

Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний
інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки

Кафедра ІІІ

Звіт

з лабораторної роботи № 1 з дисципліни
«Алгоритми та структури даних 2. Структури даних»

„Проектування і аналіз алгоритмів внутрішнього сортування”

Виконав(ла)

ІІІ-15 Плугатирьов Дмитро Валерійович
(шифр, прізвище, ім'я, по батькові)

Перевірив

Соколовський Владислав Володимирович
(прізвище, ім'я, по батькові)

Київ 2022

ЗМІСТ

1	МЕТА ЛАБОРАТОРНОЇ РОБОТИ.....	2
2	ЗАВДАННЯ.....	2
3	ВИКОНАННЯ	4
3.1	АНАЛІЗ АЛГОРИТМУ НА ВІДПОВІДНІСТЬ ВЛАСТИВОСТЯМ	4
3.2	ПСЕВДОКОД АЛГОРИТМУ.....	5
3.3	АНАЛІЗ ЧАСОВОЇ СКЛАДНОСТІ.....	9
3.4	ПРОГРАМНА РЕАЛІЗАЦІЯ АЛГОРИТМУ	10
3.4.1	Вихідний код.....	10
3.4.2	Приклад роботи.....	13
3.5	ТЕСТУВАННЯ АЛГОРИТМУ	18
3.5.1	Часові характеристики оцінювання.....	18
3.5.2	Графіки залежності часових характеристик оцінювання від розмірності масиву.....	21
	ВИСНОВОК	22
	КРИТЕРІЇ ОЦІНЮВАННЯ	24

Мета лабораторної роботи

Мета роботи – вивчити основні методи аналізу обчислювальної складності алгоритмів внутрішнього сортування і оцінити поріг їх ефективності.

Завдання

Виконати аналіз алгоритму внутрішнього сортування на відповідність наступним властивостям (таблиця 2.1):

- стійкість;
- «природність» поведінки (Adaptability);

- базуються на порівняннях;
- необхідність додаткової пам'яті (об'єму);
- необхідність в знаннях про структуру даних.

Записати алгоритм внутрішнього сортування за допомогою псевдокоду (чи іншого способу по вибору).

Провести аналіз часової складності в гіршому, кращому і середньому випадках та записати часову складність в асимптотичних оцінках.

Виконати програмну реалізацію алгоритму на будь-якій мові програмування з фіксацією часових характеристик оцінювання (кількість порівнянь, кількість перестановок, глибина рекурсивного поглиблення та інше в залежності від алгоритму).

Провести ряд випробувань алгоритму на масивах різної розмірності (10, 100, 1000, 5000, 10000, 20000, 50000 елементів) і різних наборів вхідних даних (впорядкований масив, зворотно упорядкований масив, масив випадкових чисел) і побудувати графіки залежності часових характеристик оцінювання від розмірності масиву, нанести на графік асимптотичну оцінку гіршого і кращого випадків для порівняння.

Зробити порівняльний аналіз двох алгоритмів.

Зробити узагальнений висновок з лабораторної роботи.

Таблиця 2.1 – Варіанти алгоритмів

№	Алгоритм сортування
1	Сортування бульбашкою
2	Сортування гребінцем («розчіскою»)

ВИКОНАННЯ

1.1 Аналіз алгоритму на відповідність властивостям

Аналіз алгоритму сортування бульбашкою на відповідність властивостям наведено в таблиці 3.1.

Таблиця 3.1 – Аналіз алгоритму на відповідність властивостям

Властивість	Сортування бульбашкою
Стійкість	Стійкий
«Природність» поведінки (Adaptability)	Природна
Базуються на порівняннях	Так
Необхідність в додатковій пам'яті (об'єм)	Так. В даному прикладі бралися 4 байти пам'яті на додаткову цілочисельну змінну
Необхідність в знаннях про структури даних	Так. В даному випадку використовувався звичайний масив

Аналіз алгоритму сортування гребінцем на відповідність властивостям наведено в таблиці 3.2.

Таблиця 3.2 – Аналіз алгоритму на відповідність властивостям

Властивість	Сортування гребінцем
Стійкість	Відсутня
«Природність» поведінки (Adaptability)	Природна
Базуються на порівняннях	Так
Необхідність в додатковій пам'яті (об'єм)	Так. В даному прикладі бралися 12 байтів пам'яті на додаткову цілочисельну та з фіксованою крапкою змінні.

Необхідність в знаннях про структури даних	Так. В даному випадку використовувався звичайний масив
--	--

1.2 Псевдокод алгоритму сортування «бульбашкою»

Крок 1. Визначити основні дії.

Крок 2. Прохід по всій послідовності арифметичним циклом певну кількість разів.

Крок 3. Прохід по всіх елементах послідовності.

Крок 4. Порівняння елементів, які розташовані поруч, та їх обмін місцями за певної умови.

Крок 1

початок

прохід по всій послідовності арифметичним циклом певну кількість разів

прохід по всіх елементах послідовності

порівняння елементів, які розташовані поруч, та їх обмін місцями за певної умови

кінець

Крок 2

початок

повторити для y від 0 до $\text{seq.size} - 1$

прохід по всіх елементах послідовності

порівняння елементів, які розташовані поруч, та їх обмін місцями за певної умови

все повторити

кінець

Крок 3

початок

повторити для у від 0 до seq.size - 1

повторити для і від 1 до seq.size

порівняння елементів, які розташовані поруч, та їх

обмін місцями за певної умови

все повторити

все повторити

кінець

Крок 4

початок

повторити для у від 0 до seq.size - 1

повторити для і від 1 до seq.size

якщо seq[i - 1] > seq[i]

то

temp := seq[i]

seq[i] := seq[i - 1]

seq[i - 1] := temp

все якщо

все повторити

все повторити

кінець

Псевдокод алгоритму сортування «гребінцем»

Крок 1. Визначити основні дії.

Крок 2. Виконання ітерацій по всій послідовності алгоритмом «бульбашка» до моменту зменшення кроку до 1 або менше.

Крок 3. Перебір елементів послідовності для подальших дій з ними.

Крок 4. Порівняння елементів, які розташовані поруч, та їх обмін місцями за певної умови.

Основна програма

Крок 1

початок

виконання ітерацій по всій послідовності алгоритмом «бульбашка»

до моменту зменшення кроку до 1 або менше

перебір елементів послідовності для подальших дій з ними
порівняння елементів, які розташовані поруч, та їх обмін місцями за певної умови

кінець

Крок 2

початок

factor := 1.2473309

step := array.size – 1

повторити

поки step >= 1

перебір елементів послідовності для подальших дій з ними

порівняння елементів, які розташовані поруч, та їх обмін місцями за певної умови

step := (int)(step / factor)

все повторити

кінець

Крок 3

початок

factor := 1.2473309

step := array.size – 1

```

повторити
поки step >= 1
    для i від 0 до array.size
        повторити
            порівняння елементів, які розташовані поруч, та їх
            обмін місцями за певної умови
        все повторити
        step := (int)(step / factor)
    все повторити
кінець

```

Крок 4

```

початок
    factor := 1.2473309
    step := array.size – 1
    повторити
    поки step >= 1
        для i від 0 до array.size
            повторити
                якщо array[i] > array[i + step]
                    то
                        Swap(array[i], array[i + step])
                все якщо
            все повторити
            step := (int)(step / factor)
        все повторити
    кінець

```


Підпрограми

Swap(value1, value2)

temp := value1

value1 := value2

value2 := temp

кінець

1.3 Аналіз часової складності

Метод сортування «бульбашкою»

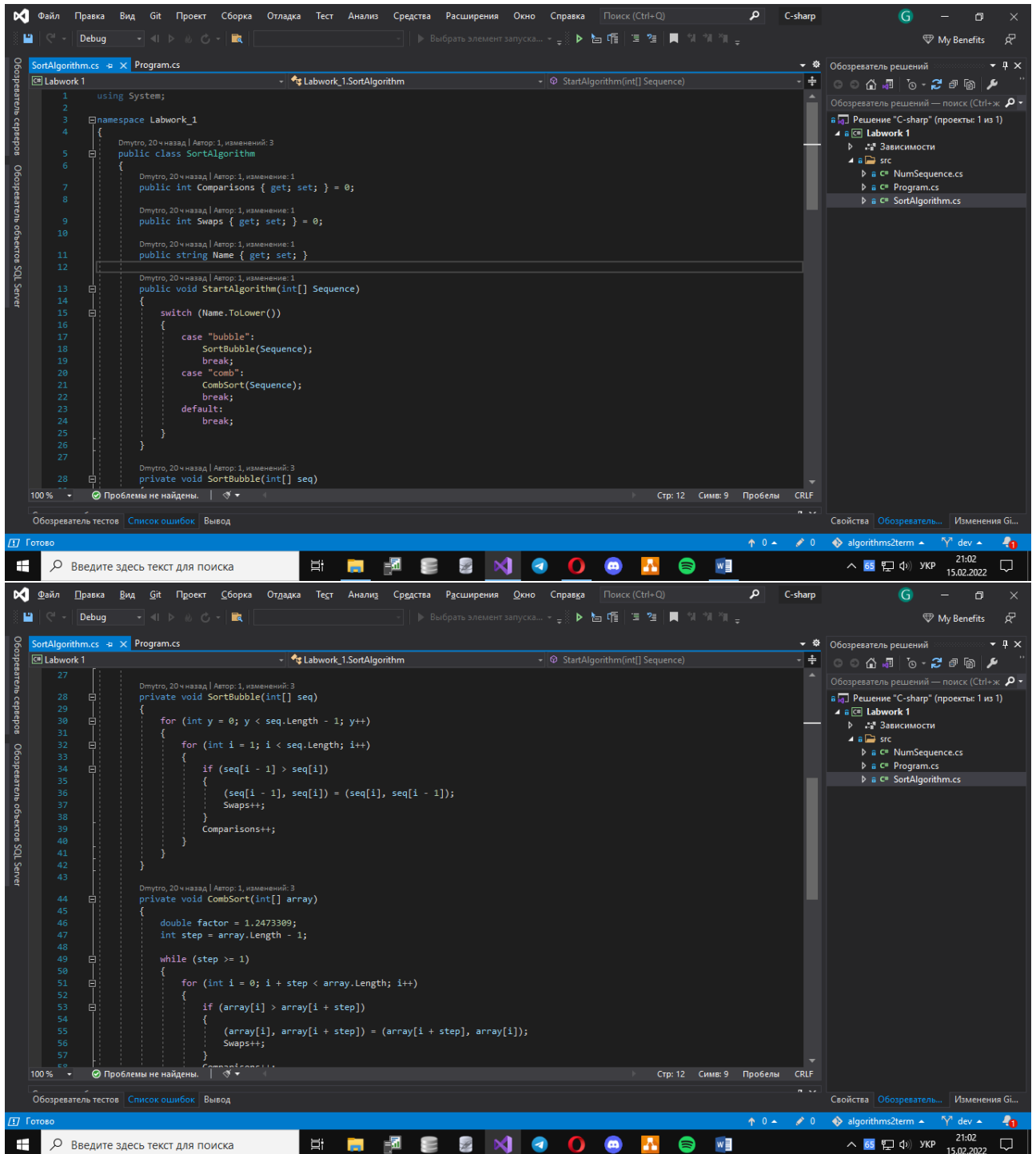
К-сть елементів	Час		
	Гірший $O(n^2)$	Середньостатистичний $O(n^2)$	Кращий $O(n)$
10	00:00:00.0000056	00:00:00.0000057	00:00:00.0000054
100	00:00:00.0000271	00:00:00.0000339	00:00:00.0000195
1000	00:00:00.0030748	00:00:00.0028864	00:00:00.0017776

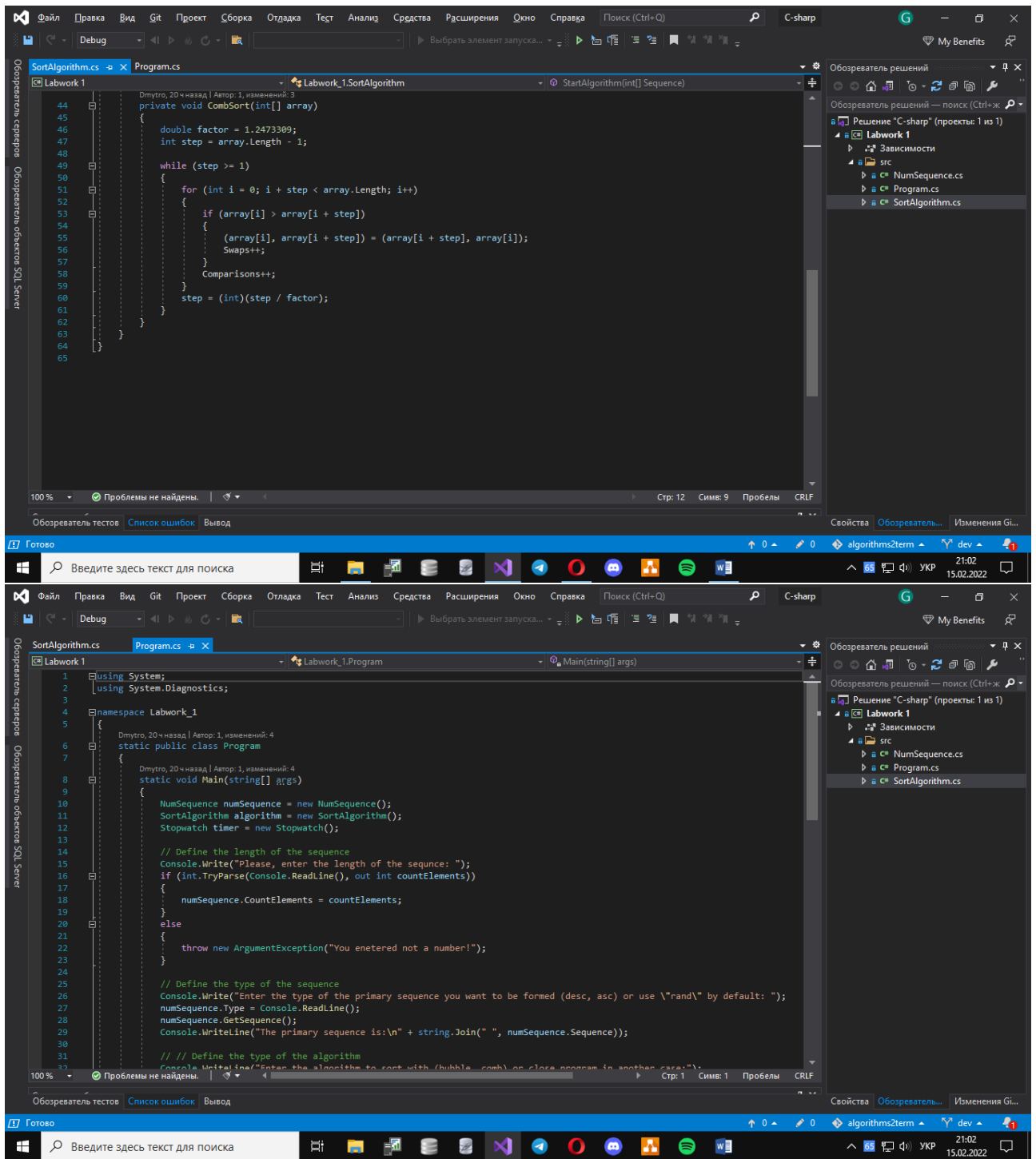
Метод сортування “гребінцем”

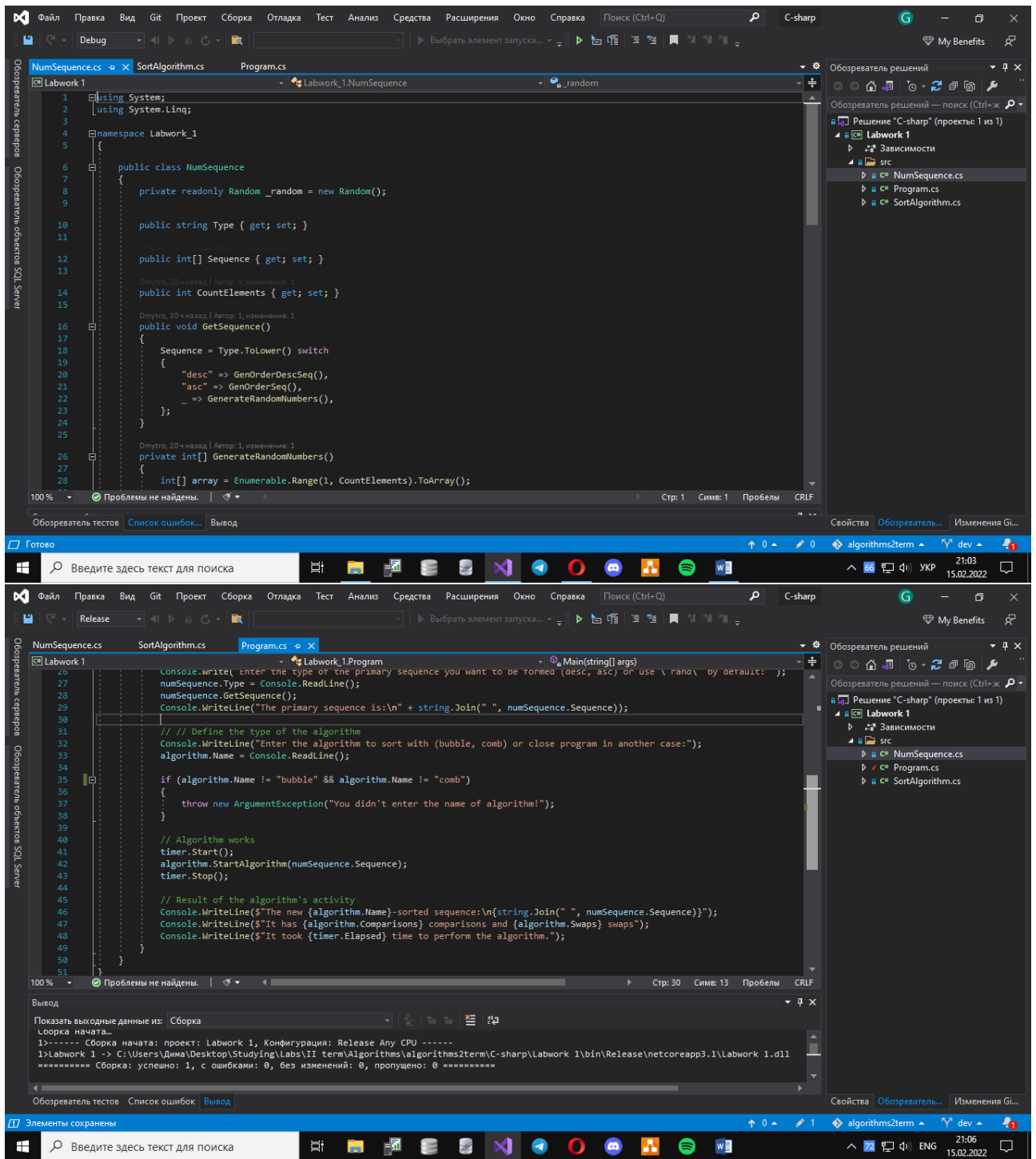
К-сть елементів	Час		
	Гірший $O(n^2)$	Середньостатистичний $O(n \log n)$	Кращий $O(n)$
10	00:00:00.0000083	00:00:00.0000087	00:00:00.0000054
100	00:00:00.0000280	00:00:00.0000116	00:00:00.0000074
1000	00:00:00.0030445	00:00:00.0001028	00:00:00.0000431

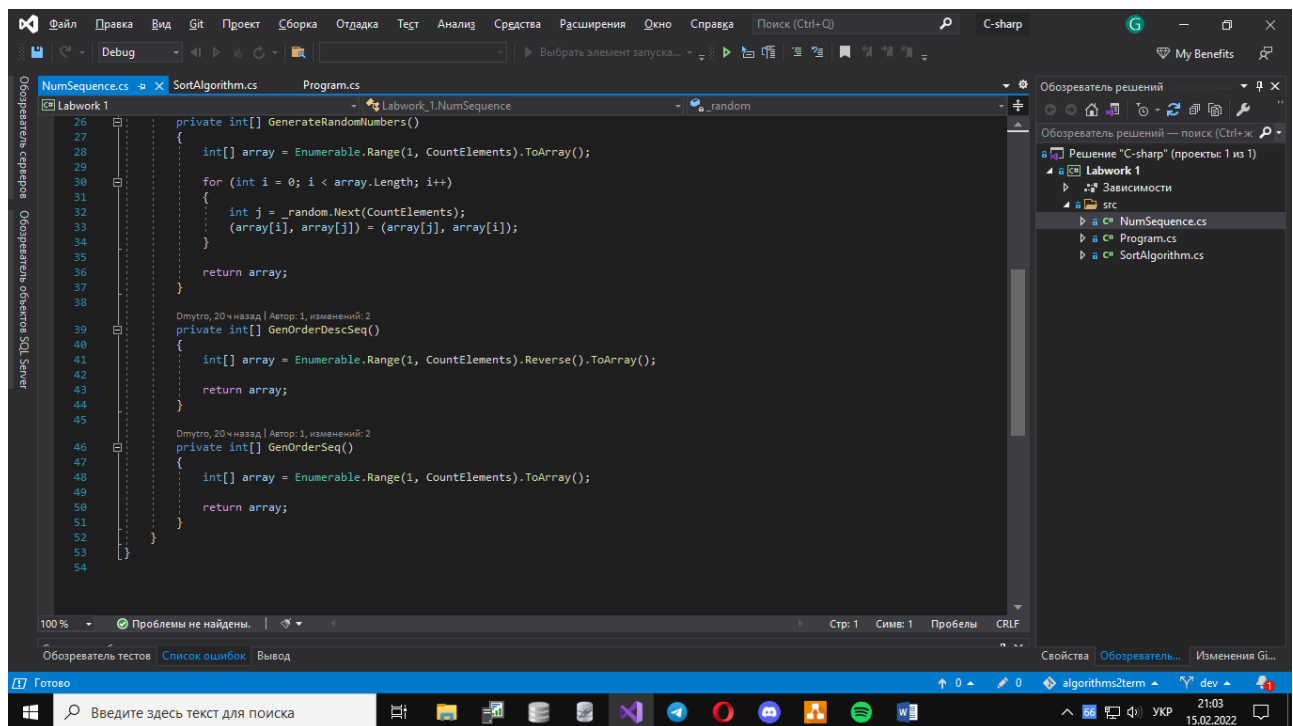
1.4 Програмна реалізація алгоритму

1.4.1 Вихідний код

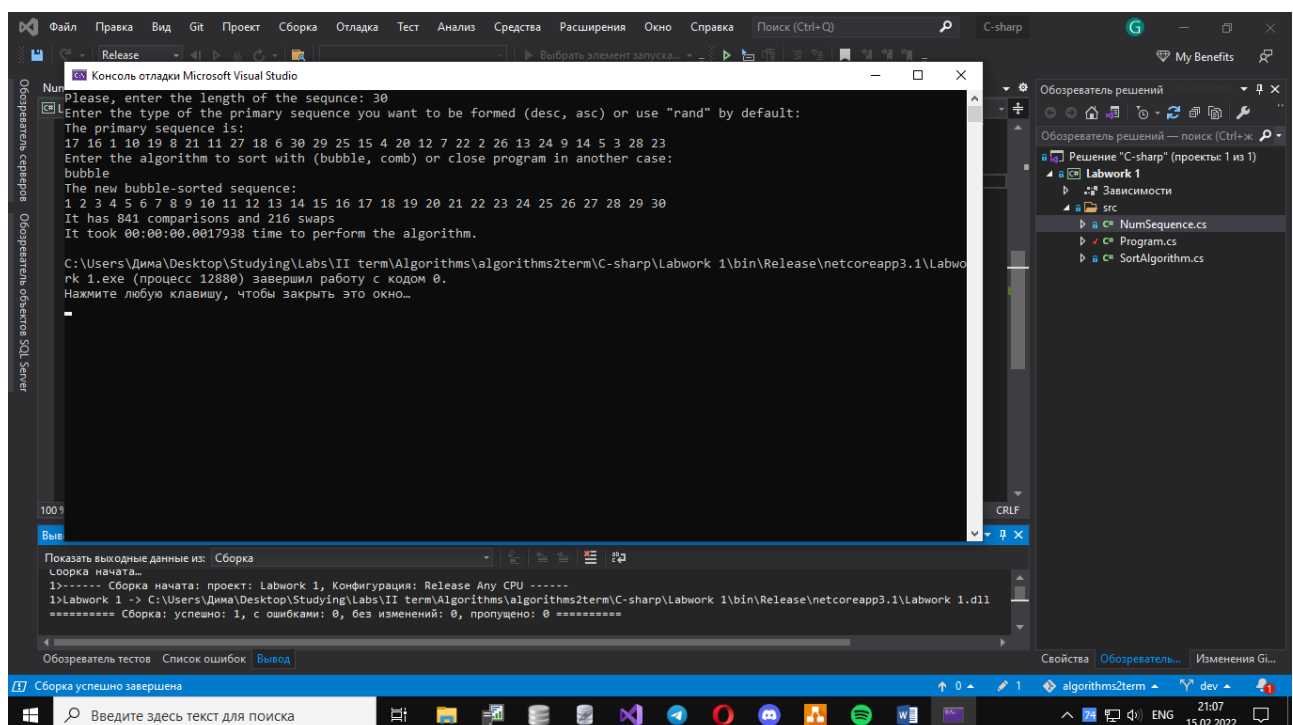


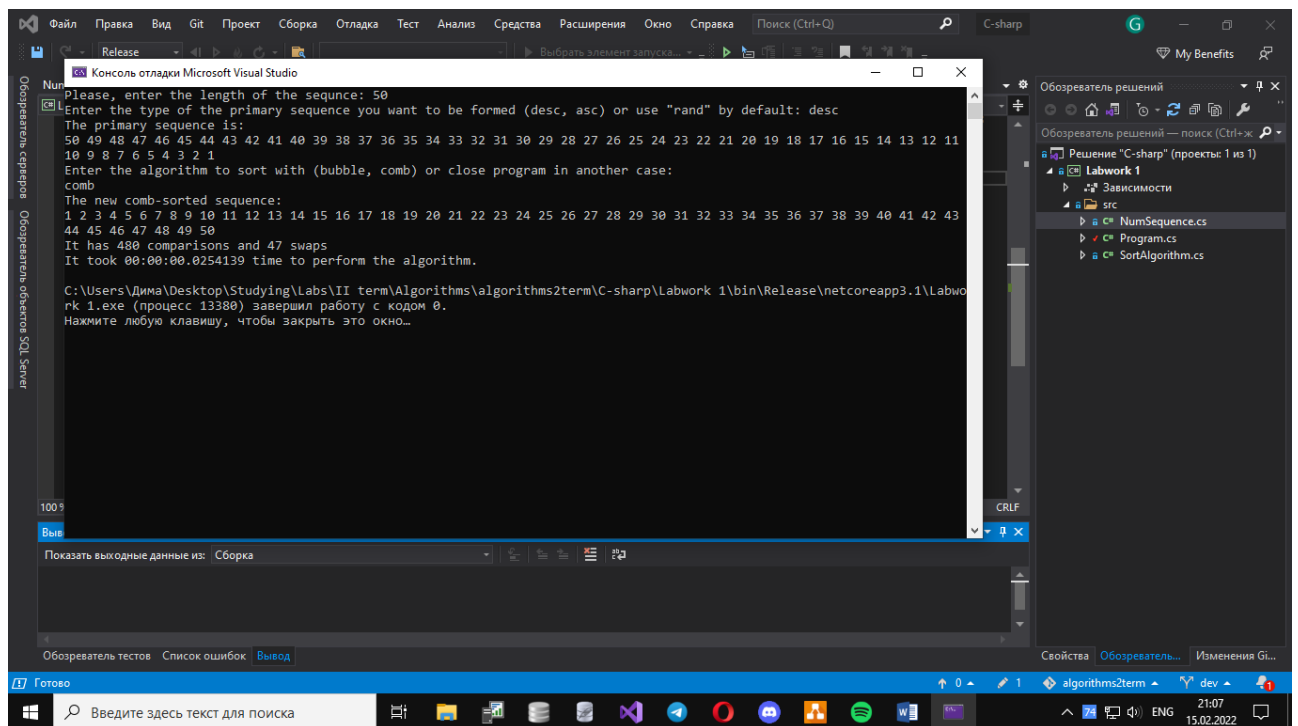






1.4.2 Приклад роботи

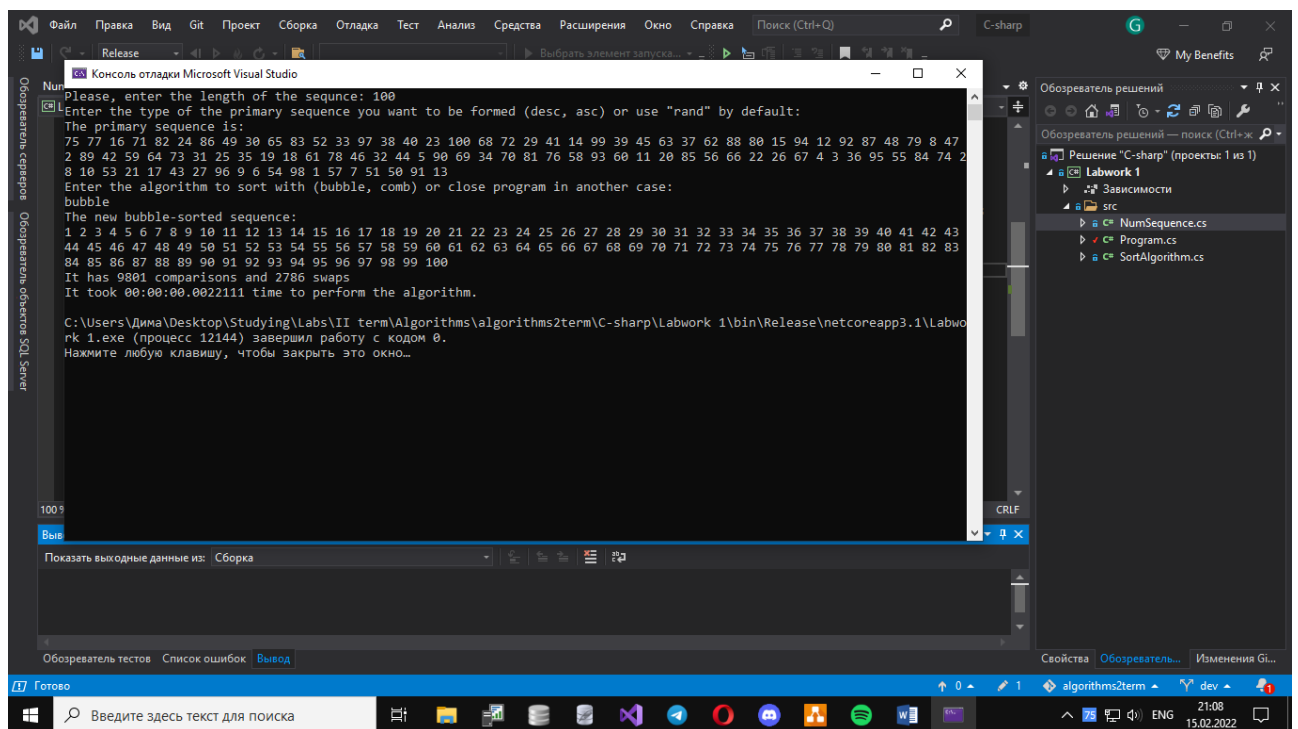




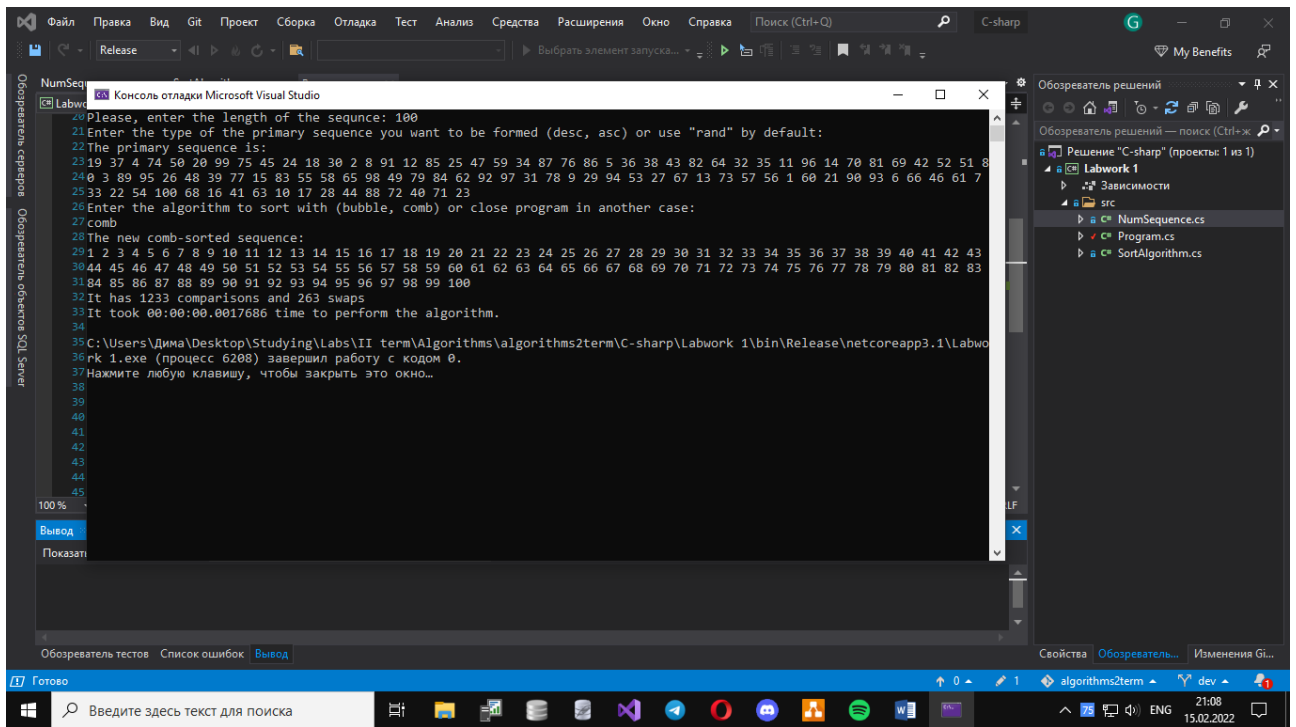
На рисунках 3.1 і 3.2 показані приклади роботи програми сортування масивів на 100 і 1000 елементів відповідно.

Рисунок 3.1 – Сортування масиву на 100 елементів

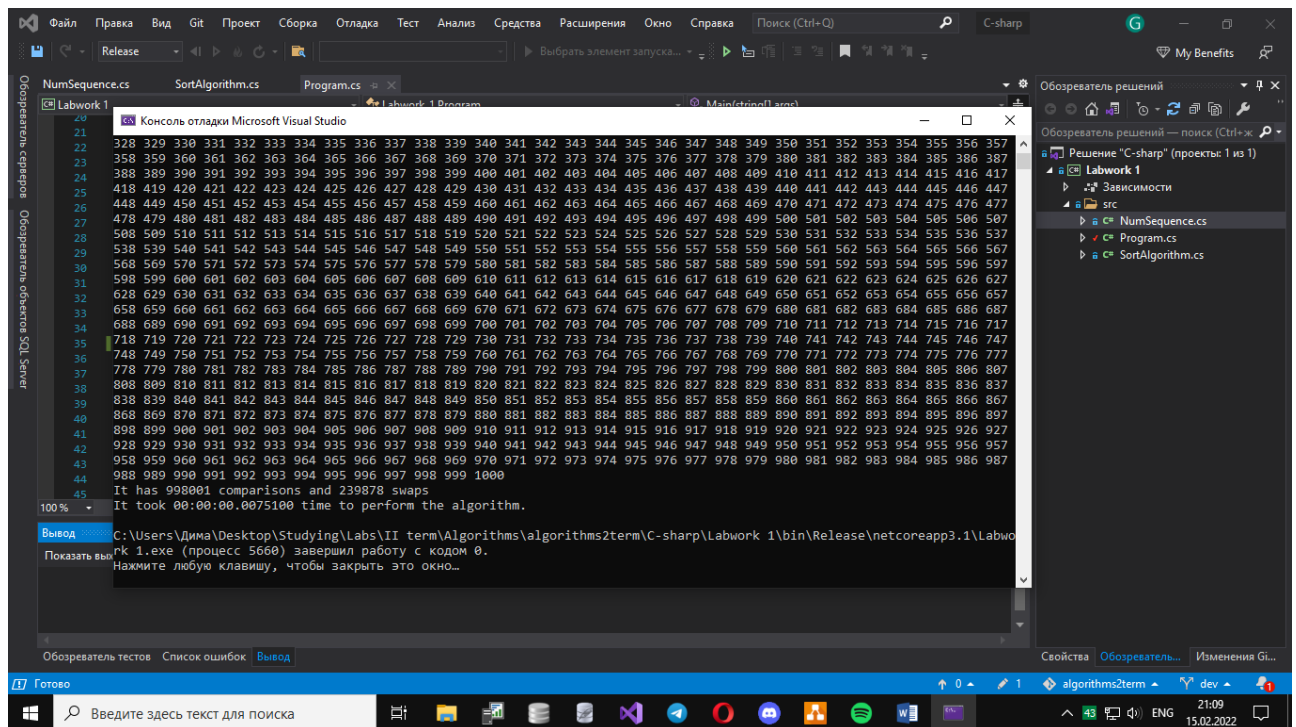
«Бульбашка»

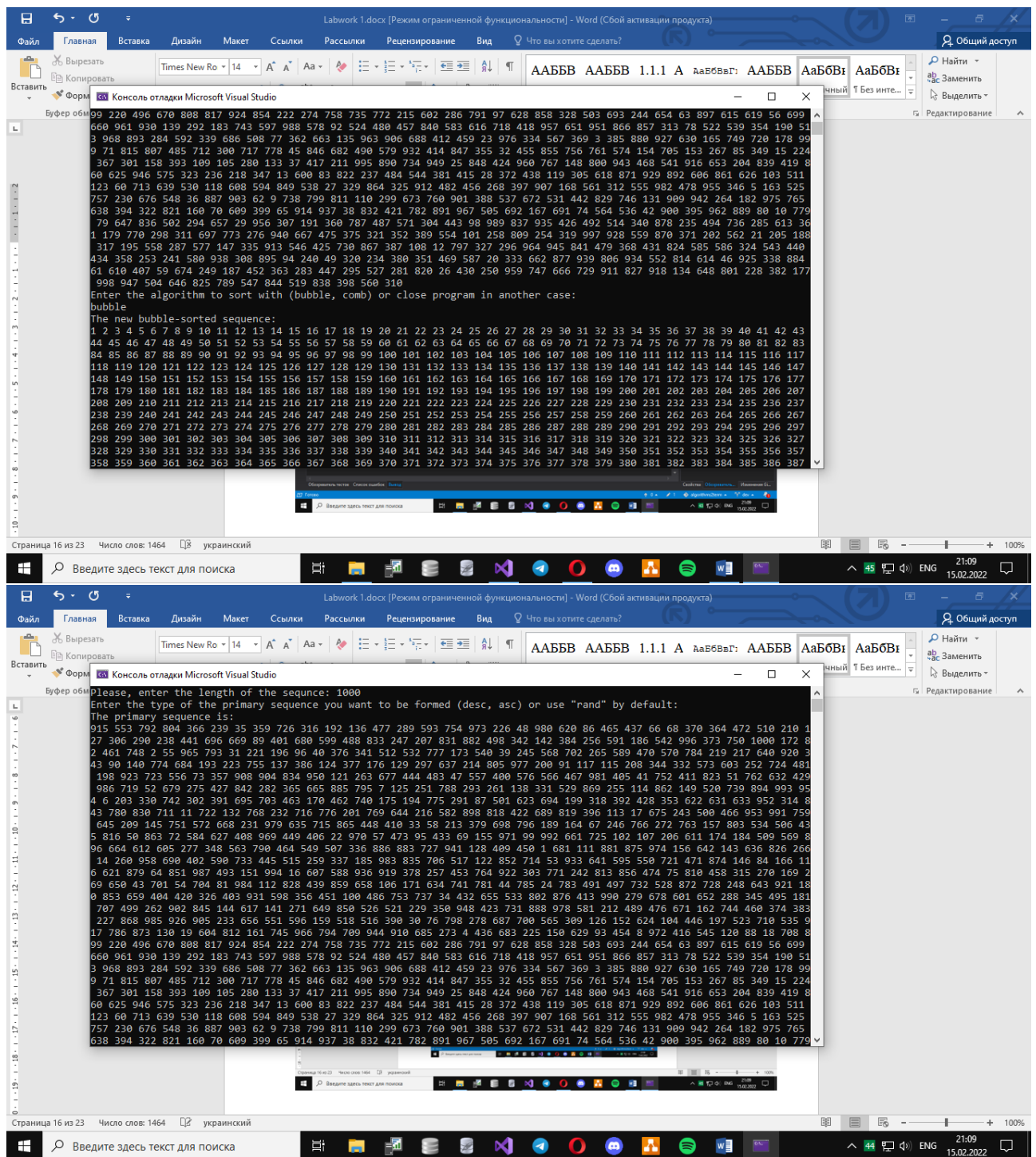


«Гребінець»

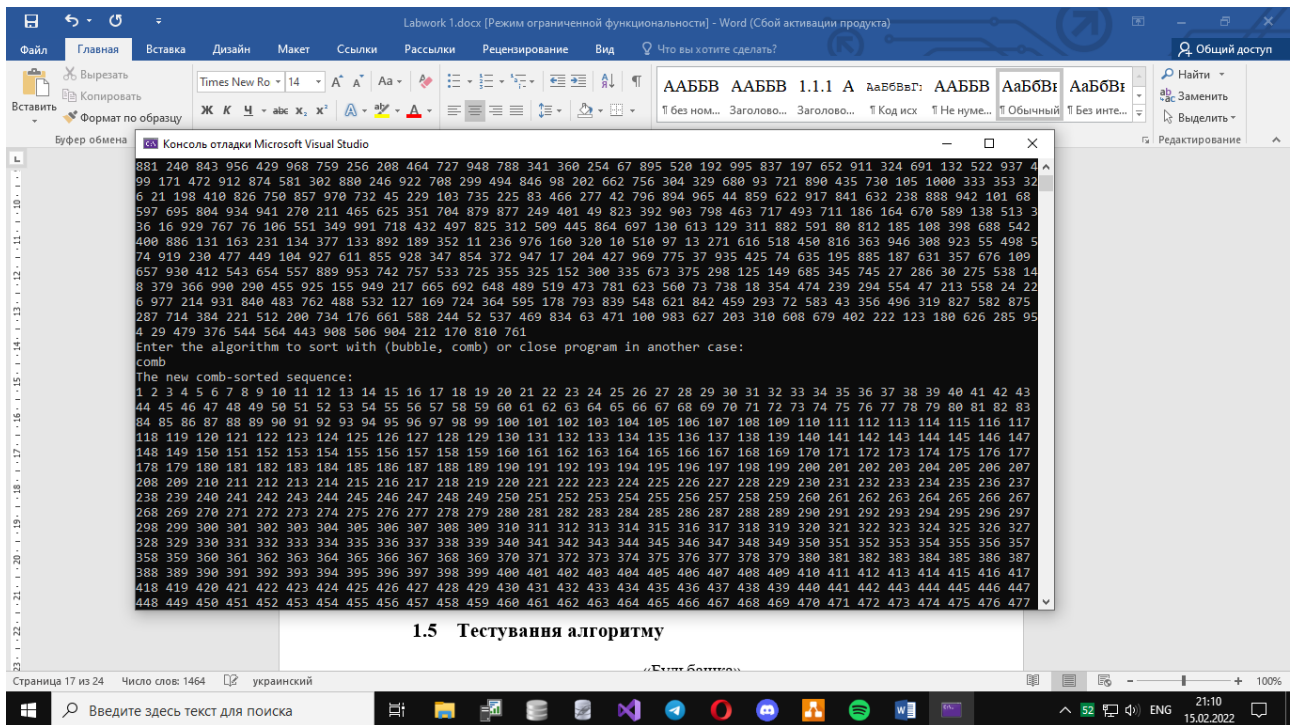
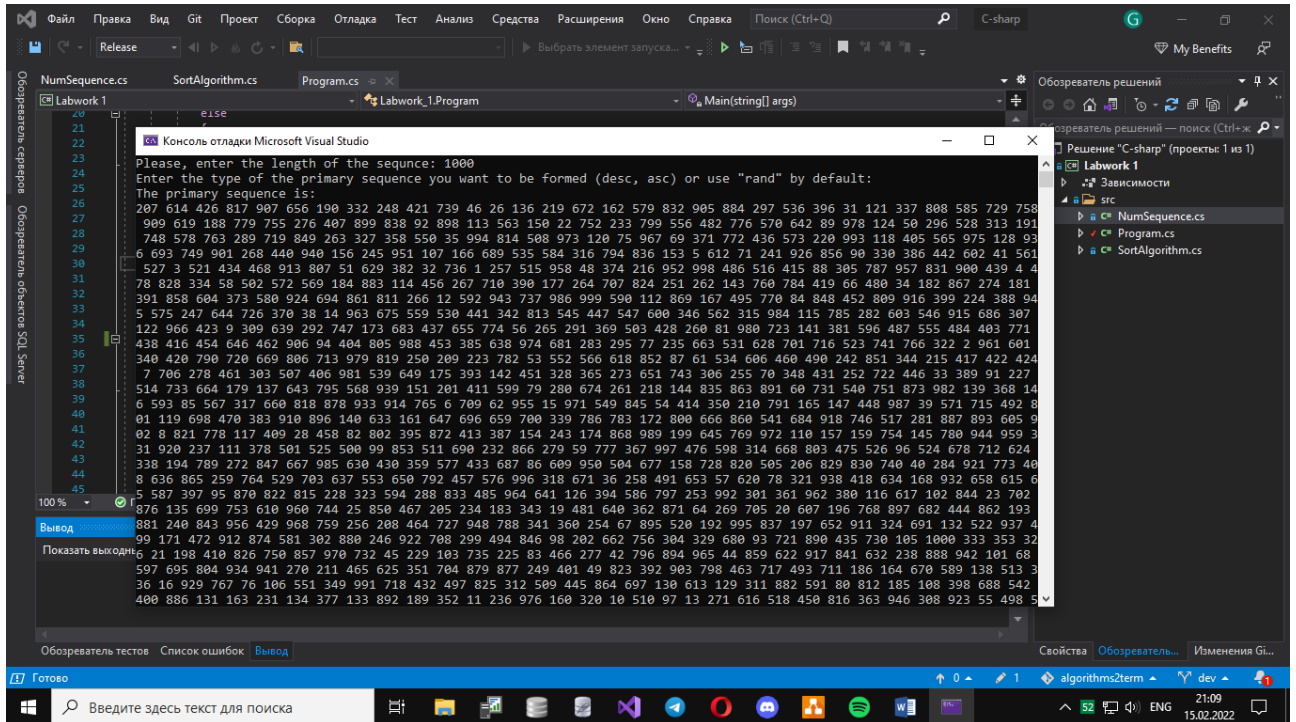


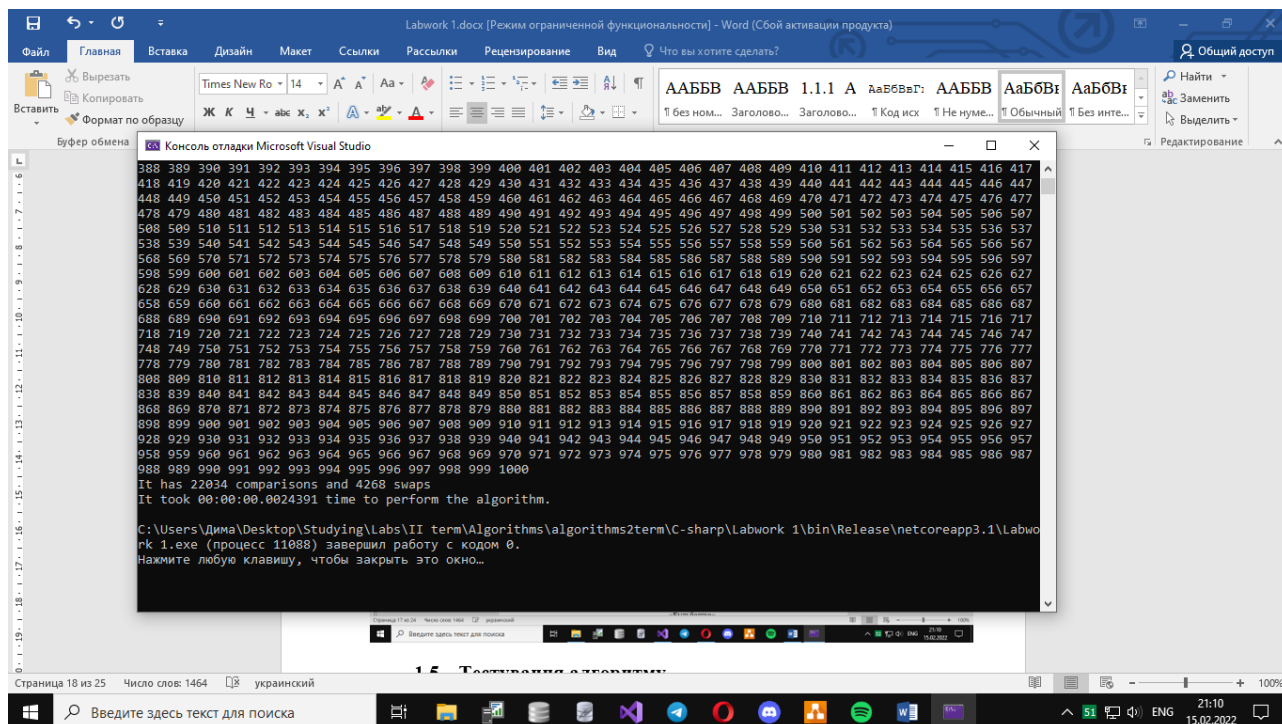
**Рисунок 3.2 – Сортування масиву на 1000 елементів
«Бульбашка»**





«Гребінець»





1.5 Тестування алгоритму

1.5.1 Часові характеристики оцінювання

В таблиці 3.2 наведені характеристики оцінювання числа порівнянь і числа перестановок алгоритму сортування бульбашки для масивів різної розмірності, коли масив містить упорядковану послідовність елементів.

Таблиця 3.2 – Характеристики оцінювання алгоритму сортування бульбашкою для упорядкованої послідовності елементів у масиві

Розмірність масиву	Число порівнянь	Число перестановок
10	81	0
100	9801	0
1000	998001	0
5000	24990001	0
10000	99980001	0
20000	399960001	0
50000	2499900001	0

Таблиця 3.2.2 – Характеристики оцінювання алгоритму сортування гребінцем для упорядкованої послідовності елементів у масиві

Розмірність масиву	Число порівнянь	Число перестановок
10	39	0
100	1233	0
1000	22034	0
5000	144865	0
10000	329653	0
20000	719246	0
50000	1997961	0

В таблиці 3.3 наведені характеристики оцінювання числа порівнянь і числа перестановок алгоритму сортування бульбашки для масивів різної розмірності, коли масиви містять зворотно упорядковану послідовність елементів.

Таблиця 3.3 – Характеристики оцінювання алгоритму сортування бульбашки для зворотно упорядкованої послідовності елементів у масиві.

Розмірність масиву	Число порівнянь	Число перестановок
10	81	45
100	9801	4950
1000	998001	499500
5000	24990001	12497500
10000	99980001	49995000
20000	399960001	199990000
50000	2499900001	1249975000

Таблиця 3.3.2 – Характеристики оцінювання алгоритму сортування гребінця для зворотно упорядкованої послідовності елементів у масиві.

Розмірність масиву	Число порівнянь	Число перестановок
10	39	5
100	1233	106
1000	22034	1528
5000	144865	9110
10000	329653	19164
20000	719246	40636
50000	1997961	109794

У таблиці 3.4 наведені характеристики оцінювання числа порівнянь і числа перестановок алгоритму сортування бульбашки для масивів різної розмірності, масиви містять випадкову послідовність елементів.

Таблиця 3.4 – Характеристика оцінювання алгоритму сортування бульбашкою за спаданням для випадкової послідовності елементів у масиві.

Розмірність масиву	Число порівнянь	Число перестановок
10	81	24
100	9801	2762
1000	998001	262360
5000	24990001	6393886
10000	99980001	26013701
20000	399960001	103898756
50000	2499900001	646797020

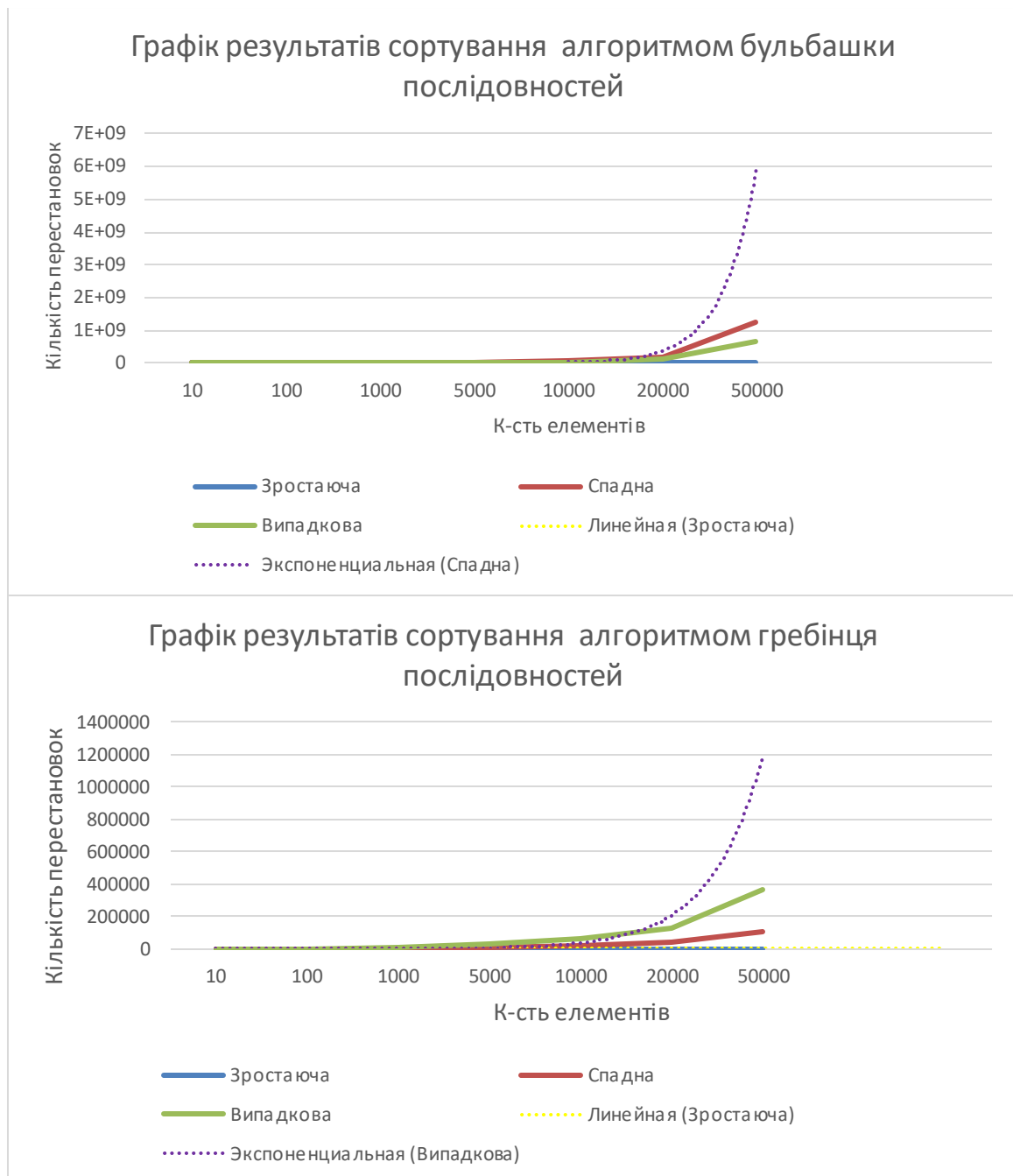
Таблиця 3.4.2 – Характеристика оцінювання алгоритму сортування гребінцем для випадкової послідовності елементів у масиві.

Розмірність масиву	Число порівнянь	Число перестановок
10	39	5
100	1233	219
1000	22034	4206

5000	144865	27094
10000	329653	59774
20000	719246	130433
50000	1997961	367660

1.5.2 Графіки залежності часових характеристик оцінювання від розмірності масиву

На рисунку 3.3 показані графіки залежності часових характеристик оцінювання від розмірності масиву для випадків, коли масиви містять упорядковану послідовність елементів (синій графік), коли масиви містять зворотно упорядковану послідовність елементів (червоний графік), коли масиви містять випадкову послідовність елементів (зелений графік), також показані асимптотичні оцінки гіршого (фіолетовий графік) і кращого (жовтий графік) випадків для порівняння. Рисунок 3.3 – Графіки залежності часових характеристик оцінювання



Висновок

При виконанні даної лабораторної роботи я скористався таймером, вимірюючи час виконання алгоритму, закріпив свої знання з правил використання алгоритмів та їх доречності в залежності від ситуації. Мною було проведено тестування алгоритму сортування «бульбашкою» та «гребінцем» шляхом підстановки різних значень, які представляють кількість елементів послідовності. Як видно з отриманих результатів, алгоритм бульбашки краще працює з малою кількістю елементів, гребінця – з великою.

КРИТЕРІЇ ОЦІНЮВАННЯ

У випадку здачі лабораторної роботи до 21.02.2022 включно максимальний бал дорівнює – 5. Після 21.02.2022 – 28.02.2022 максимальний бал дорівнює – 2,5. Після 28.02.2022 робота не приймається

Критерії оцінювання у відсотках від максимального балу:

- аналіз алгоритму на відповідність властивостям – 10%;
- псевдокод алгоритму – 15%;
- аналіз часової складності – 25%;
- програмна реалізація алгоритму – 25%;
- тестування алгоритму – 20%;
- висновок – 5%.