раз встречается число, а потом индексы равных элементов через запятую.

Листинг кода:

```
def linearsearch(input_name, output_name):
    input_file = open(input_name, 'r')
    output_file = open(output_name, 'w')

    numbers = list(map(int, input_file.readline().split()))
    v = int(input_file.readline())

    repeat_indexes = []
    for i in range(len(numbers)):
        if numbers[i] == v:
            repeat_indexes.append(i)

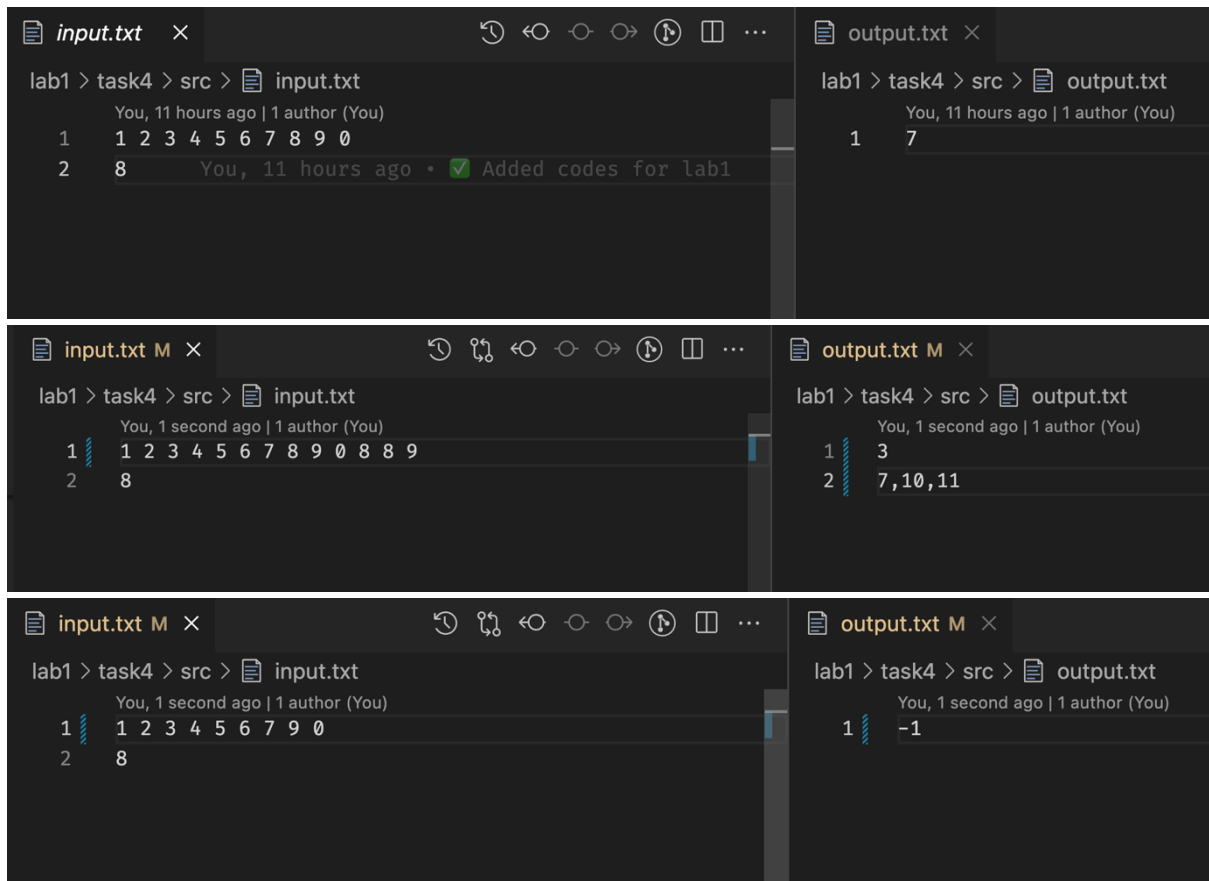
    if len(repeat_indexes) == 0:
        output_file.write("-1")
    elif len(repeat_indexes) == 1:
        output_file.write(str(repeat_indexes[0]))
    else:
        output_file.write("{}\n".format(len(repeat_indexes)))
        output_file.write(','.join(map(str, repeat_indexes)))

    input_file.close()
    output_file.close()
```

Текстовое объяснение решения:

Сначала открываются файлы для чтения и записи данных. Из входного файла считываются последовательность чисел и искомое число  $V$ . Затем происходит линейный проход по последовательности, в ходе которого индексы всех найденных вхождений числа  $v$  добавляются в список `repeat_indexes`. Если число не встречается, программа выводит  $-1$ . Если найдено одно вхождение, выводится индекс этого элемента. Если найдено несколько вхождений, сначала выводится их количество, а затем индексы всех вхождений через запятую. В конце файлы закрываются.

Результаты работы кода:



Вывод по задаче:

В этой задаче мы реализовали линейный поиск, а также различный вывод данных в зависимости от результата.

## Задание 5. Сортировка выбором

Необходимо написать код сортировки выбором (Selection-sort), 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  $A$ .

Листинг кода:

```
def selectionsort(input_name, output_name):
    input_file = open(input_name, 'r')
    output_file = open(output_name, 'w')

    n = int(input_file.readline())
    arr = list(map(int, input_file.readline().split()))

    for i in range(n):
        min_index = i
        for j in range(i + 1, n):
            if arr[j] < arr[min_index]:
                min_index = j
        (arr[i], arr[min_index]) = (arr[min_index], arr[i])

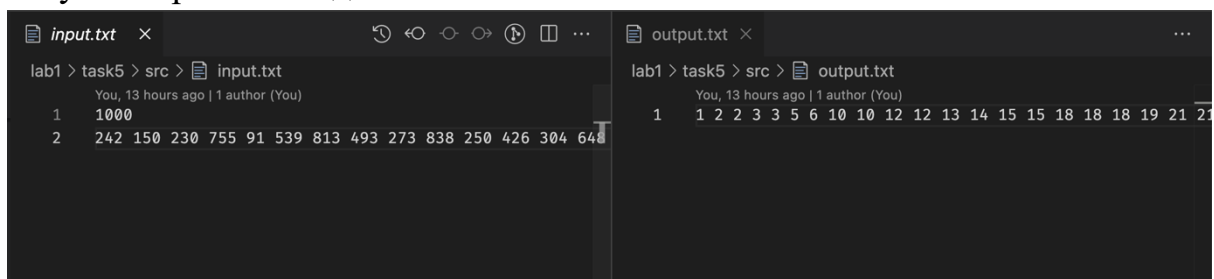
    output_file.write(' '.join(map(str, arr)))

    input_file.close()
    output_file.close()
```

Текстовое объяснение решения:

Сначала открываются входной и выходной файлы, считывается количество элементов и сам массив. Внешний цикл перебирает каждый элемент массива, начиная с первого, а внутренний цикл ищет наименьший элемент в оставшейся части массива. После нахождения минимального элемента, он меняется местами с текущим элементом. Таким образом, массив постепенно становится отсортированным. В конце отсортированный массив записывается в выходной файл.

Результат работы кода:



```
lab1 > task5 > src > input.txt
You, 13 hours ago | 1 author (You)
1 1000
2 242 150 230 755 91 539 813 493 273 838 250 426 304 648

lab1 > task5 > src > output.txt
You, 13 hours ago | 1 author (You)
1 1 2 2 3 3 5 6 10 10 12 12 13 14 15 15 18 18 19 21 21
```

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.00051 ms	0.01805 MB
Верхняя граница диапазона значений входных данных из текста задачи	0.19209 ms	0.11826 MB

Вывод по задаче:

В этой задаче мы узнали о новом методе сортировке Selection-sort, а затем успешно реализовали его. Такой подход обеспечивает стабильную работу для сортировки массива за фиксированное количество операций, что делает его подходящим для небольших наборов данных.

## Задание 10. Палиндром

В этой задаче нам необходимо составить палиндром максимальной длины из данного набора символов. Если палиндромов максимальной длины несколько, то нужно выбрать первый из них в алфавитном порядке.

Листинг кода:

```
def count_letters(string):
    letters = {}
    for letter in string:
        if letter in letters:
            letters[letter] += 1
        else:
            letters[letter] = 1
    return letters

def palindrome(input_name, output_name):
    input_file = open(input_name, 'r')
    output_file = open(output_name, 'w')

    input_file.readline()
    letters = count_letters(input_file.readline().replace("\n", ""))

    even_part = []
    odd_part = []

    for letter, freq in sorted(letters.items()):
        if freq % 2 == 0:
            even_part.append(letter * (freq // 2))
        else:
            even_part.append(letter * (freq // 2))
            odd_part.append(letter)

    left = ''.join(even_part)
    center = odd_part[0] if odd_part else ''
    right = left[::-1]

    output_file.write(left + center + right)

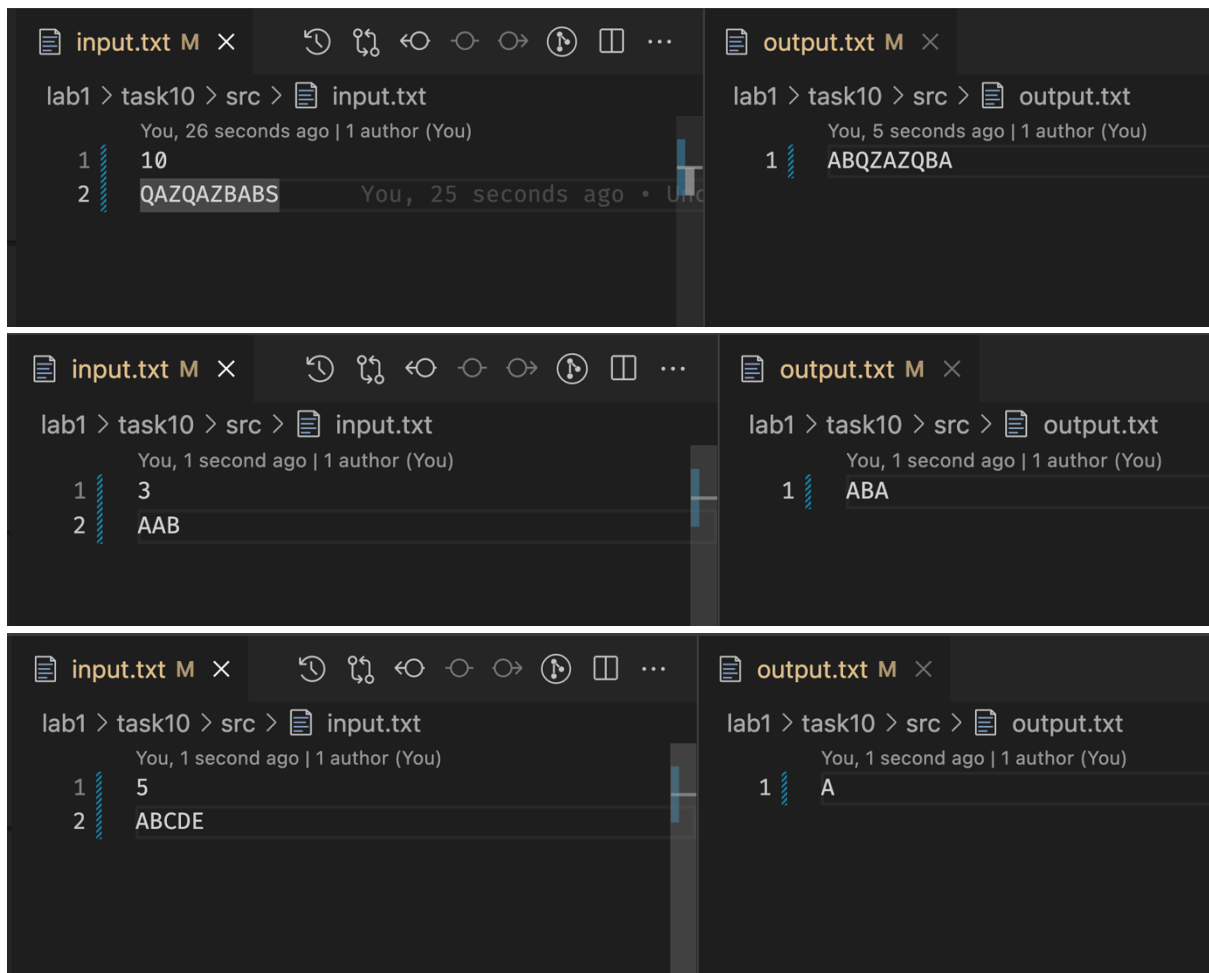
    input_file.close()
    output_file.close()
```

Текстовое объяснение решения:

Сначала открываются входной и выходной файлы, и считываются данные. Затем вызывается вспомогательная функция `count_letters`, которая подсчитывает количество каждого символа в строке и возвращает словарь с частотой появления каждой буквы. После этого символы сортируются по алфавиту, и каждый символ с чётной частотой добавляется к левой и правой части будущего палиндрома, а символы с нечётной частотой сохраняются

для центральной части. В результате формируются три части палиндрома: левая – из половин символов с чётной частотой, центральная – если есть символ с нечётной частотой, и правая – зеркальное отображение левой. Готовый палиндром записывается в выходной файл.

Результаты работы кода:



	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.00025 ms	0.01805 MB
Верхняя граница диапазона значений входных данных из текста задачи	0.02936 ms	0.34772 MB

Вывод по задаче:

Таким образом, мы успешно написали код для алгоритма для составления палиндрома из набора символов.

## **Вывод**

В ходе данной лабораторной работы были изучены и реализованы различные алгоритмы сортировки и поиска, такие как сортировка вставками, выбором, сортировка вставками в обратном порядке, а также линейный поиск. В процессе работы также был изучен алгоритм составления палиндрома максимальной длины из набора символов. Каждое задание позволило глубже понять принципы работы алгоритмов сортировки и поиска, а также их применение на практике.