**Міністерство освіти і науки України**

**Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"**

**Факультет інформатики та обчислювальної техніки**

**Кафедра ІПІ**

**Звіт**

з лабораторної роботи № 1 з дисципліни

«Алгоритми та структури даних 2. Структури даних»

„ **Проектування і аналіз алгоритмів внутрішнього сортування**”

**Виконав(ла)**

(шифр, прізвище, ім'я, по батькові)

*ІП-15 Плугатирьов Дмитро Валерійович ійович*

**Перевірив**

(прізвище, ім'я, по батькові)

*Соколовський Владислав Володимирович*

Київ 2022

Зміст

[1 Мета лабораторної роботи 2](#_Toc69772242)

[2 ЗаВдання 2](#_Toc69772243)

[3 Виконання 4](#_Toc69772244)

[3.1 Аналіз алгоритму на відповідність властивостям 4](#_Toc69772245)

[3.2 Псевдокод алгоритму 5](#_Toc69772246)

[3.3 Аналіз часової складності 9](#_Toc69772247)

[3.4 Програмна реалізація алгоритму 10](#_Toc69772248)

[3.4.1 Вихідний код 10](#_Toc69772249)

[3.4.2 Приклад роботи 13](#_Toc69772250)

[3.5 Тестування алгоритму 18](#_Toc69772251)

[3.5.1 Часові характеристики оцінювання 18](#_Toc69772252)

[3.5.2 Графіки залежності часових характеристик оцінювання від розмірності масиву 21](#_Toc69772253)

[Висновок 22](#_Toc69772254)

[Критерії оцінювання 24](#_Toc69772255)

**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

## 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) | Ні |
| Необхідність в знаннях про структури даних | 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) | Ні |
| Необхідність в знаннях про структури даних | Масив |

## 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**

isSwap := true

**повторити для** y **від** 0 **до** seq.size – 1 **та** isSwap == true

прохід по всіх елементах послідовності

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

**все повторити**

**кінець**

*Крок 3*

**Початок**

isSwap := true

**повторити для** y **від** 0 **до** seq.size – 1 **та** isSwap == true

isSwap := false

**повторити для** i **від** 1 **до** seq.size - y

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

**все повторити**

**все повторити**

**кінець**

*Крок 4*

**Початок**

isSwap := true

**повторити для** y **від** 0 **до** seq.size – 1 **та** isSwap == true

isSwap := false

**повторити для** i **від** 1 **до** seq.size - y

**якщо** seq[i - 1] > seq[i]

**то**

seq[i - 1] += seq[i]

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

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

isSwap := true

**все якщо**

**все повторити**

**все повторити**

**кінець**

**Псевдокод алгоритму сортування «гребінцем»**

*Крок 1*. Визначити основні дії.

*Крок 2*. Виконання порівнянь крайових елементів, проходячи проміжком між ними по послідовності, поступово зменшуючи його модуль та задіяння алгоритму «бульбашка» за зменшення відстані між порівнюваними елементами до 1.

*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 порівнянь крайових елементів, проходячи проміжком між ними по послідовності, поступово зменшуючи його модуль та задіяння алгоритму «бульбашка» за зменшення відстані між порівнюваними елементами до 1

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**

seq[i + step] += seq[i]

seq[i] := seq[i + step] - seq[i]

seq[i + step] := seq[i + step] - seq[i]

**все якщо**

**все повторити**

step := (int)(step / factor)

**все повторити**

**кінець**

## Аналіз часової складності

**Метод сортування «бульбашкою»**

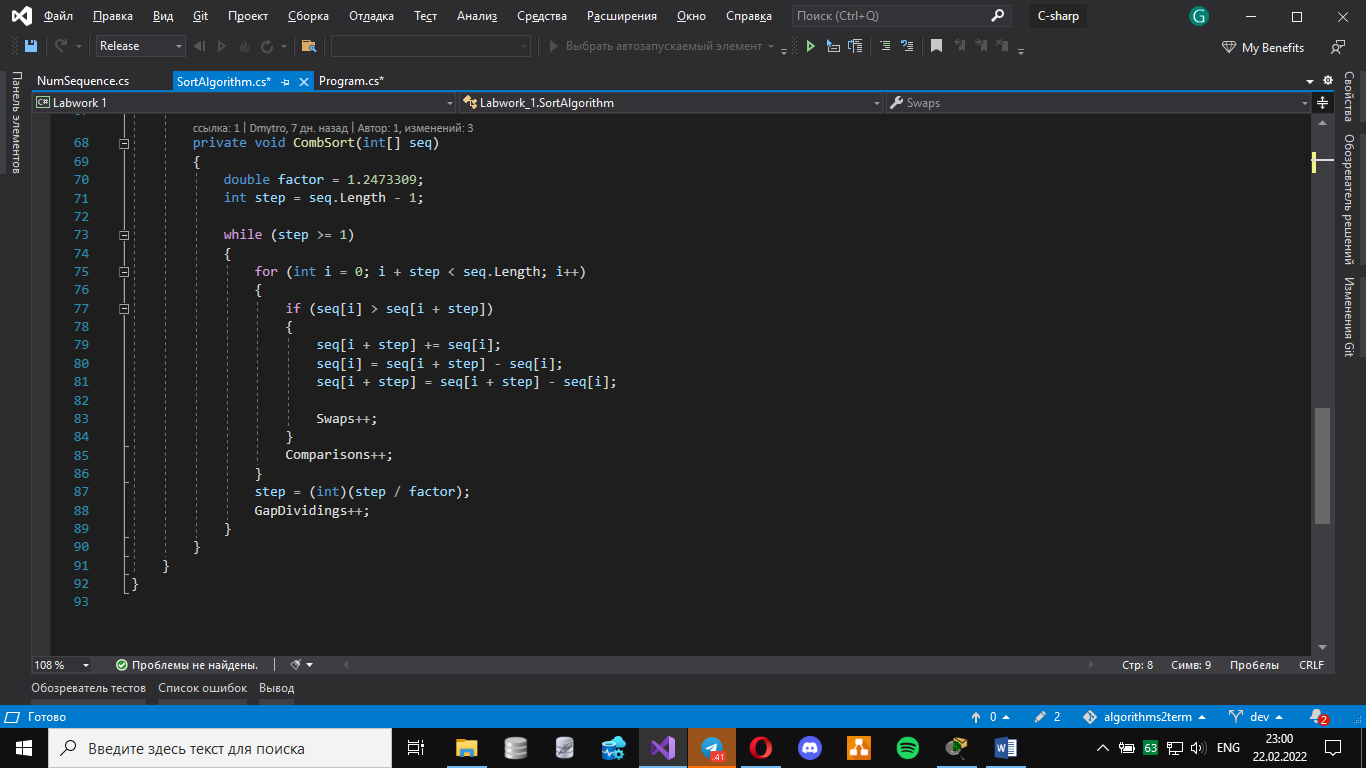
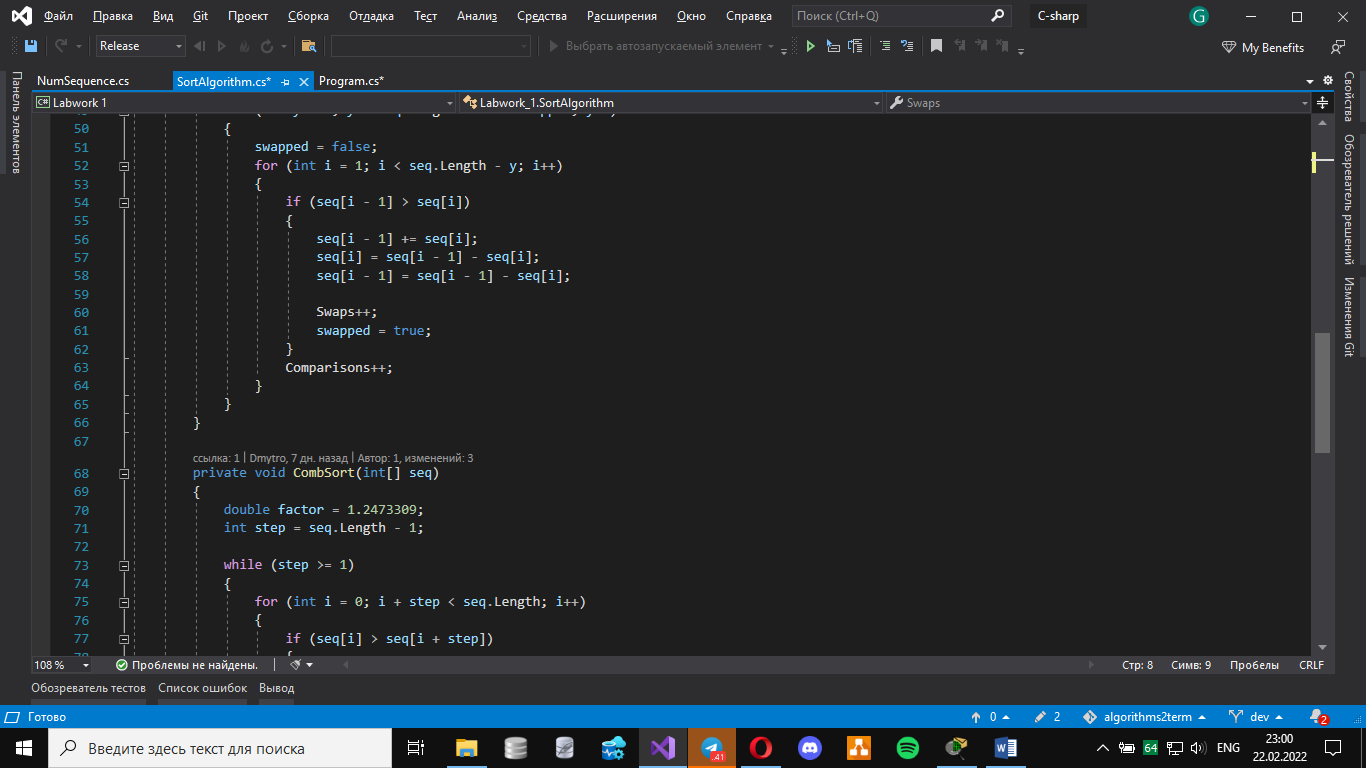
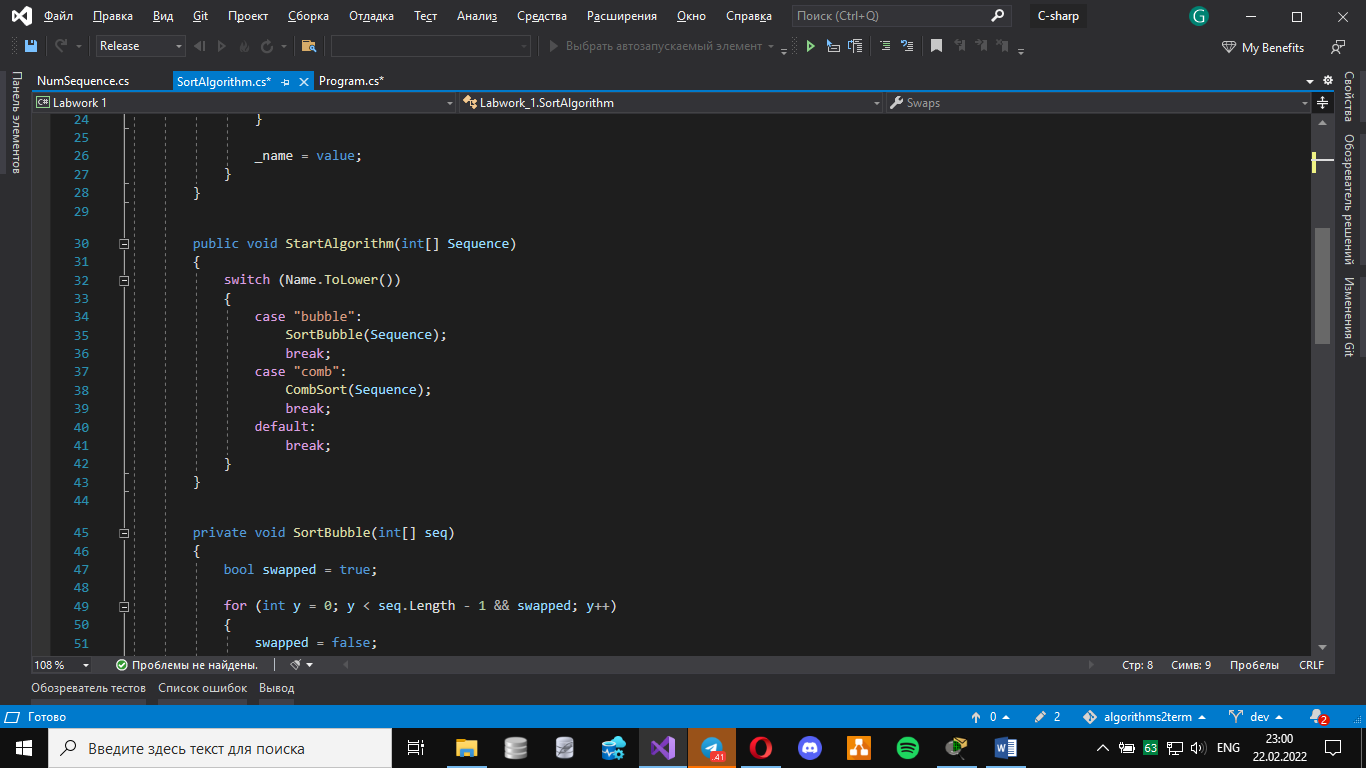
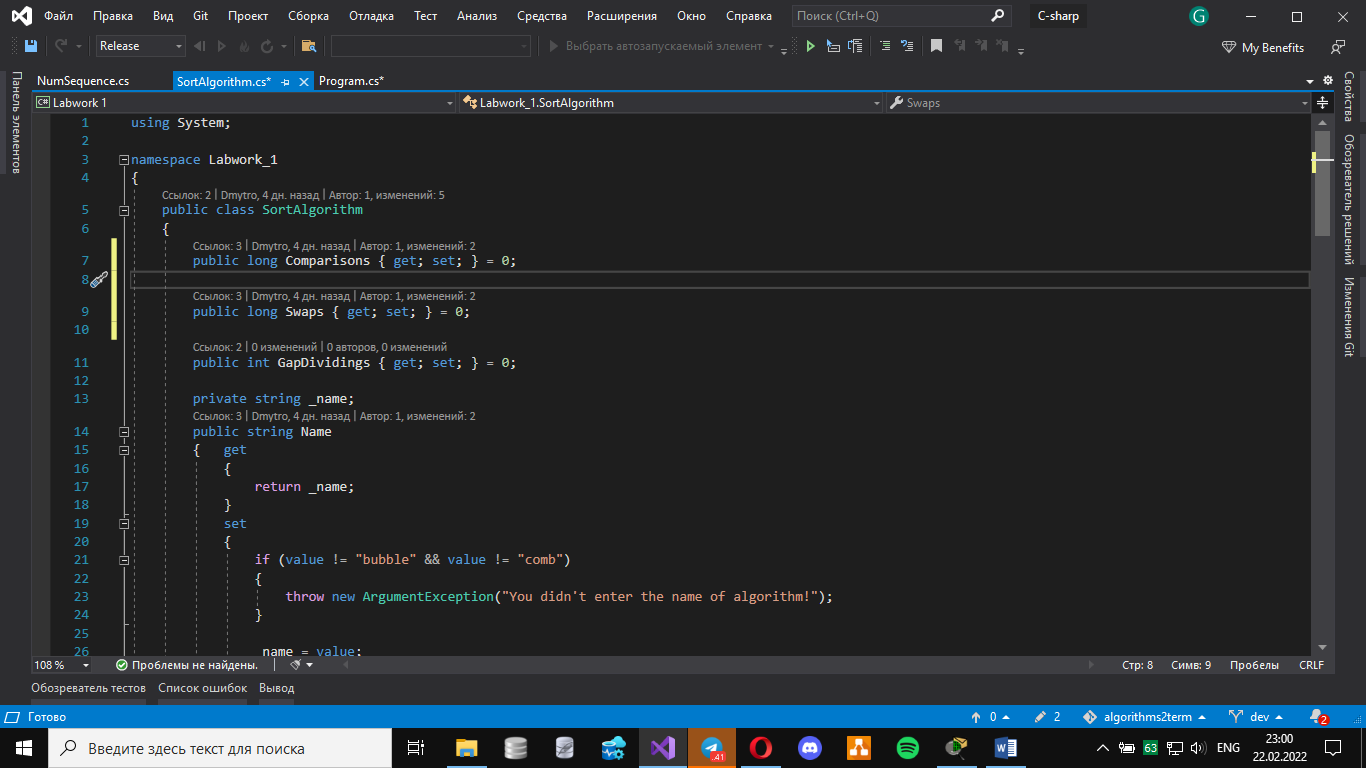
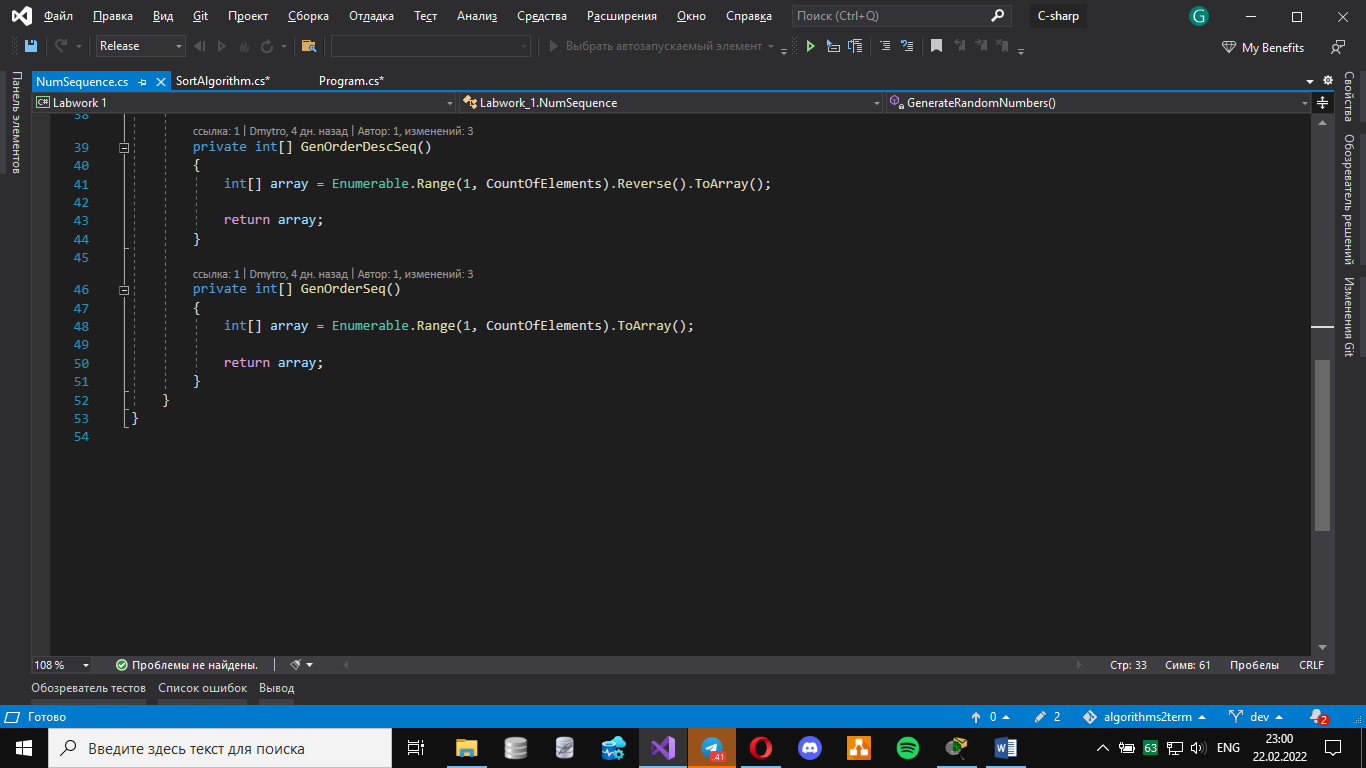
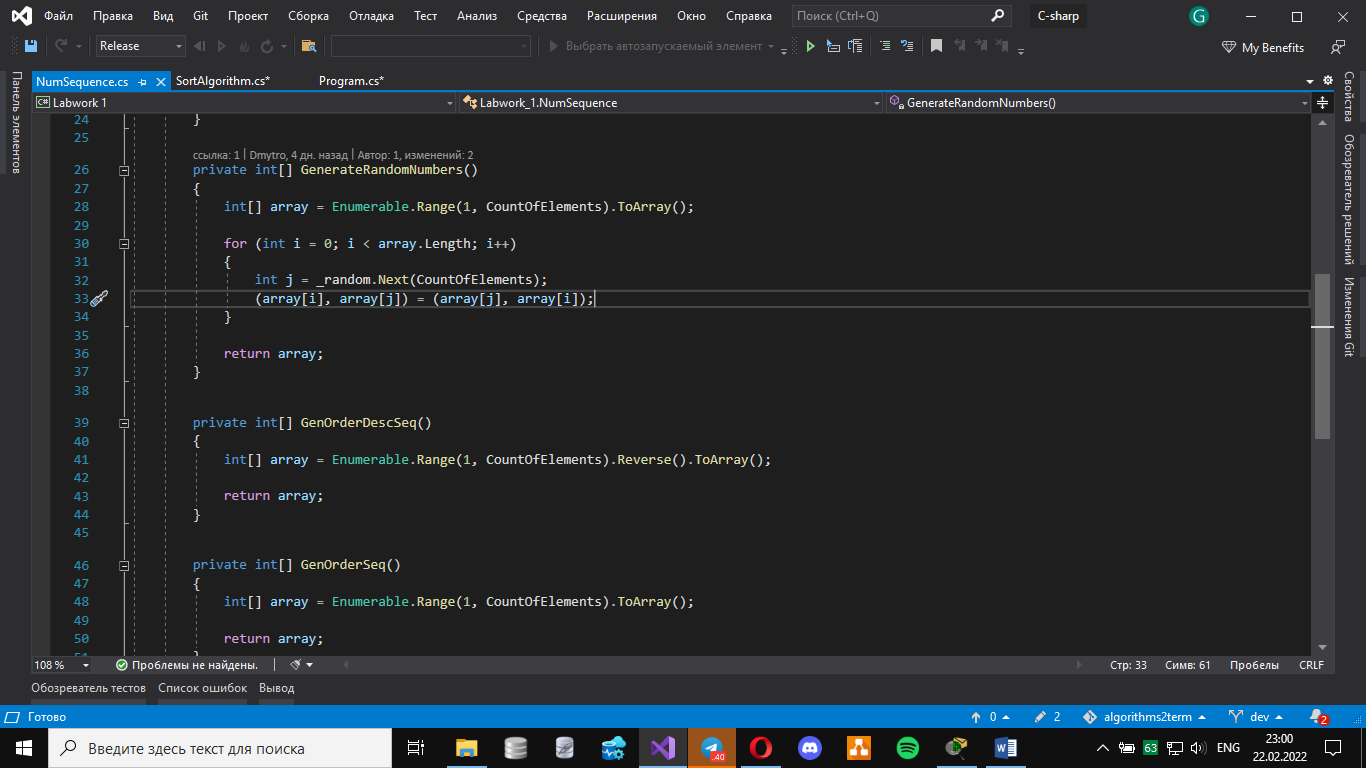
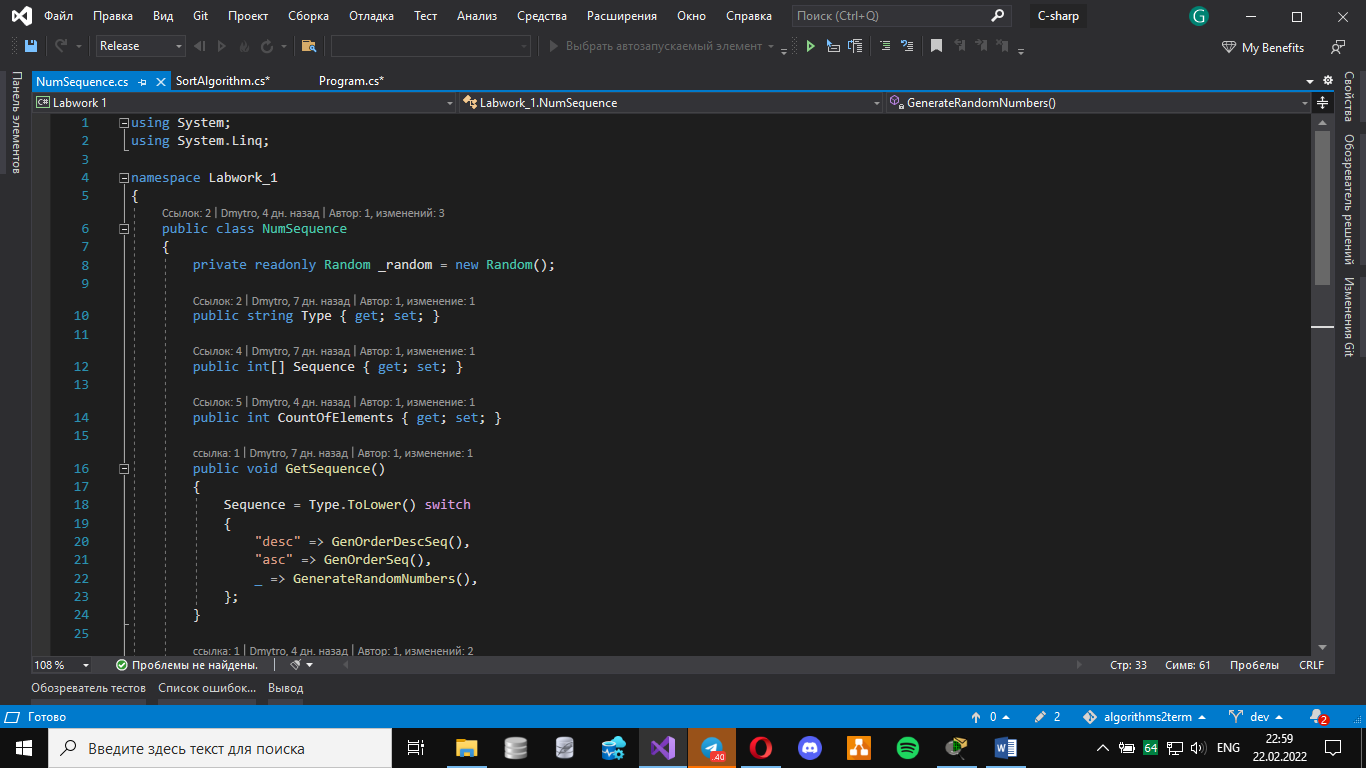
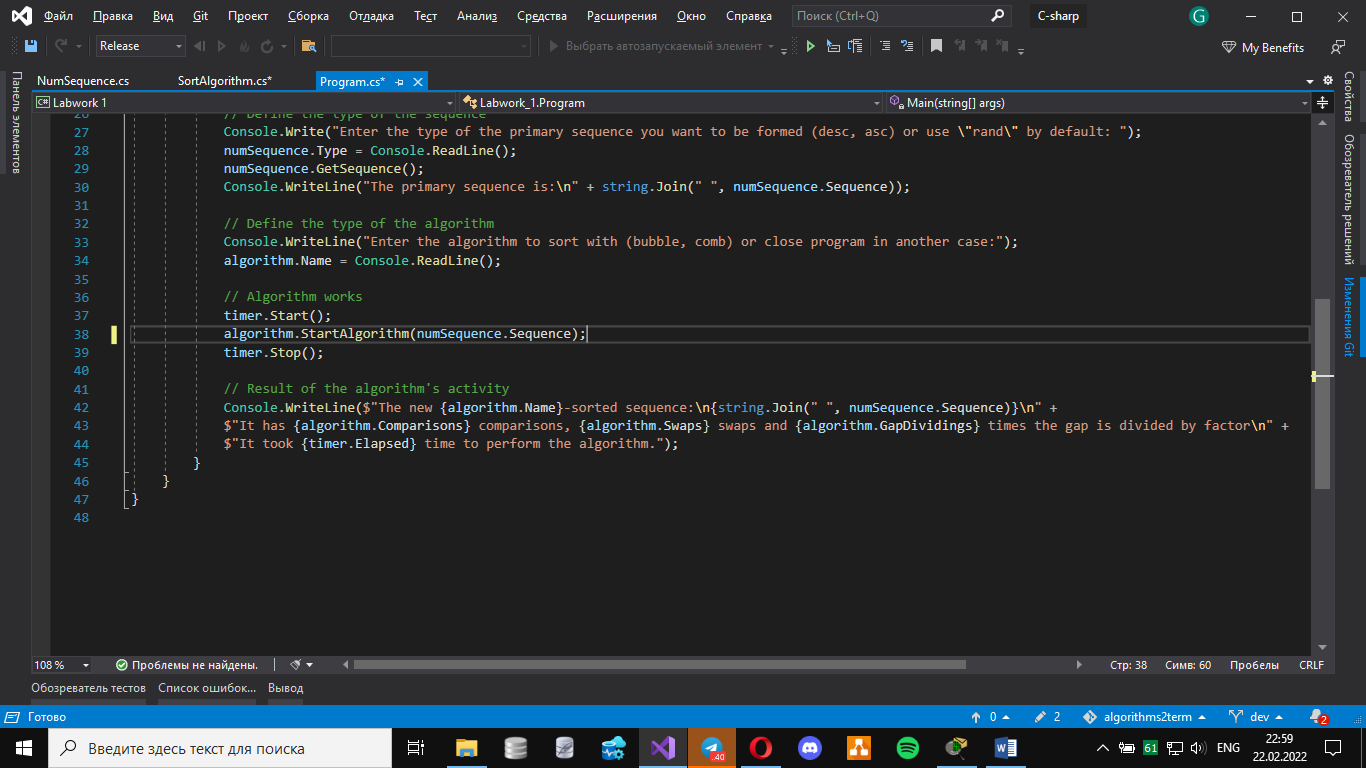
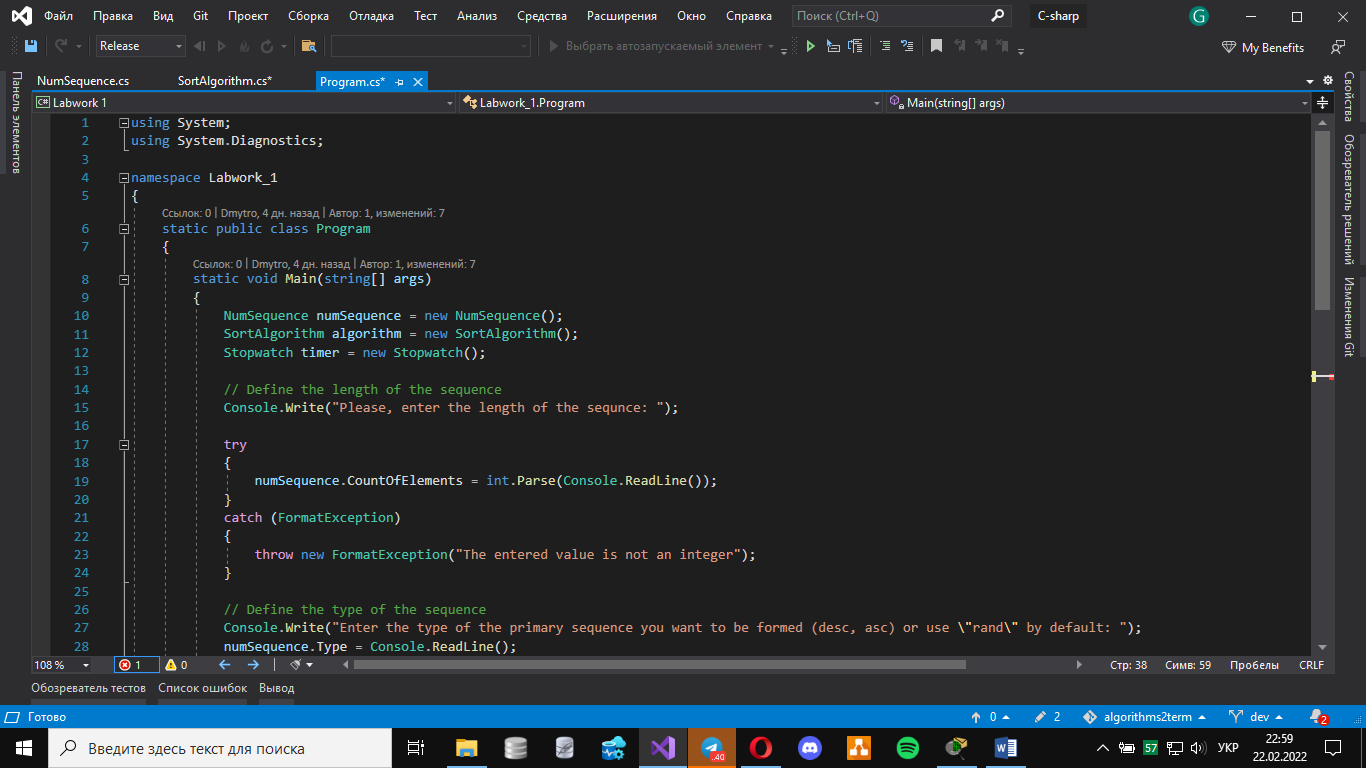
|  |  |  |
| --- | --- | --- |
| **Перестановки** | | |
| **Гірший**  *О(n²)* | **Середньостатистичний**  *О(n²)* | **Кращий**  *О(n)* |
| T(n) = 1 + n + (n - 1) + n(n+1)/2 + 5n(n-1)/2 =  3n2 => O(n2) | T(n) = 1+n+(n-1)+n(n+1)/2+5n(n-1)/2 = 2n+n2/2+n/2+5n2/2-5n/2  => O(n2) | T(n) = 1+n+(n-1)+n(n+1)/2 =>  O(n) |

**Метод сортування “гребінцем”**

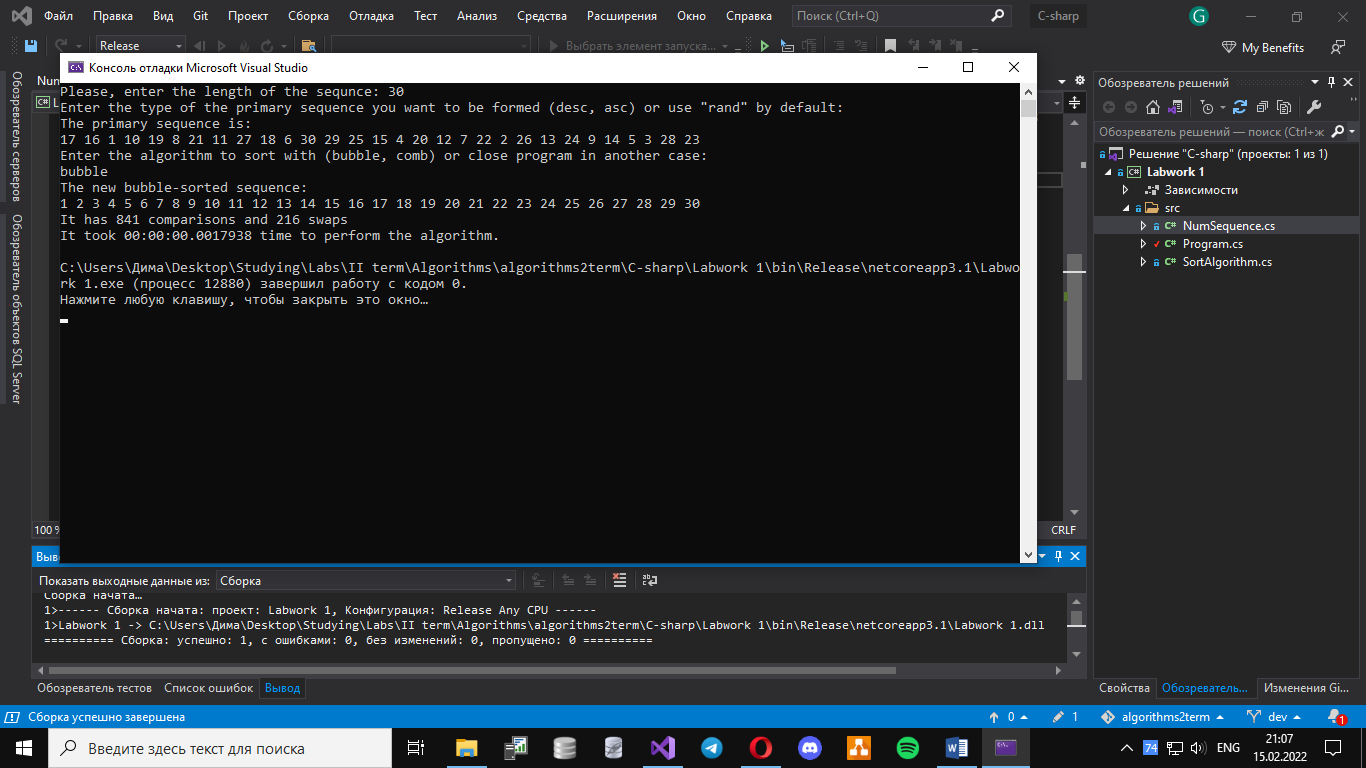
|  |  |  |
| --- | --- | --- |
| **Час** | | |
| **Гірший**  O(*n*2) | **Середньостатистичний**  Ω(N2/2p) | **Кращий**  O(*nlogn*) |
|  |  |  |

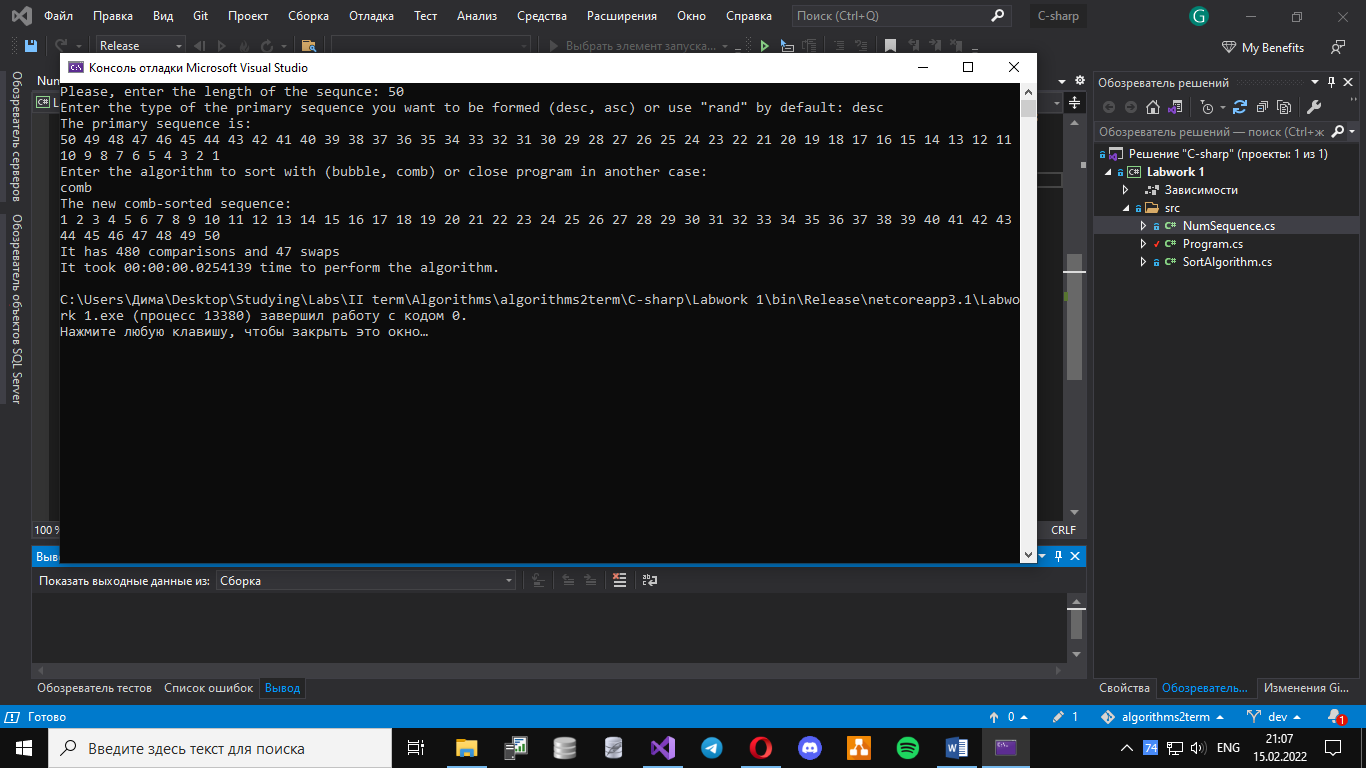
## Програмна реалізація алгоритму

### Вихідний код



### **Приклад роботи**

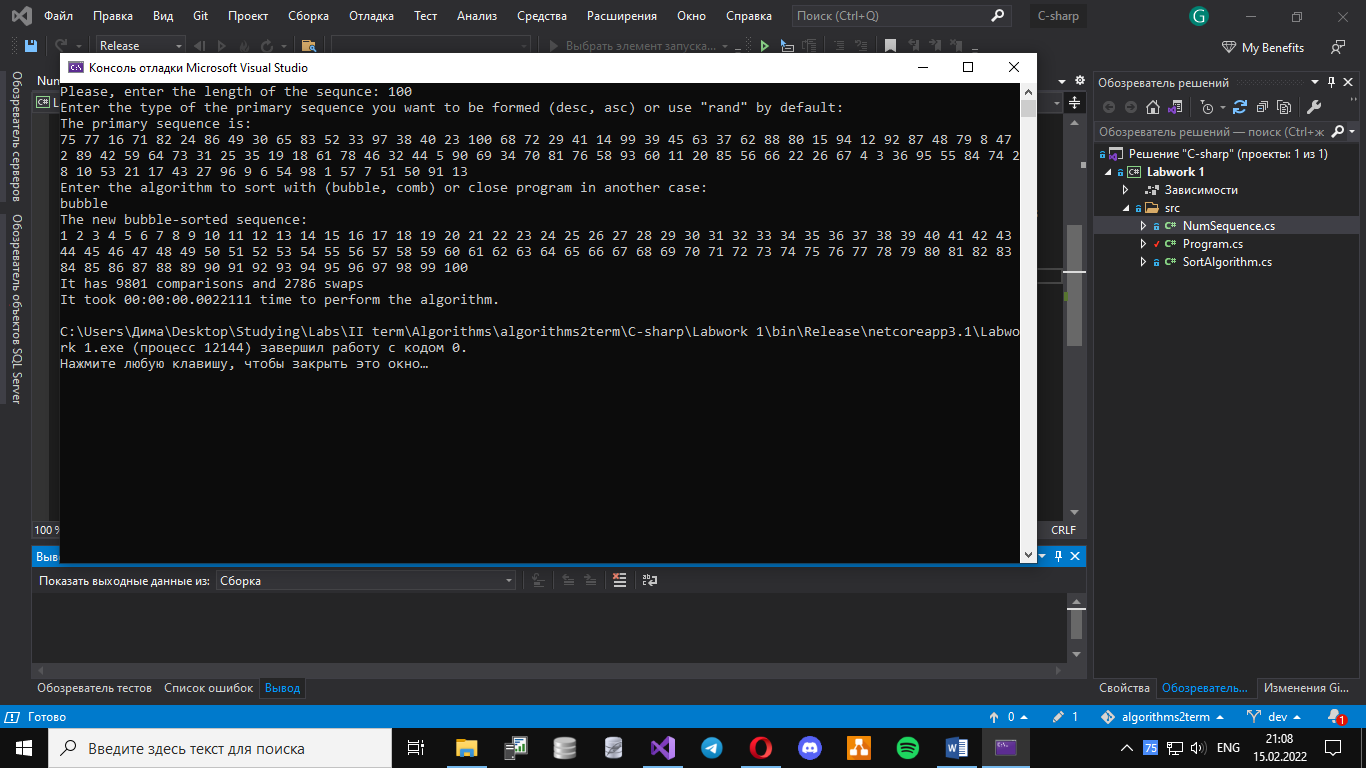




На рисунках 3.1 і 3.2 показані приклади роботи програми сортування масивів на 100 і 1000 елементів відповідно.

Рисунок 3.1 – Сортування масиву на 100 елементів

«Бульбашка»



«Гребінець»

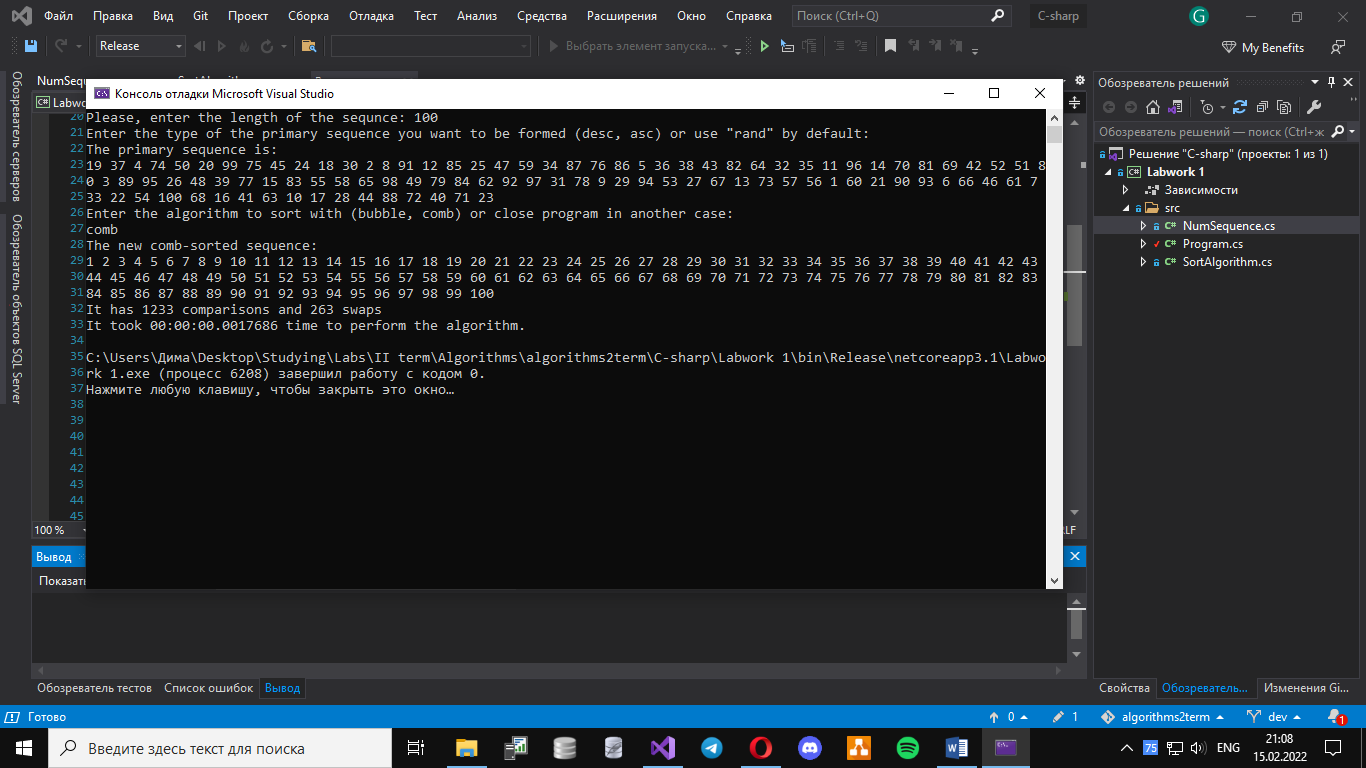
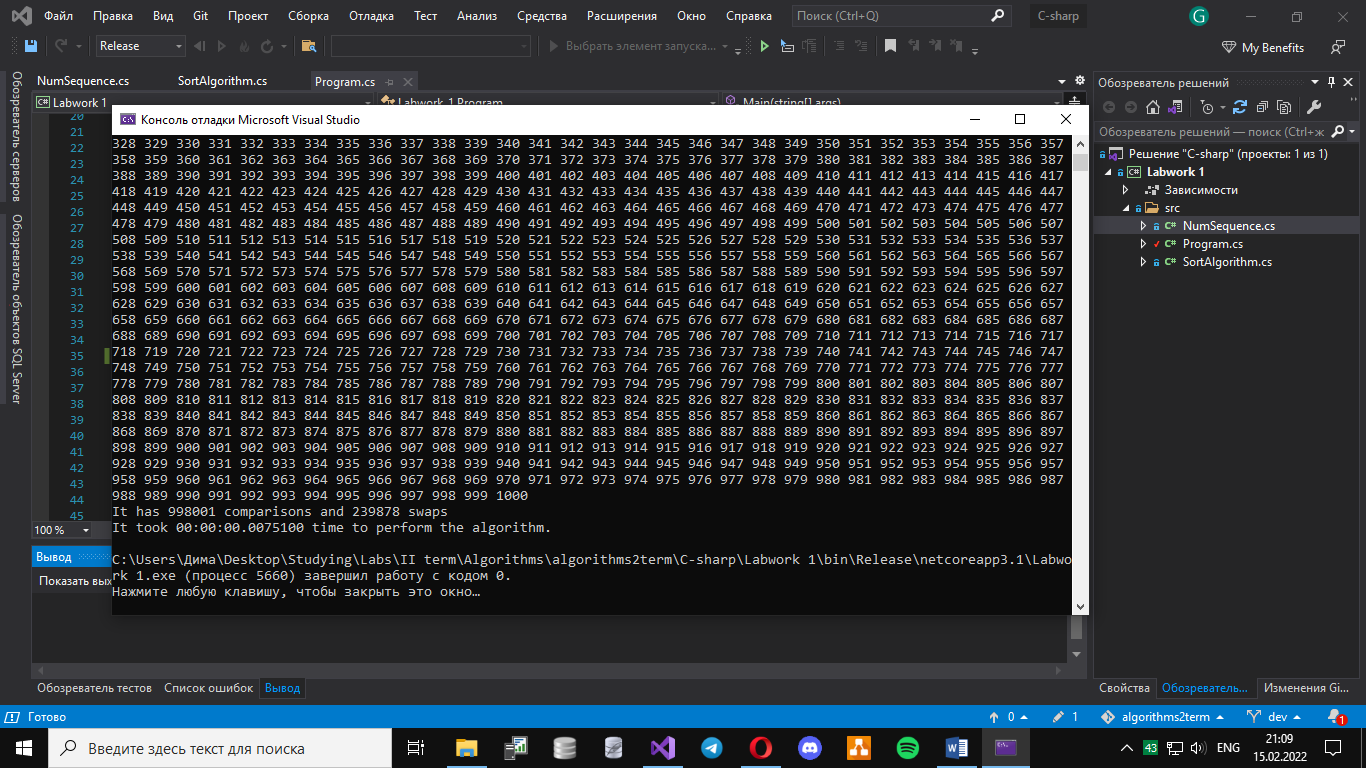
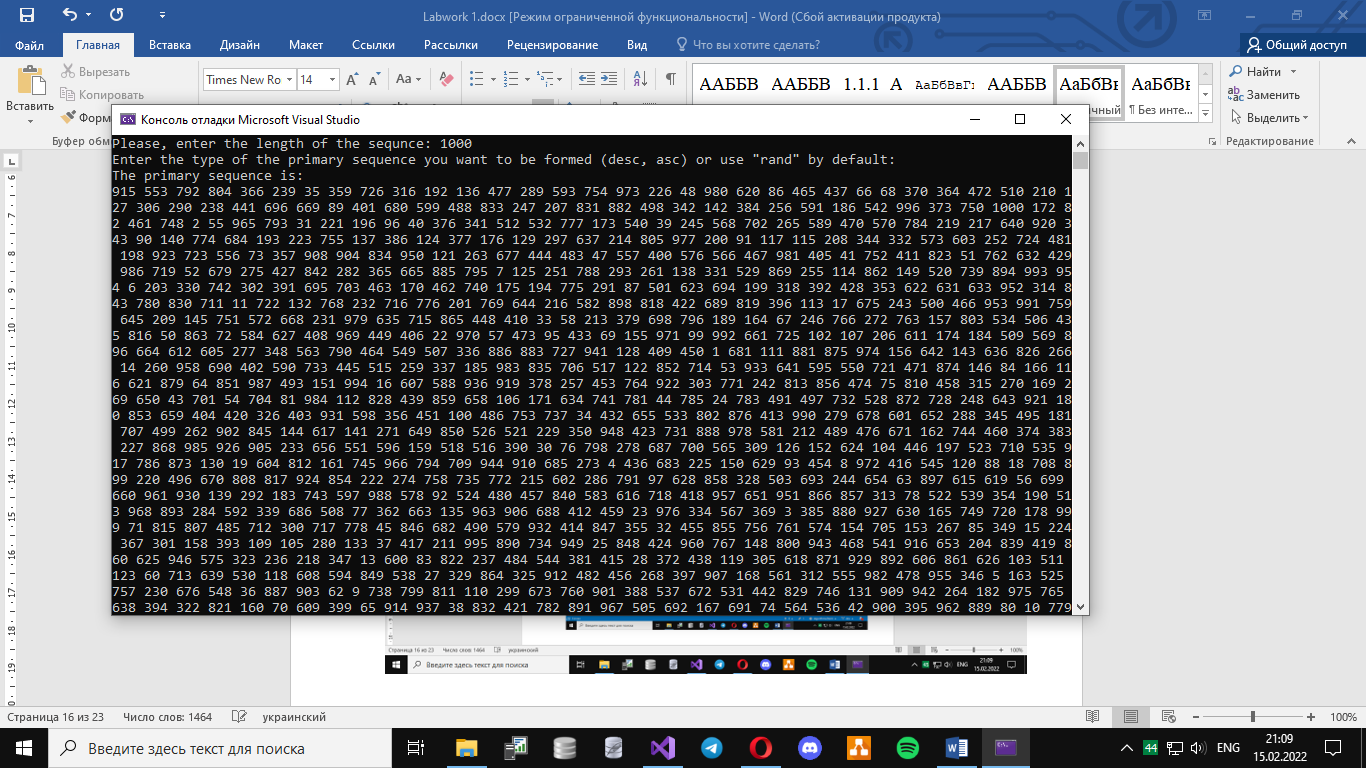
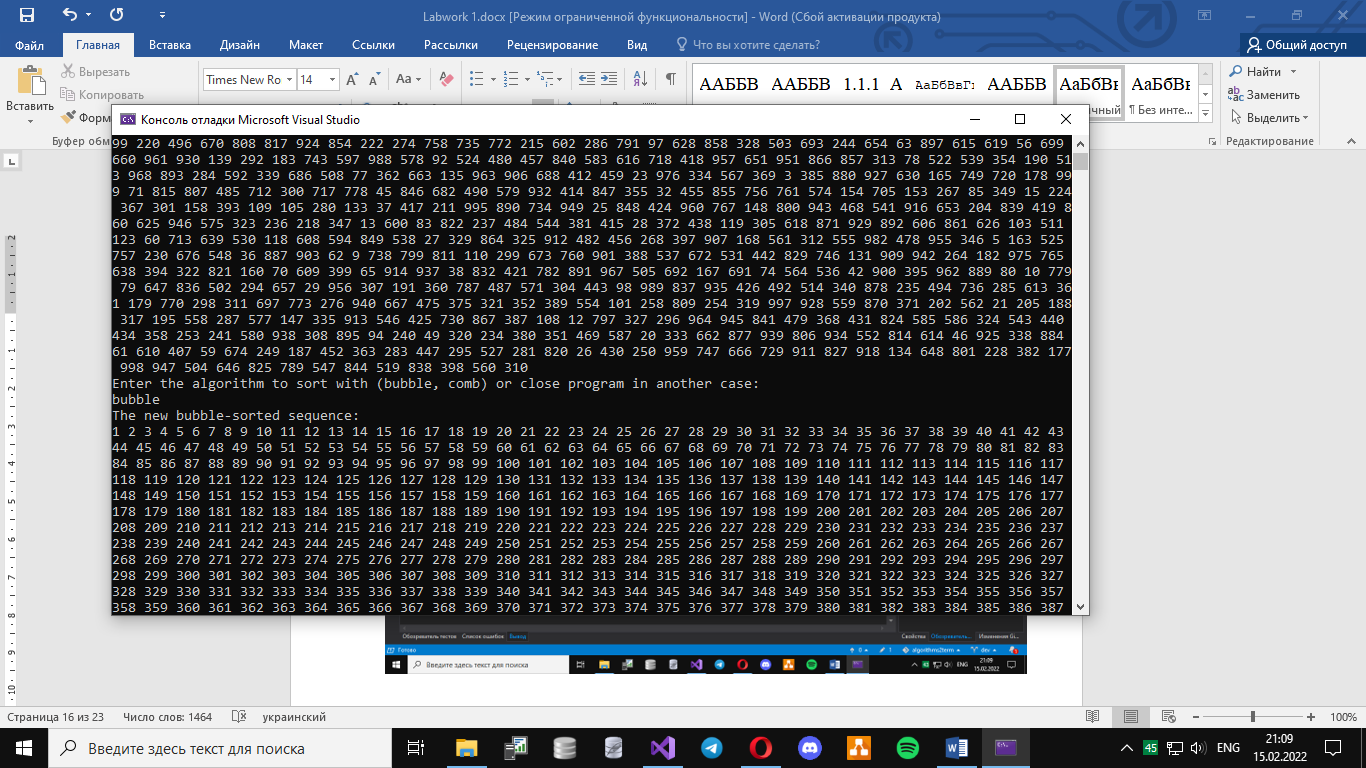
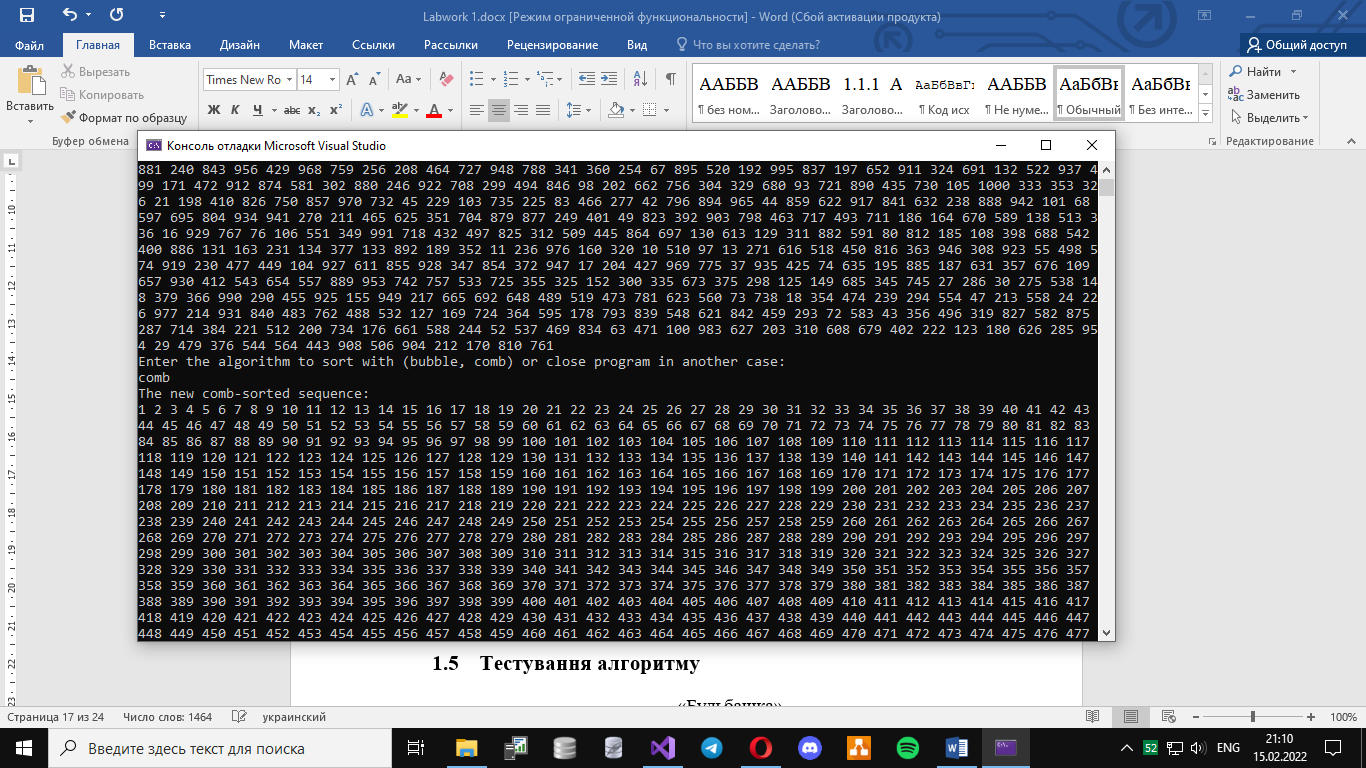
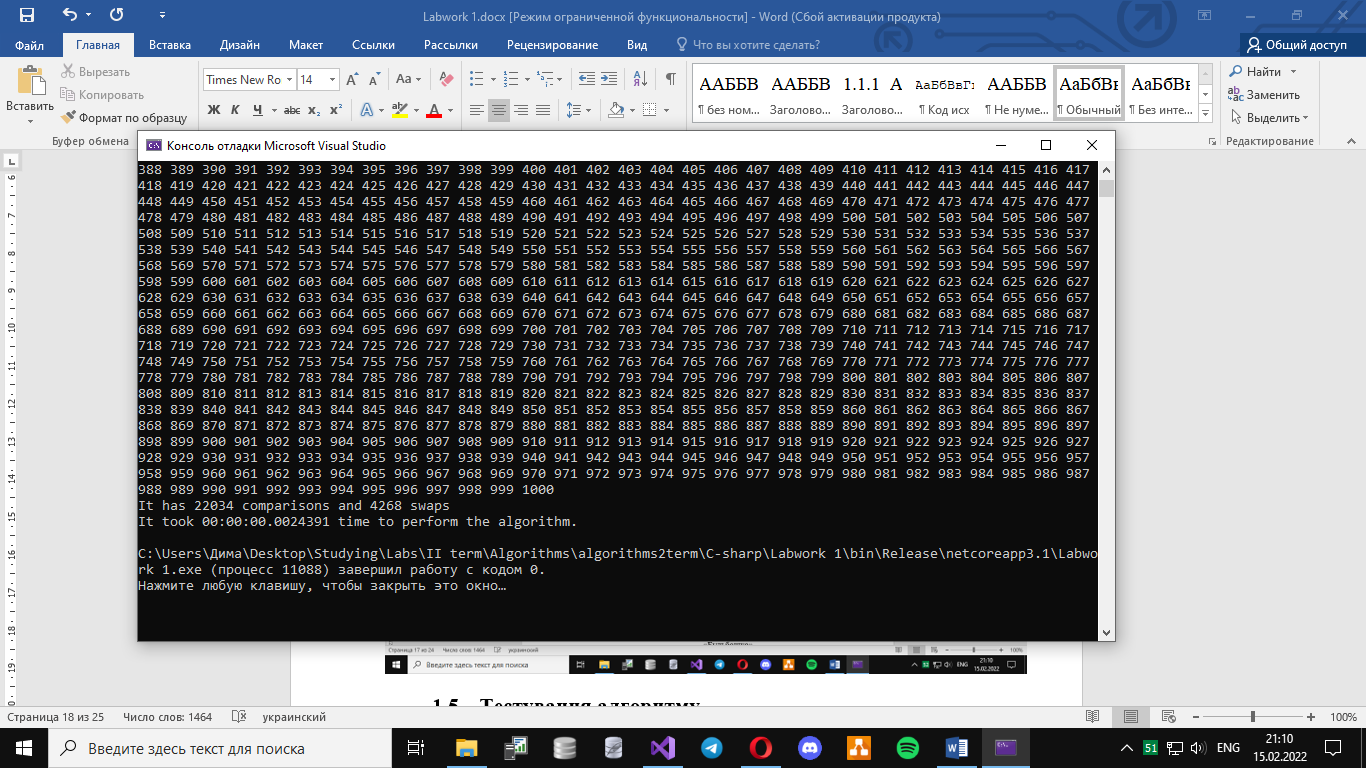
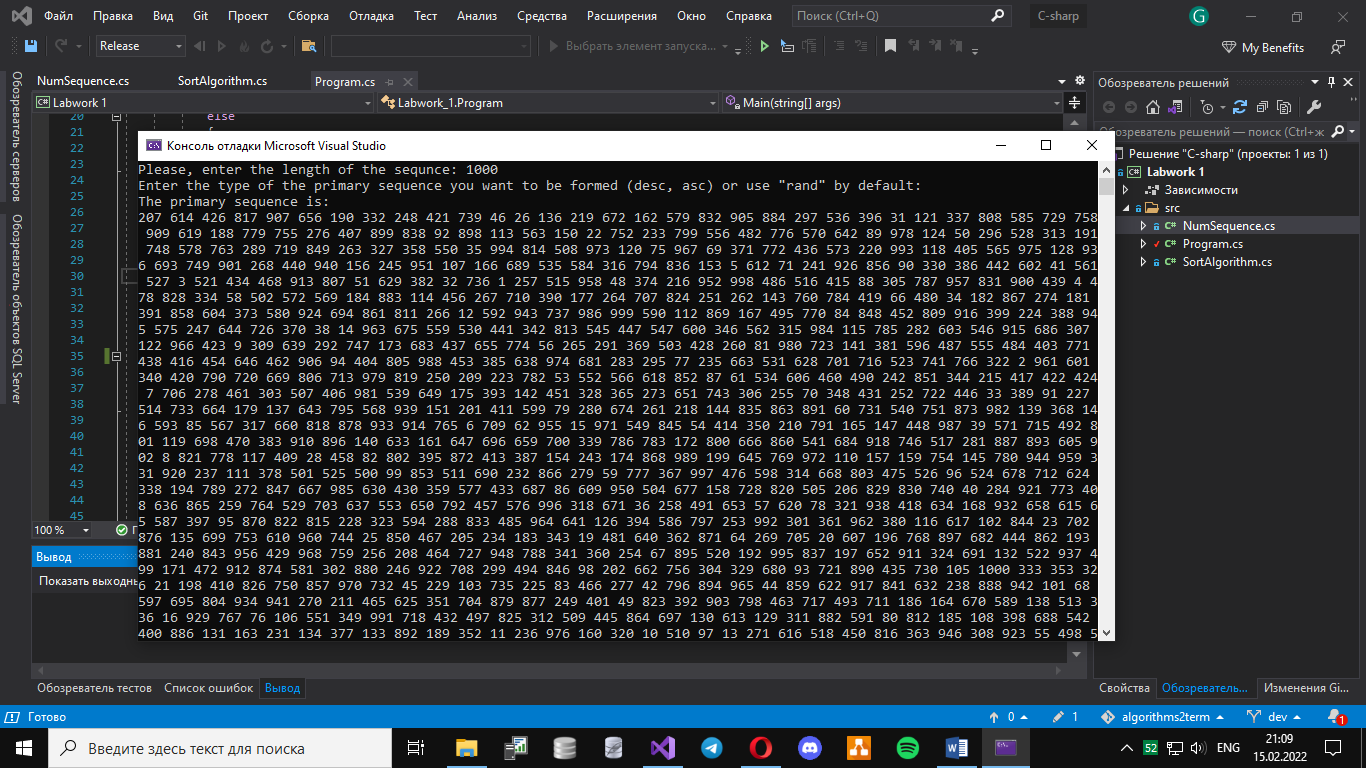


Рисунок 3.2 – Сортування масиву на 1000 елементів

«Бульбашка»



«Гребінець»

## Тестування алгоритму

### 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 | 9 | 0 |
| 100 | 99 | 0 |
| 1000 | 999 | 0 |
| 5000 | 4999 | 0 |
| 10000 | 9999 | 0 |
| 20000 | 19999 | 0 |
| 50000 | 49999 | 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 | 45 | 45 |
| 100 | 4950 | 4950 |
| 1000 | 499500 | 499500 |
| 5000 | 12497500 | 12497500 |
| 10000 | 49995000 | 49995000 |
| 20000 | 199990000 | 199990000 |
| 50000 | 1249975000 | 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 | 42 | 21 |
| 100 | 4905 | 2730 |
| 1000 | 499175 | 241950 |
| 5000 | 12492940 | 6021947 |
| 10000 | 49990722 | 23845484 |
| 20000 | 199981872 | 96323542 |
| 50000 | 1249963372 | 603441649 |

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

|  |  |  |
| --- | --- | --- |
| Розмірність масиву | Число порівнянь | Число перестановок |
| 10 | 39 | 4 |
| 100 | 1233 | 219 |
| 1000 | 22034 | 4206 |
| 5000 | 144865 | 27094 |
| 10000 | 329653 | 59774 |
| 20000 | 719246 | 130433 |
| 50000 | 1997961 | 367660 |

### 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 проводив розрахунки, вимірюючи час виконання алгоритму, закріпив свої знання зі способів використання алгоритмів та їх доречності в залежності від ситуації. Мною було проведене тестування алгоритму сортування «бульбашкою» та «гребінцем» шляхом підстановки різних значень, які представляють кількість елементів послідовності та їх безпосередні порівняння на графіках.

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%.