МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний аерокосмічний університет ім. М.Є. Жуковського
«Харківський авіаційний інститут»

Факультет радіоелектроніки, комп'ютерних систем та інфокомунікацій

Кафедра комп'ютерних систем, мереж і кібербезпеки (503)

Лабораторна робота № *1*

Дослідження алгоритмів сортування
(назва лабораторної роботи)

з дисципліни *Моделі та структури даних*
(шифр)

ХАІ.503.525а.03О.123-Комп'ютерна інженерія, ПЗ №9629619

Виконав студент гр. 525а *Литвиненко А.В.*
(№ групи) (П.І.Б.)
21.09.2022
(підпис, дата)

Перевірив канд. техн. наук, доцент

А. В. Шостак
(підпис, дата) (П.І.Б.)

Харків – 2022

Тема роботи: дослідження алгоритмів сортування

Мета роботи: дослідити і вивчити найпростіші алгоритми сортування, навчитися використовувати на практиці

Варіант 12

Задача 1

Частина 1. Постановка завдання

Умова:

1. Разработать проект для исследования алгоритмов сортировки в соответствии с вариантом (диаграмма вариантов использования проекта представлена на рис. 1)

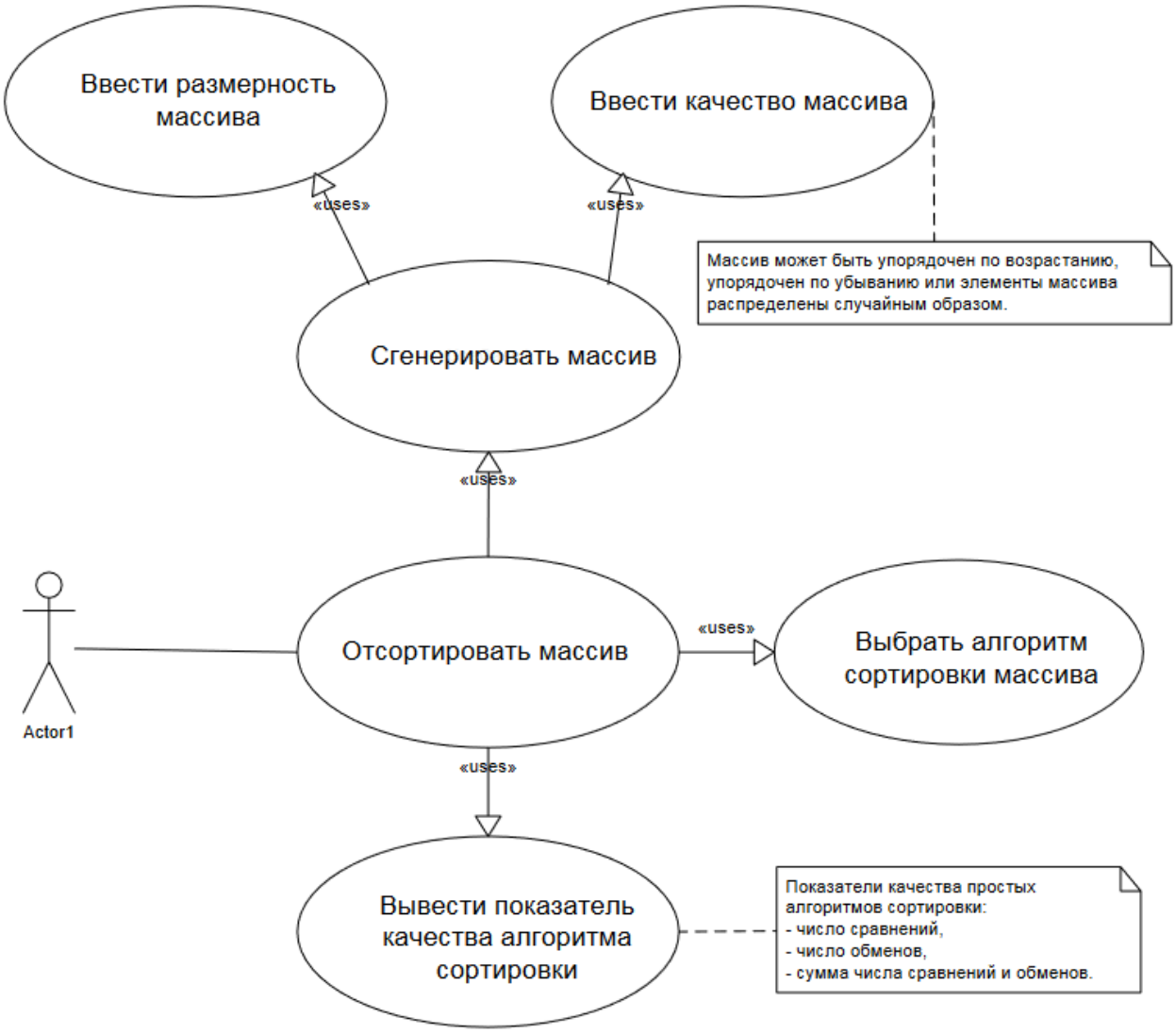


Рис. 1. Диаграмма вариантов использования проекта лабораторной работы № 1.

2. Разработать интерфейс проекта, позволяющий:
- задавать размерность и качество массива;
- осуществлять выбор алгоритма сортировки для исследования;
- осуществлять вывод информации о результатах исследования алгоритма сортировки (исходный и отсортированный массив, пошаговую работу

алгоритма сортировки (при небольшой размерности массива), показатели качества работы алгоритма сортировки).
3. Создать подпрограмму для генерации целочисленного массива различного качества. В подпрограмме предусмотреть:
- задание размерности массива;
- задание диапазона чисел массива;
- создание массива с равномерно распределенными случайными числами;
- создание упорядоченного по возрастанию массива;
- создание упорядоченного по убыванию массива.
4. Создать подпрограмму, реализующую алгоритм сортировки в соответствии с вариантом. В подпрограмме предусмотреть:
- определение числа сравнений;
- определение числа обменов;
- определение суммы обменов и сравнений;
- сортировку массива по возрастанию.

Умова з додатка:

12.	[0,600]	2, 4

Частина 2. Тексти програми

main.py

```
# import some functions
from time import time
from random import randint

# Main class
class Main:
    def __init__(self) -> None:
        self._array = []

    @staticmethod
    def show(_array: list) -> list:
        return " ".join(list(map(str, _array)))

    # static method for generating an array
    @staticmethod
    def genArray(_array: list,
            start: int = 0, end: int = 10, step:int = 1,
            startNum:int = 0, endNum:int = 100, orderby:int=0) -> list:
        _array = []
        for i in range(start, end, step):
            _array.append(randint(startNum, endNum))

        if orderby == -1:
            _array = _array.sort(reverse=True)
        elif orderby == 1:
            _array = _array.sort(reverse=False)
        else:
            pass

        return _array


    # property that shows an array
    @property
    def array(self) -> None:
        return self._array
        print(" ".j)
        for i in range(len(self._array)):
            print(self._array[i], end=" ")

    # propety.setter that set an array
```

```
    @array.setter
    def setArray(self, arr: list) -> None:
        self._array = arr.copy()


    # static method for buble sort
    @staticmethod
    def buble_sort(_array: list) -> list:
        array = _array.copy()
        size = len(array)
        flag = size < 15
        compare_count = 0
        cmp = 0
        exc = 0
        exchange_count = 0
        start_time = time()

        for i in range(size-1):
            for j in range(0, size-i-1):
                if array[j] > array[j + 1]:
                    if flag:
                        print(f"\tExchanging {array[j]} <> {array[j+1]} ({j}, {j+1})")
                    array[j], array[j + 1] = array[j + 1], array[j]
                    exc += 1
                cmp += 1
            print("\t[?] Compares count: %d" % cmp)
            print("\t[?] Exchange count: %d" % exc)
            if flag:
                print(i+1, Main.show(array), cmp, exc)
            compare_count += cmp
            exchange_count += exc
            cmp = 0
            exc = 0
        print("\t[!] Total compares count: %d" % compare_count)
        print("\t[!] Total exchange count: %d" % exchange_count)
        print(f"\t[!] Exchanges to compares count: {round(exchange_count /
compare_count * 100)}%")
        print("\t--- %.5s seconds ---" % (time()-start_time))

        return array

    # staticmethod for selection sort
    @staticmethod
    def selection_sort(_array: list) -> list:
```

```
        array = _array.copy()
        size = len(array)
        flag = size < 15
        compare_count = 0
        exchange_count = 0
        cmp = 0
        exc = 0

        start_time = time()
        for ind in range(size):
            min_index = ind
            for j in range(ind + 1, size):
                if array[j] < array[min_index]:
                    min_index = j
                cmp += 1
            if flag:
                print(f"\tMin index: {min_index}")
                print(f"\tExchanging {array[ind]} <> {array[min_index]} ({ind} <>
{min_index})")
            (array[ind], array[min_index]) = (array[min_index], array[ind])
            exc += 1

            if flag:
                print(ind+1, Main.show(array), cmp, exc)

            print("\t[?] Compares count: %d" % cmp)
            print("\t[?] Exchange count: %d" % exc)

            compare_count += cmp
            exchange_count += exc
            cmp = 0
            exc = 0

        print("\t[!] Total compares count: %d" % compare_count)
        print("\t[!] Total exchange count: %d" % exchange_count)
        print(f"\t[!] Exchanges to compares count: {round(exchange_count /
compare_count * 100)}%")
        print("\t--- %.5s seconds ---" % (time()-start_time))

        return array


# prints menu
def Menu():
```

```
    print("""
    - 0. Show menu
    - 1. Set  array
    - 2. Show array
    - 3. Gen  array
    - 4. Buble sort
    - 5. Selection sort
    - 6. Exit
    """)


# main function
def main():
    op = 0
    arr = Main()
    # infinity loop
    while True:
        match op:
            case 0:
                Menu()
            case 1:
                arr = list(map(int, input("Enter array by space: ").split(" ")))
            case 2:
                print(Main.show(arr))
            case 3:
                start = int(input("Enter start: "))
                end = int(input("Enter end: "))
                step = int(input("Enter step: "))
                startNum = int(input("Enter start number: "))
                endNum = int(input("Enter end number: "))
                orderBy = int(input("Enter order by (-1 - descending, 1 - ascending): "))
                arr = Main.genArray(arr,start,end,step,startNum,endNum,orderBy)

                print("\nResult: ", Main.show(arr), end="")
            case 4:
                print("Bubble sort")
                print("\nResult: ", Main.show(Main.buble_sort(arr)), end="")
            case 5:
                print("Selection sort")
                print("\nResult: ", Main.show(Main.selection_sort(arr)), end="")
            case 6:
                print("Exiting...")
                break
            case _:
```

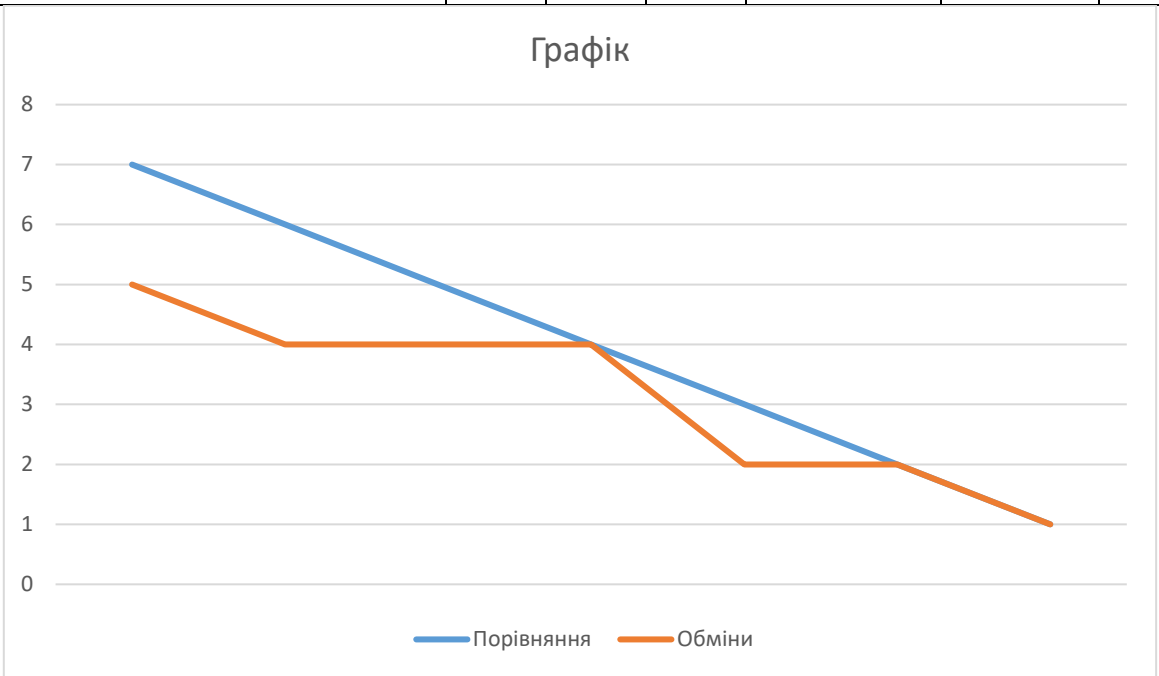
```
            print("Invalid operation")
        op = int(input("\nEnter operation: "))


# if main -> run program
if __name__ == "__main__":
    main()
    print("Program finished")
```

Частина 3. Приклад сортування бульбашкою

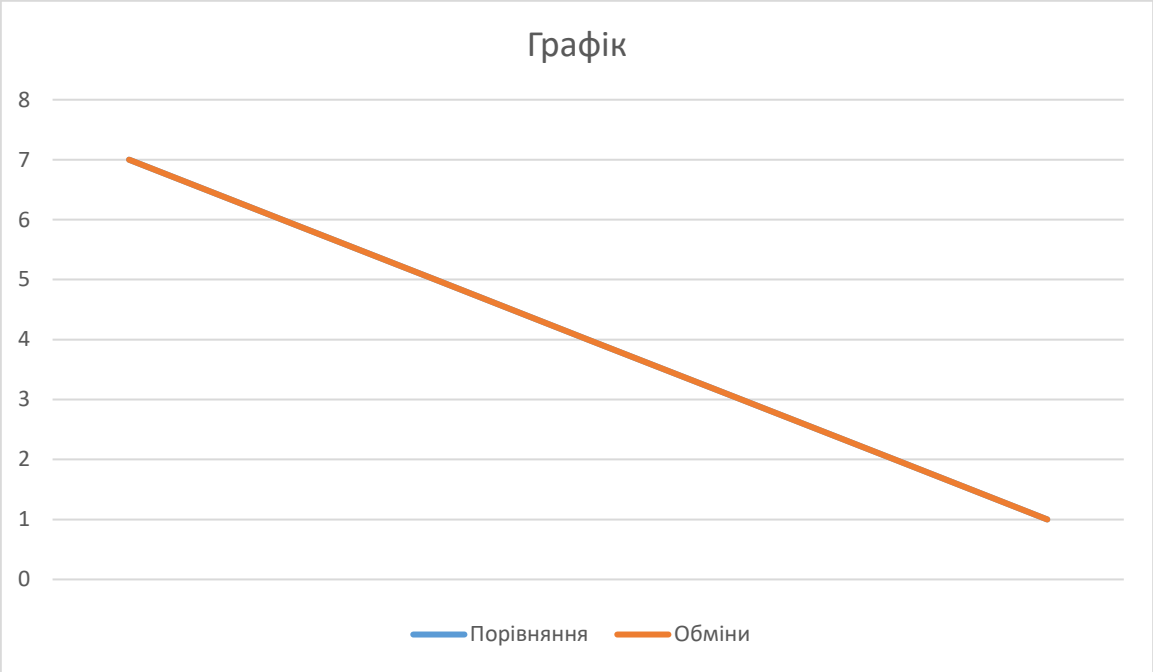
Таблиця №1

I	Стан послідовності								Порівнянь	Обмінів	Сума
1	327	450	502	211	293	83	72	528	7	5	12
2	327	450	211	293	83	72	502	528	6	4	10
3	327	211	293	83	72	450	502	528	5	4	9
4	211	293	83	72	327	450	502	528	4	4	8
5	211	83	72	293	327	450	502	528	3	2	5
6	83	72	211	293	327	450	502	528	2	2	4
7	72	83	211	293	327	450	502	528	1	1	2
									28	22	50



Таблиця №2

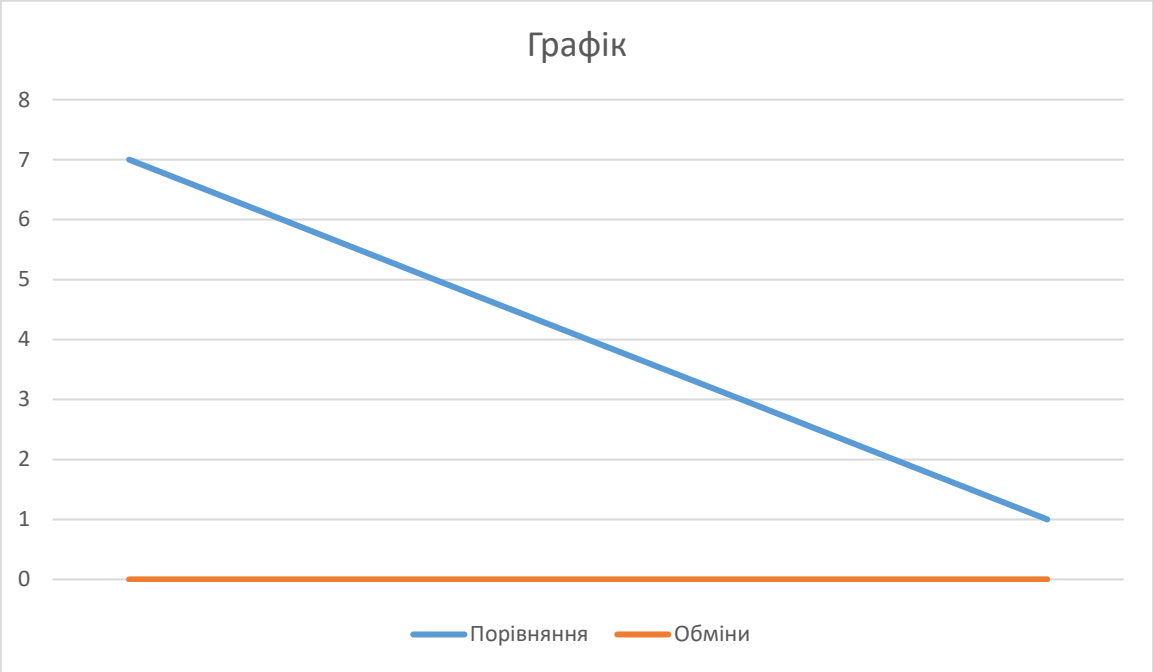
I	Стан послідовності								Порівнянь	Обмінів	Сума
1	502	450	327	293	211	83	72	528	7	7	14
2	450	327	293	211	83	72	502	528	6	6	12
3	327	293	211	83	72	450	502	528	5	5	10
4	293	211	83	72	327	450	502	528	4	4	8
5	211	83	72	293	327	450	502	528	3	3	6
6	83	72	211	293	327	450	502	528	2	2	4
7	72	83	211	293	327	450	502	528	1	1	2
									28	28	56



Таблиця №3

Масив: [327 450 502 211 293 83 72 528]

I	Стан послідовності								Порівнянь	Обмінів	Сума
1	72	83	211	293	327	450	502	528	7	0	7
2	72	83	211	293	327	450	502	528	6	0	6
3	72	83	211	293	327	450	502	528	5	0	5
4	72	83	211	293	327	450	502	528	4	0	4
5	72	83	211	293	327	450	502	528	3	0	3
6	72	83	211	293	327	450	502	528	2	0	2
7	72	83	211	293	327	450	502	528	1	0	1
									28	0	28

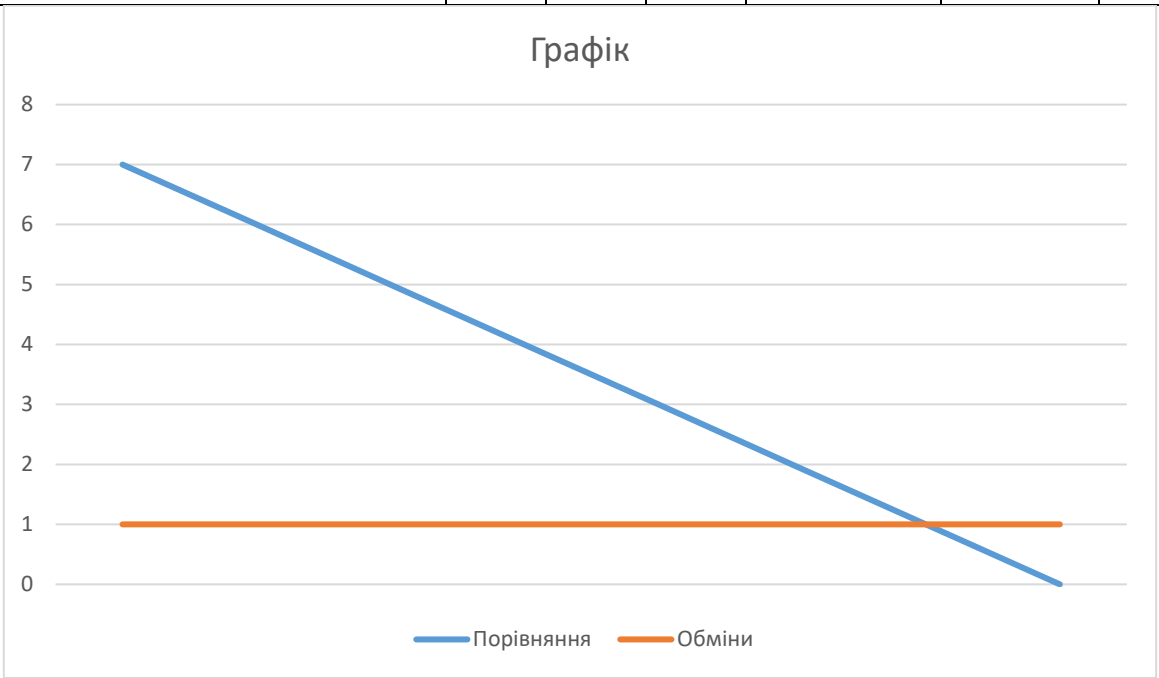


Частина 4. Приклад сортування вибором

Таблиця №4

Масив: [528 502 450 327 293 211 83 72]

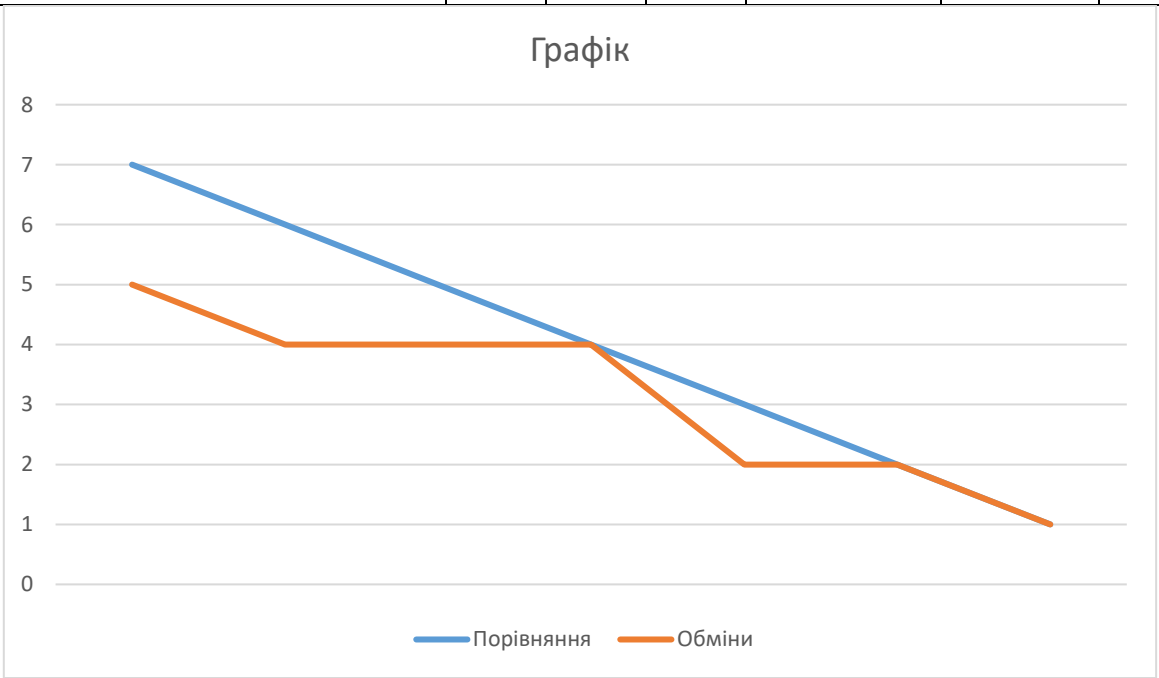
I	Стан послідовності								Порівнянь	Обмінів	Сума
1	72	450	502	211	293	83	327	528	7	1	8
2	72	83	502	211	293	450	327	528	6	1	7
3	72	83	211	502	293	450	327	528	5	1	6
4	72	83	211	293	502	450	327	528	4	1	5
5	72	83	211	293	327	450	502	528	3	1	4
6	72	83	211	293	327	450	502	528	2	1	3
7	72	83	211	293	327	450	502	528	1	1	2
8	72	83	211	293	327	450	502	528	0	1	1
									28	8	36



Таблиця №5

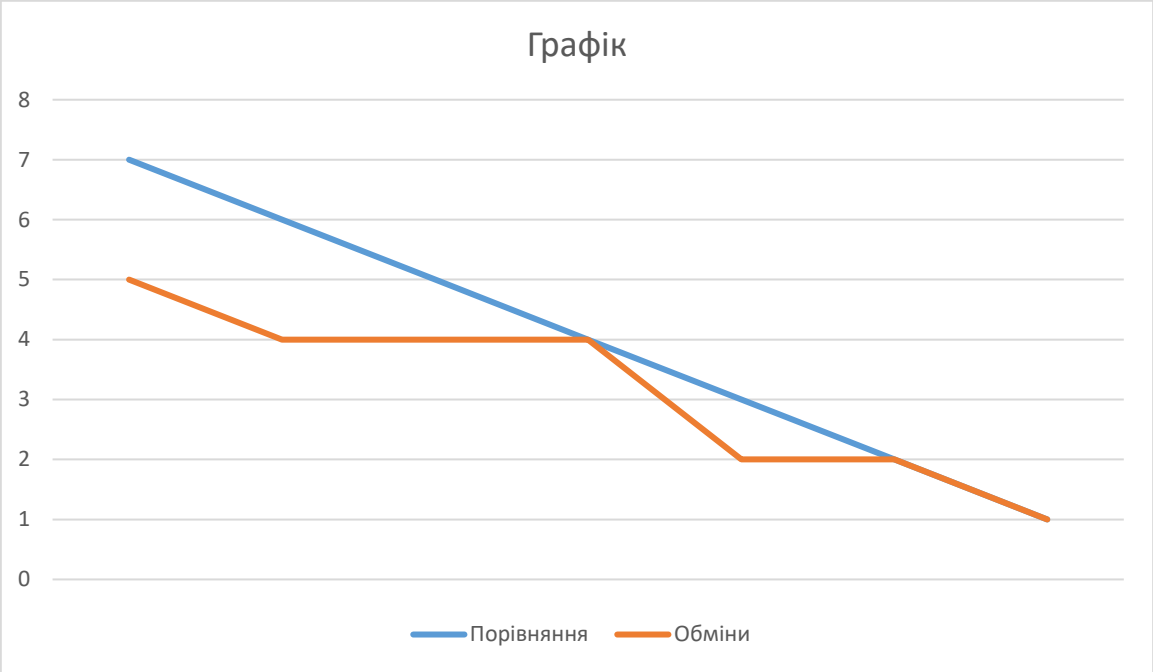
Масив: [72 83 211 293 327 450 502 528]

I	Стан послідовності								Порівнянь	Обмінів	Сума
1	72	83	211	293	327	450	502	528	7	1	8
2	72	83	211	293	327	450	502	528	6	1	7
3	72	83	211	293	327	450	502	528	5	1	6
4	72	83	211	293	327	450	502	528	4	1	5
5	72	83	211	293	327	450	502	528	3	1	4
6	72	83	211	293	327	450	502	528	2	1	3
7	72	83	211	293	327	450	502	528	1	1	2
8	72	83	211	293	327	450	502	528	0	1	1
									28	8	36



Таблиця №6

I	Стан послідовності								Порівнянь	Обмінів	Сума
1	327	450	502	211	293	83	72	528	7	5	12
2	327	450	211	293	83	72	502	528	6	4	10
3	327	211	293	83	72	450	502	528	5	4	9
4	211	293	83	72	327	450	502	528	4	4	8
5	211	83	72	293	327	450	502	528	3	2	5
6	83	72	211	293	327	450	502	528	2	2	4
7	72	83	211	293	327	450	502	528	1	1	2
									28	22	50



Висновки

Під час цієї лабораторної роботи я досліджував два алгоритми сортування, а саме сортування бульбашкою та вибором. Алгоритм сортування бульбашкою показав є набагато швидшим за алгоритм сортування вибором, бо кожного наступного разу змінна ***i*** збільшується і цикл проходить все меншу і меншу частину масиву, а алгоритм сортування вибором кожного разу проходить по всьому масиву і вибирає найменший елемент і ставить його на початок (або найбільший). За графіками і таблицями видно, як у обох алгоритмів змінюється кількість порівнянь з кількістю замін, за допомогою моєї програми можна краще розібратися у цьому, бо програма автоматично все розподіляє і показує кожний етап сортування, швидкість виконання та відношення порівнянь до замін.