Міністерство освіти і науки України

Національний технічний університет України «Київський політехнічний

інститут імені Ігоря Сікорського"

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 1 з дисципліни

««Алгоритми та структури даних 2. Структури даних»

«Проектування і аналіз алгоритмів внутрішнього сортування»

Варіант 14

Виконав студент: ІП-15 Кондрацька Соня Леонідівна

Перевірив: Соколовський Владислав Володимирович

Київ-2022

# 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 | Алгоритм стійкий |
| «Природність» поведінки (Adaptability) | Алгоритм є природним |
| Базуються на порівняннях | Алгоритм базується на порівняннях |
| Необхідність в додатковій пам'яті (об'єм) | Не потребує додаткової пам’яті |
| Необхідність в знаннях про структури даних | Для використання алгоритму потрібно мати базові знання про структури даних |

## Псевдокод алгоритму

**Підпрограма** sort(arr, size)

**Початок**

flag: = 0

**повторити для і від 0 до size**

**повторити для j від 0 до size-i-1**

**якщо** arr[j] > arr[j + 1] **то**

flag:=+1

temp: = arr[j]

arr[j]: = arr[j + 1]

arr[j + 1]: = temp

**все якщо**

**все повторити**

**якщо** flag== 0 **то**

break

**все якщо**

**все повторити**

**Кінець**

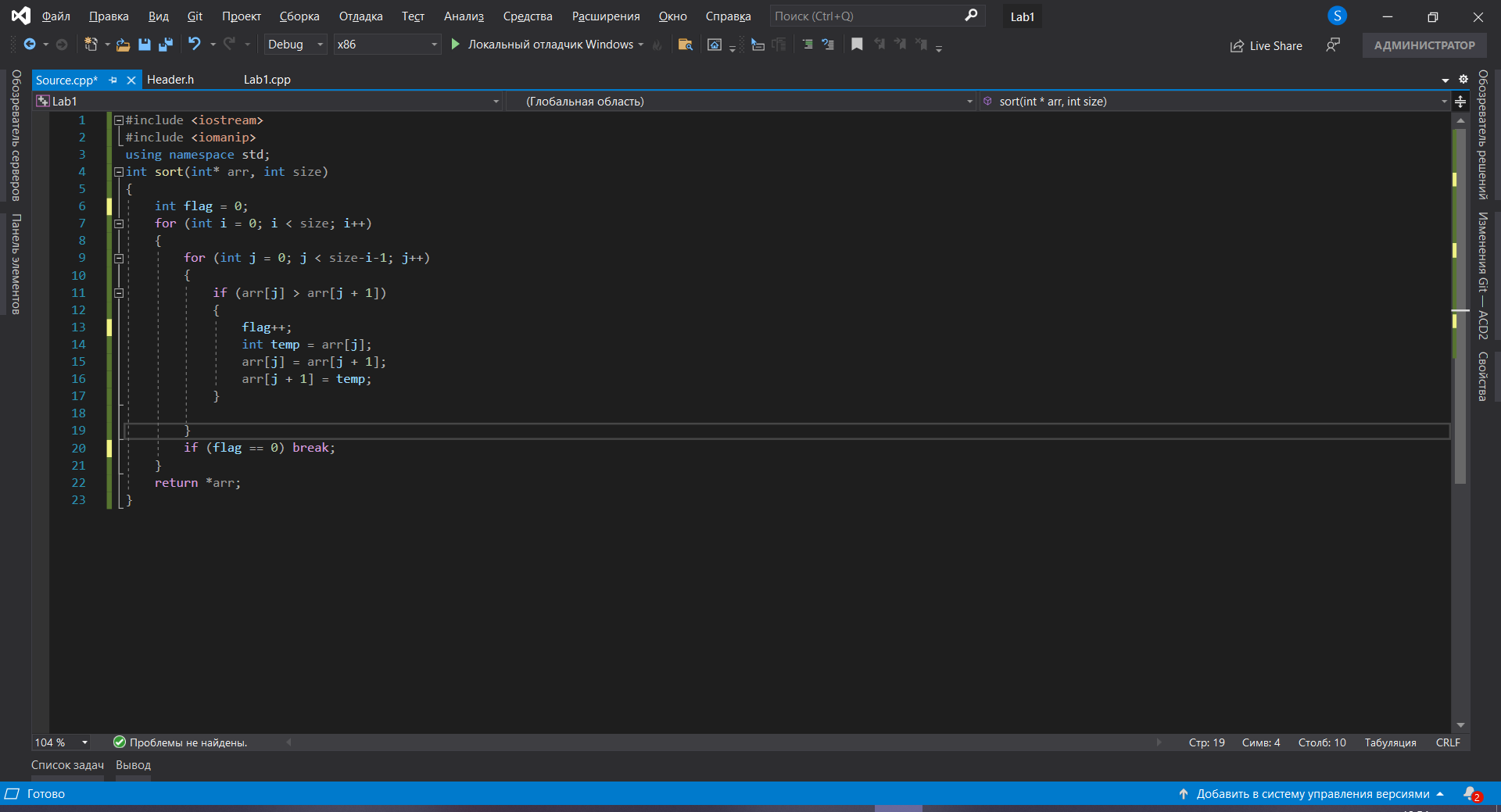
## Аналіз часової складності

Найгірший випадок: О(n²)

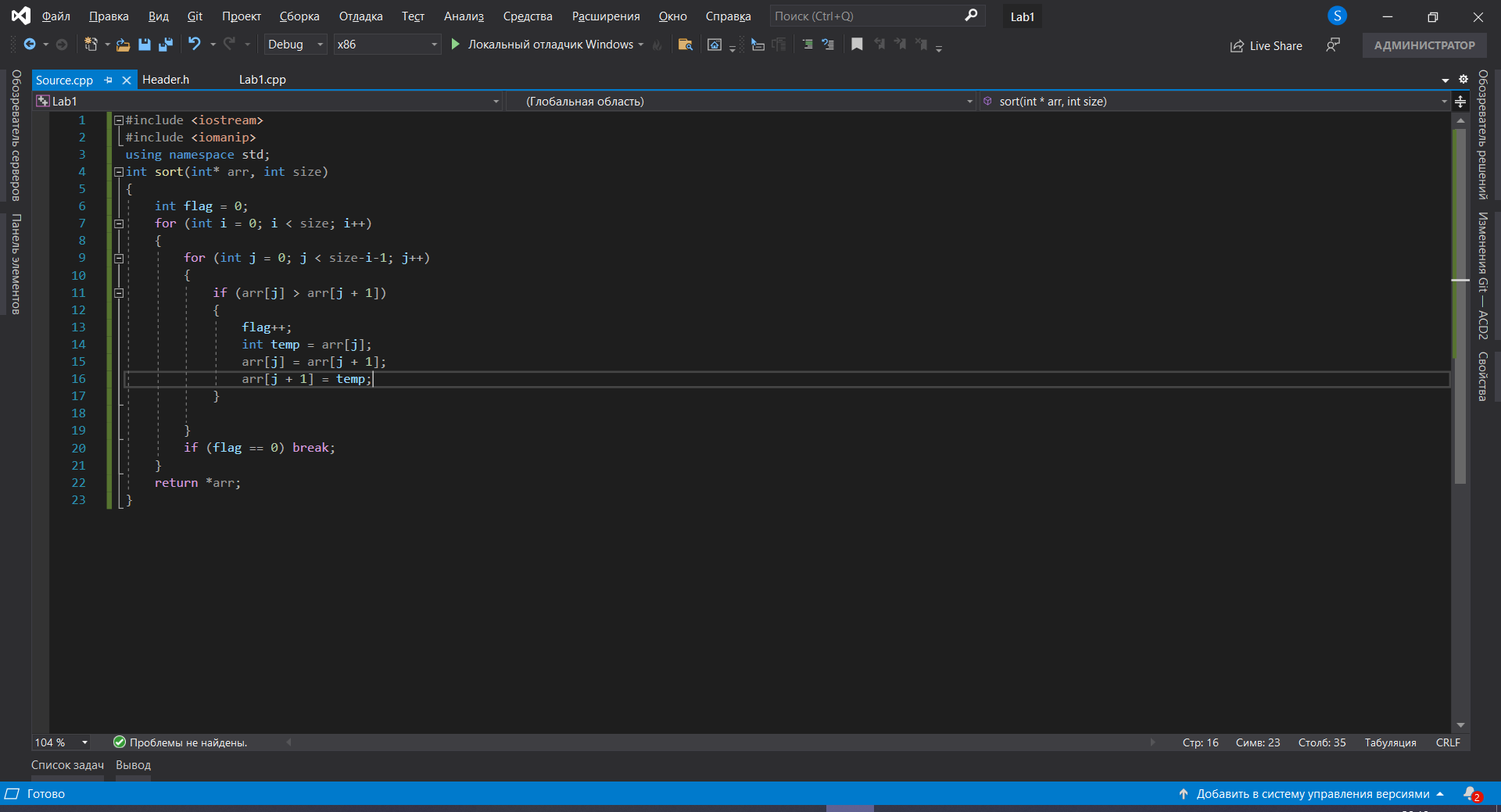
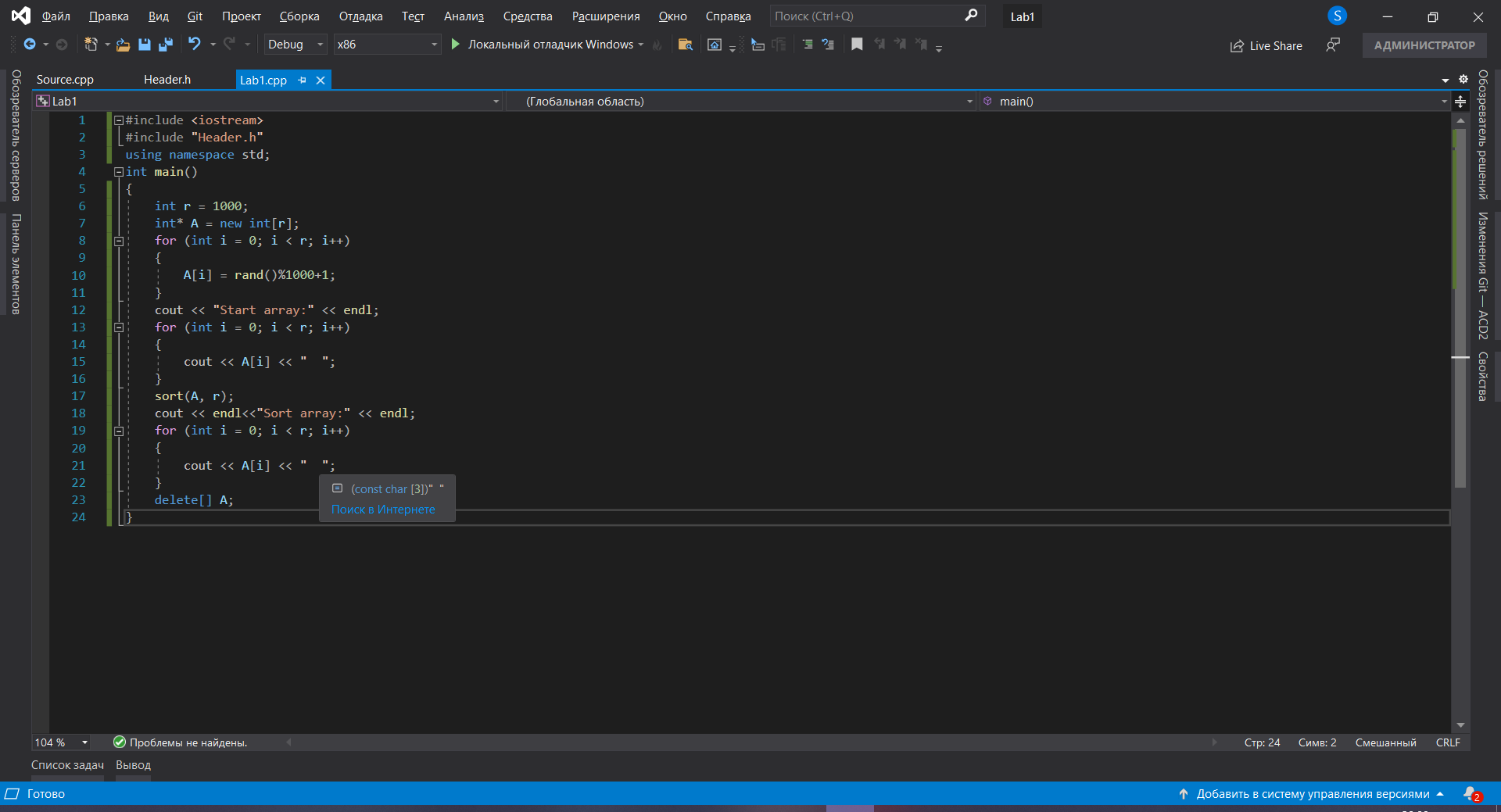
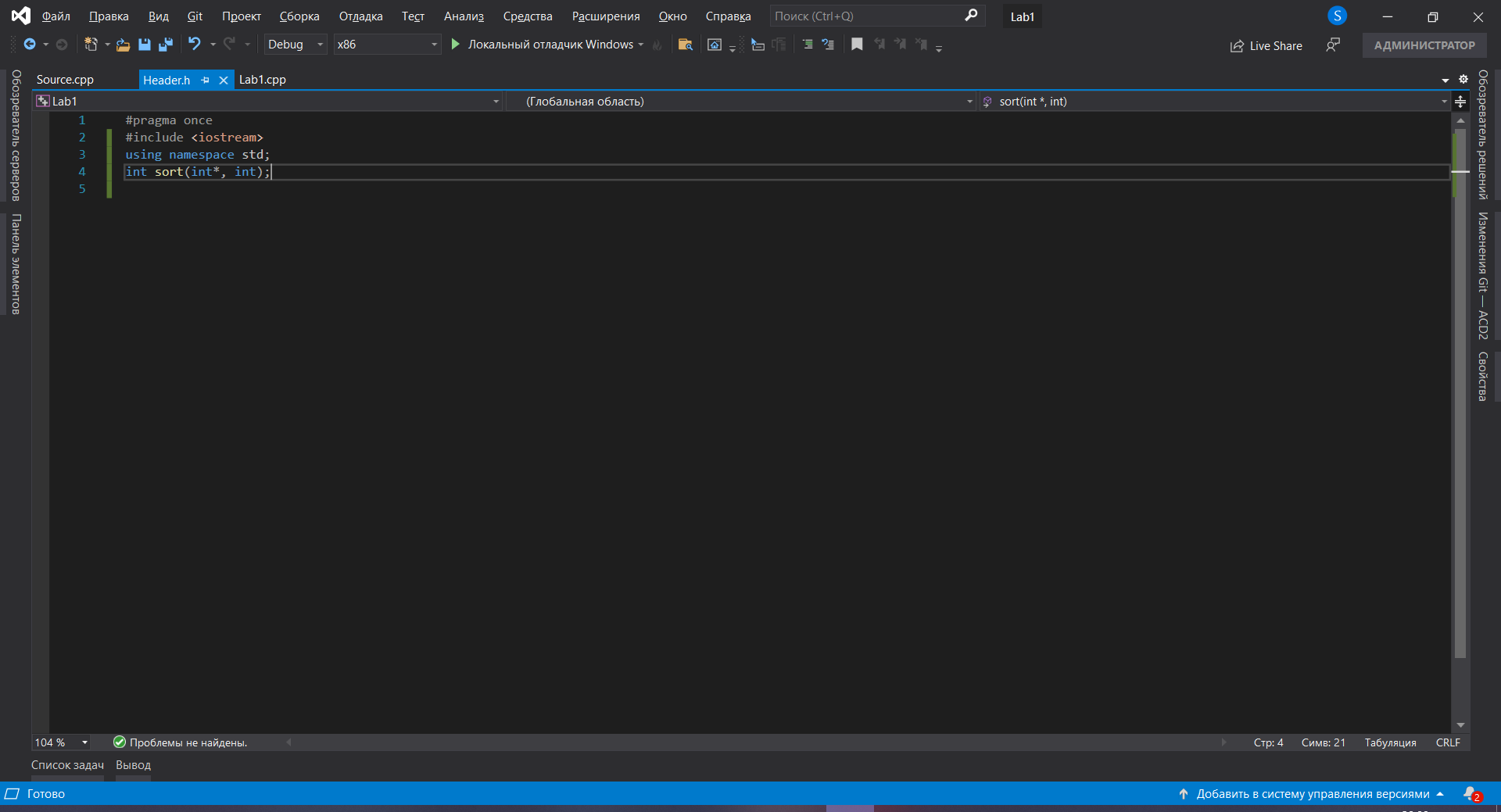
Середній випадок: О(n²)

Найкращий випадок: О(n)

## Програмна реалізація алгоритму



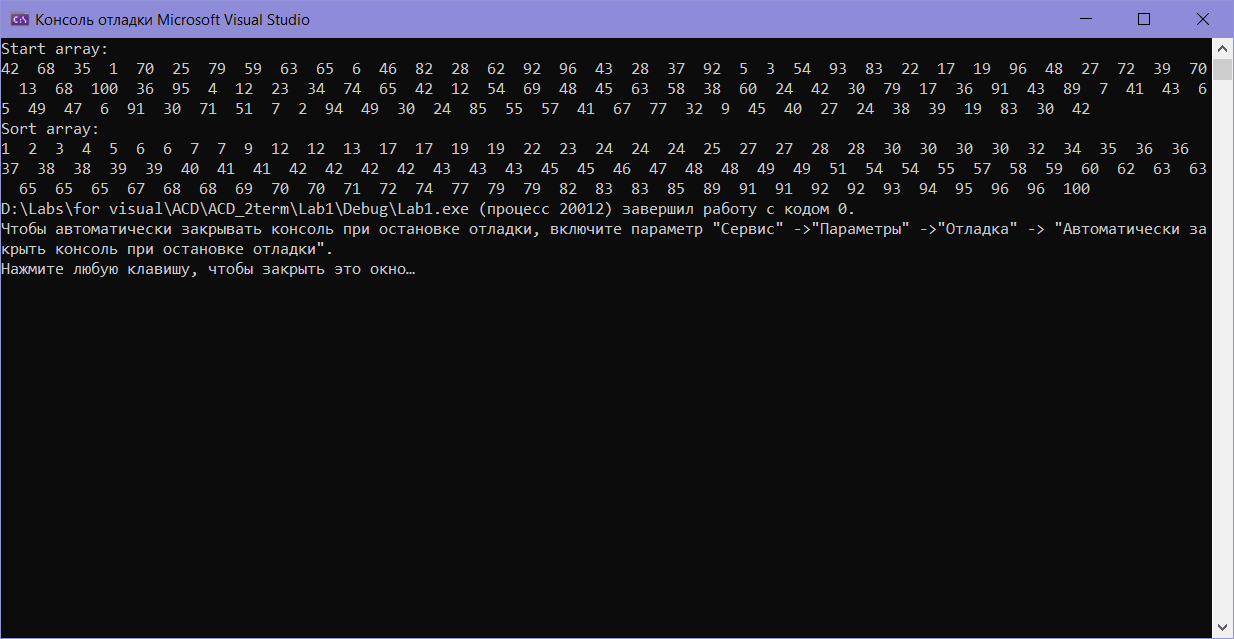
### Вихідний код

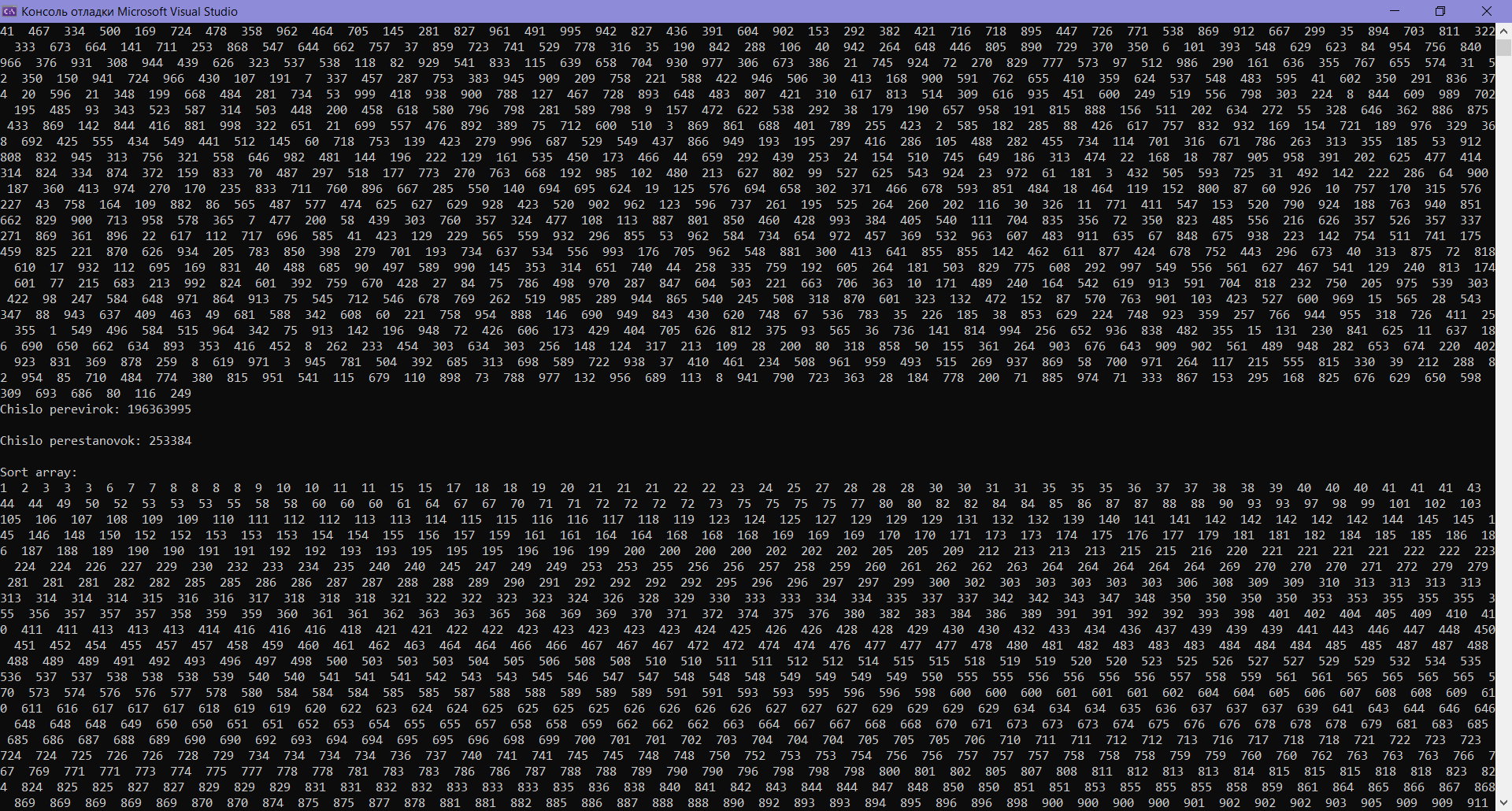


### Приклад роботи

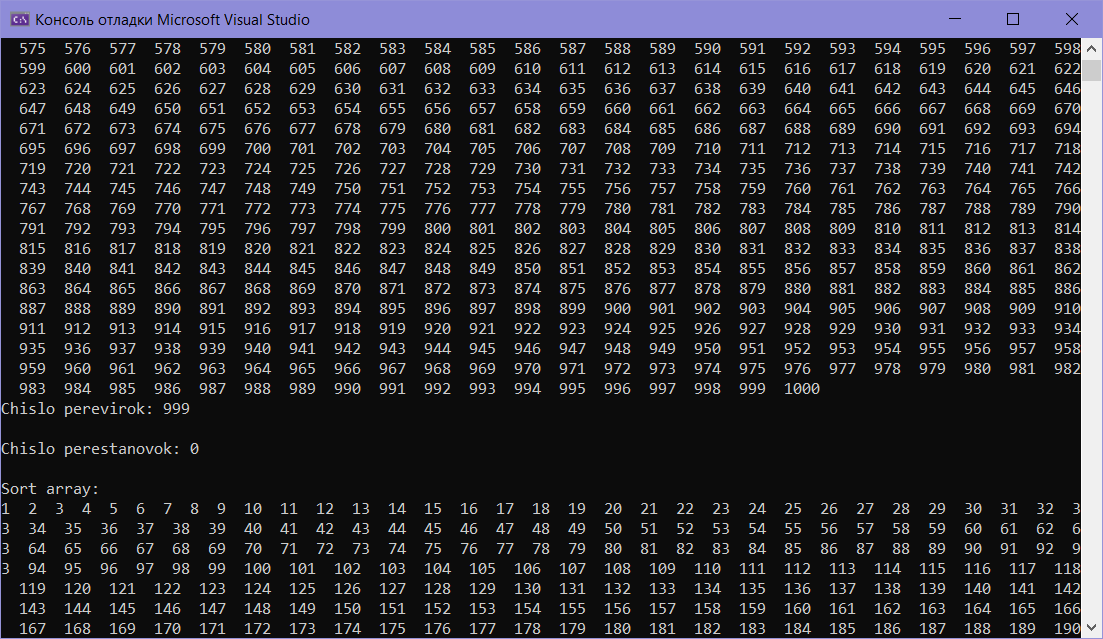
На рисунках 3.1 і 3.2 показані приклади роботи програми сортування масивів на 100 і 1000 елементів відповідно.

Рисунок 3.1 – Сортування масиву на 100 елементів

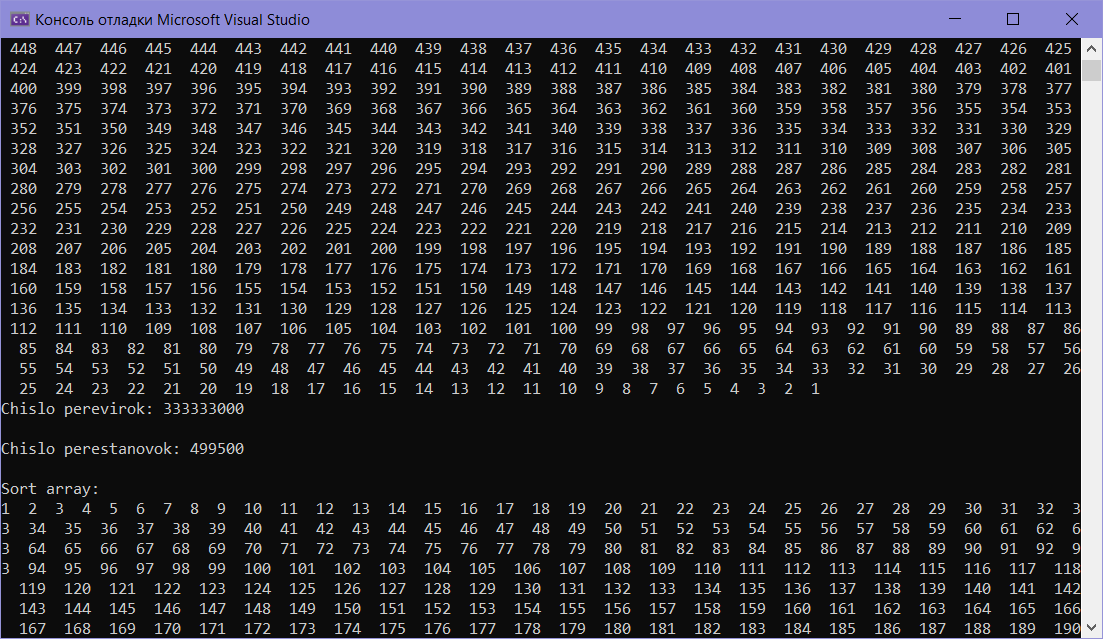
Рисунок 3.2 – Сортування масиву на 1000 елементів

Число перестановок та перевірок для:

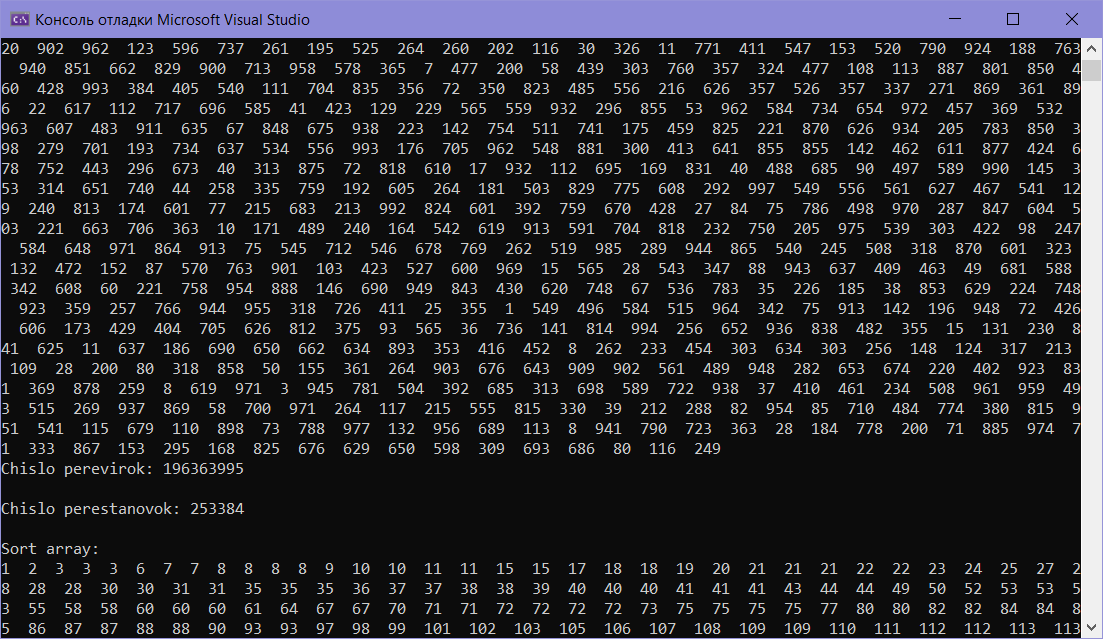
а)Найкращого випадку



б)Найгіршого випадку



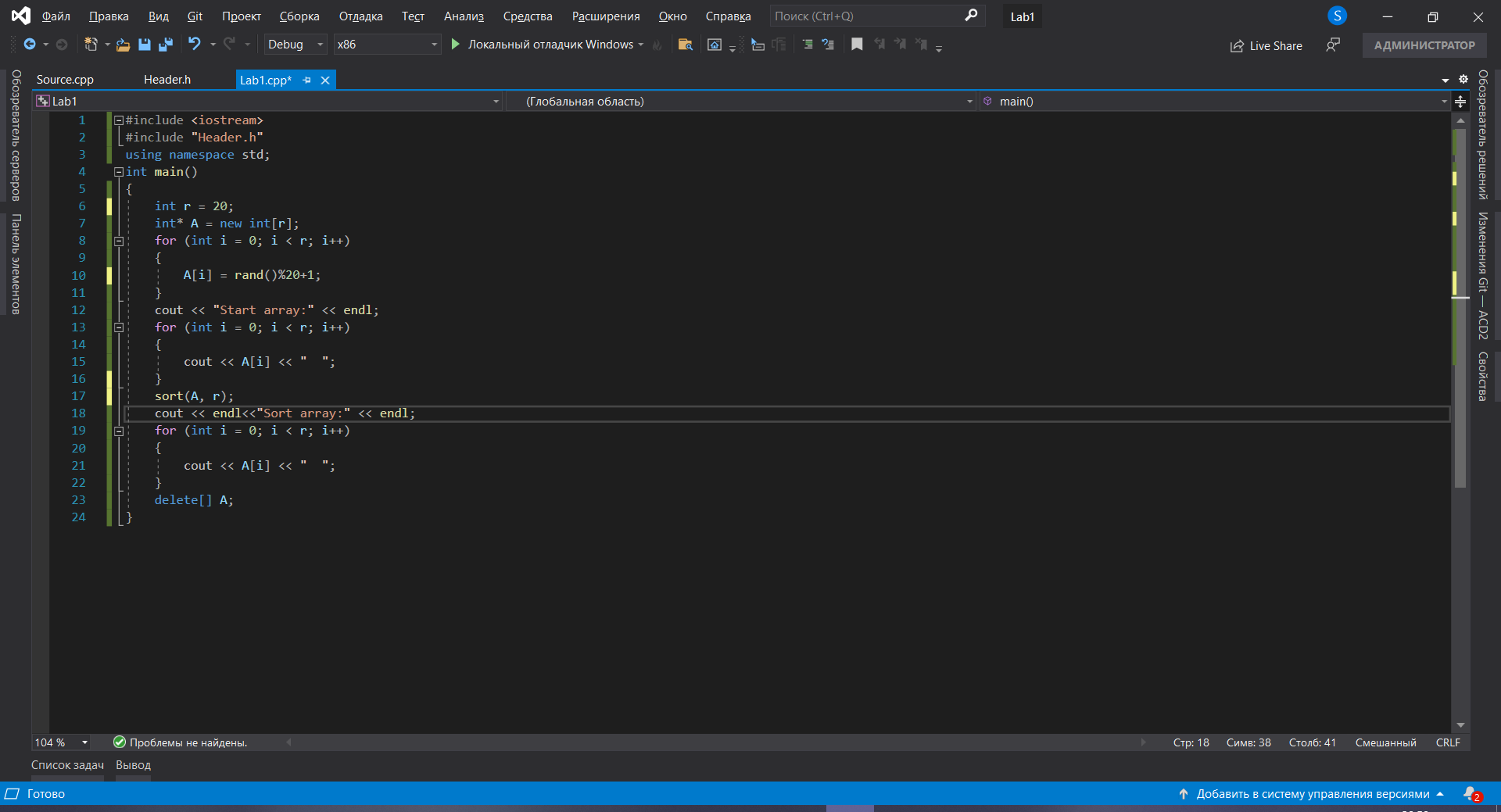
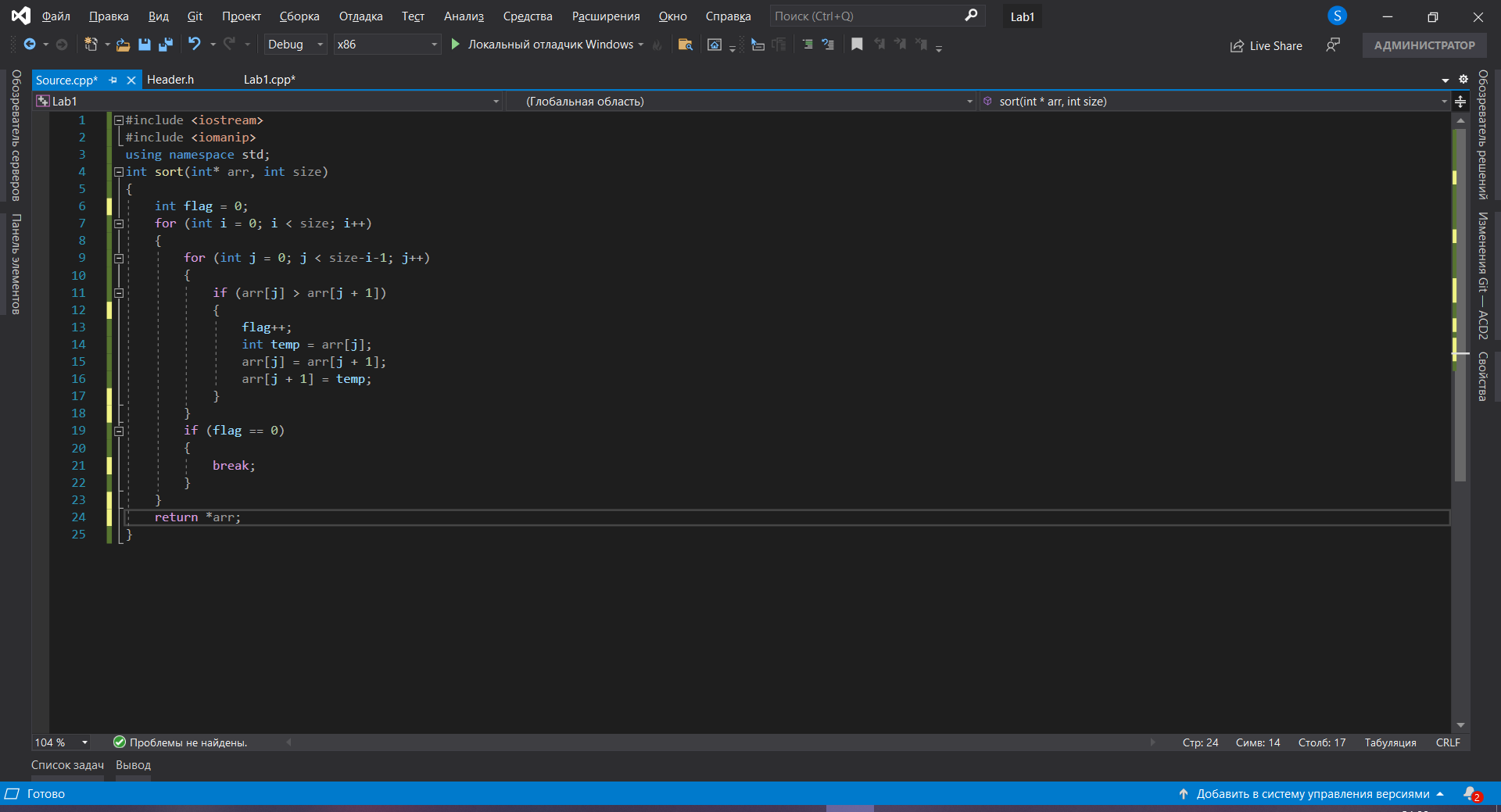
в)випадкового

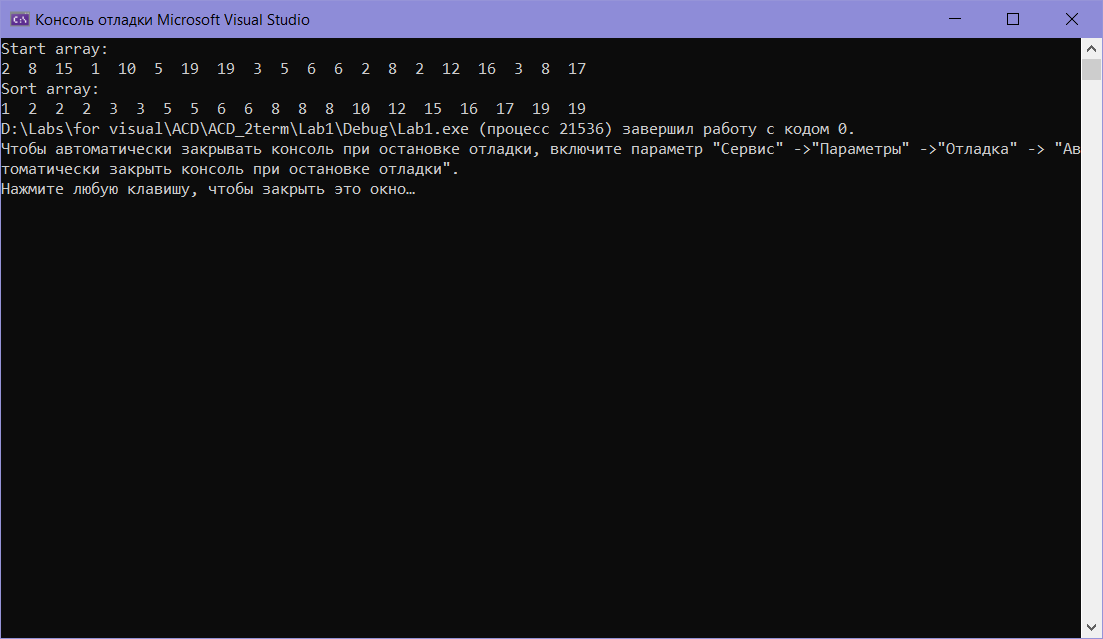


## Тестування алгоритму

Додатково протестуємо алгоритм сортування бульбашкою на випадково згенерованому масиві з розмірністю 20 елементів (елементи можуть повторюватись).

Код:

Результат виконання програми:

Як бачимо, програма коректно відсортувала випадково згенерований масив. Отже, наш алгоритм сортування бульбашкою працює правильно.

### 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 бульбашки 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 |

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 | 4 950 | 950 |
| 1000 | 499 500 | 499 500 |
| 5000 | 12 497 500 | 12 497 500 |
| 10000 | 49 995 000 | 49 995 000 |
| 20000 | 199 990 000 | 199 990 000 |
| 50000 | 1 249 975 000 | 1 249 975 000 |

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 бульбашки для випадкової послідовності елементів у масиві.

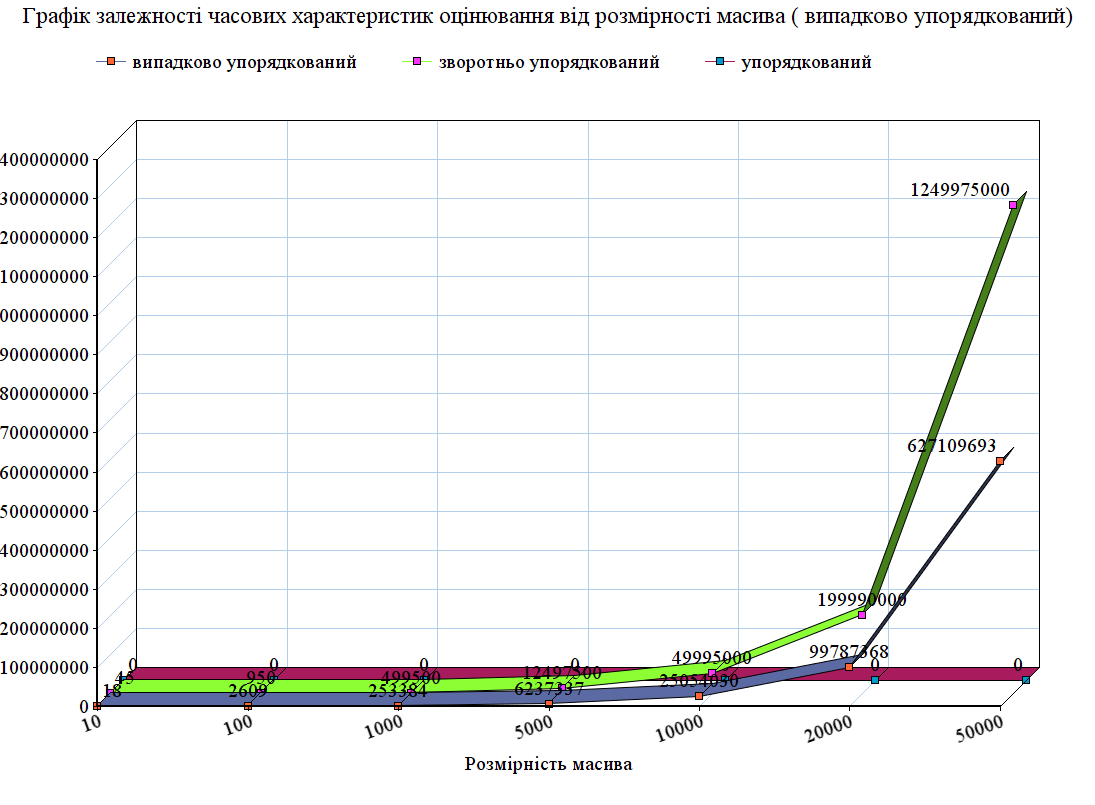
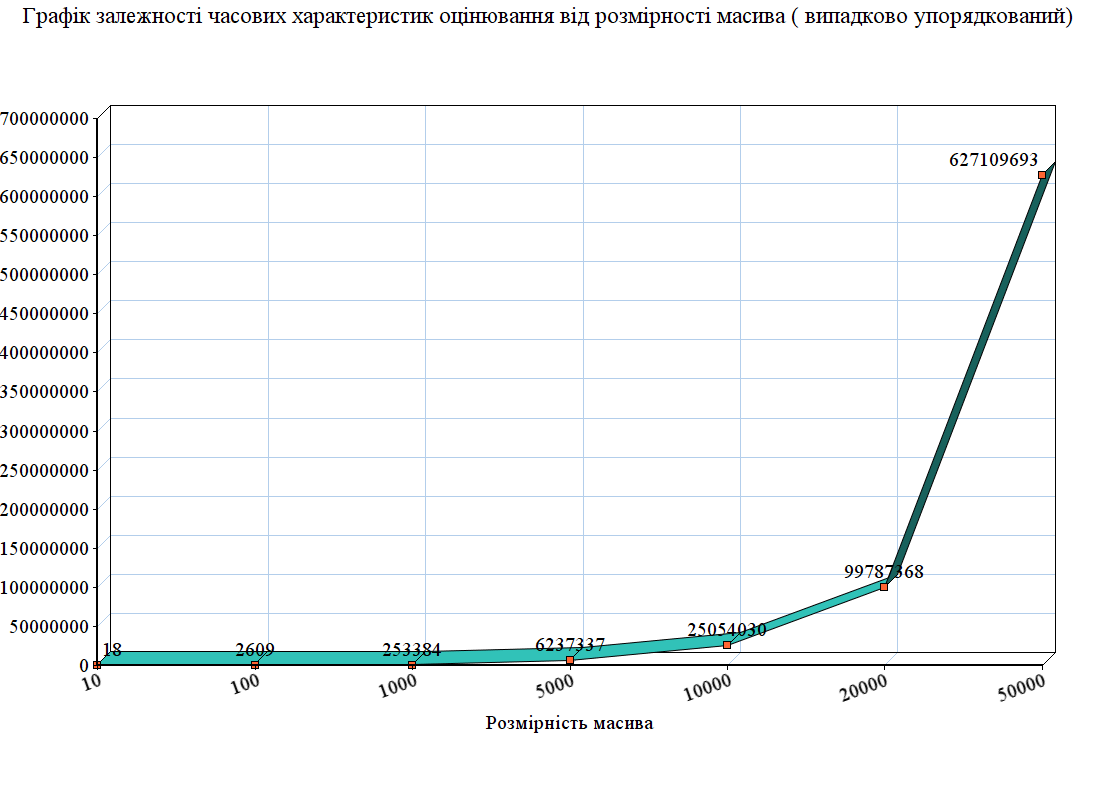
|  |  |  |
| --- | --- | --- |
| Розмірність масиву | Число порівнянь | Число перестановок |
| 10 | 45 | 18 |
| 100 | 4 950 | 2 609 |
| 1000 | 499 500 | 253 384 |
| 5000 | 12 497 500 | 6 237 337 |
| 10000 | 49 995 000 | 25 054 030 |
| 20000 | 199 990 000 | 99 787 368 |
| 50000 | 1 249 975 000 | 627 109 693 |

### Графіки залежності часових характеристик оцінювання від розмірності масиву

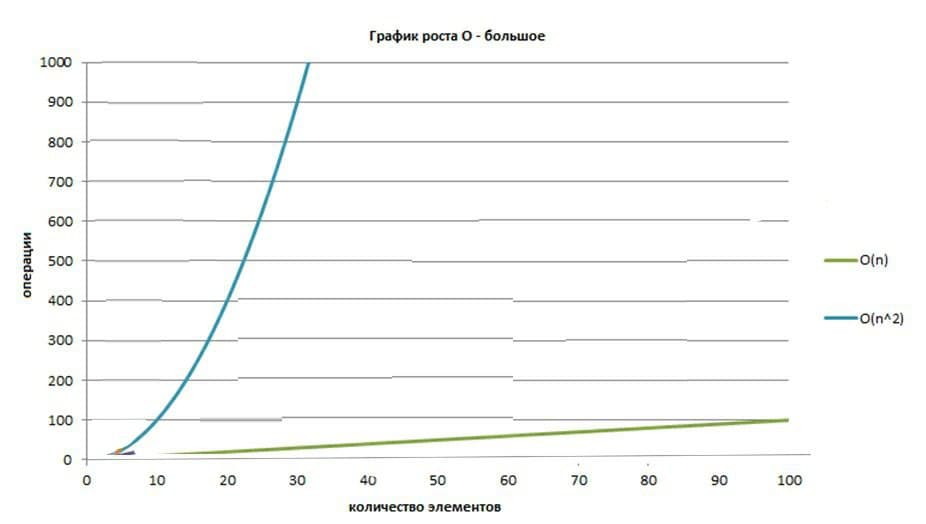
На рисунку 3.3 показані графіки залежності часових характеристик оцінювання від розмірності масиву для випадків, коли масиви містять упорядковану послідовність елементів (зелений графік), коли масиви містять зворотно упорядковану послідовність елементів (червоний графік), коли масиви містять випадкову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

Рисунок 3.3 – Графіки залежності часових характеристик оцінювання





Асимптотична оцінка в кращому і гіршому випадках



## 2.1 Аналіз алгоритму на відповідність властивостям

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

Таблиця 3.1 – Аналіз алгоритму на відповідність властивостям

|  |  |
| --- | --- |
| **Властивість** | **Сортування гребінцем** |
| Стійкість | Алгоритм стійкий |
| «Природність» поведінки (Adaptability) | Алгоритм є природним |
| Базуються на порівняннях | Алгоритм базується на порівняннях |
| Необхідність в додатковій пам'яті (об'єм) | Не потребує додаткової пам’яті |
| Необхідність в знаннях про структури даних | Для використання алгоритму потрібно мати базові знання про структури даних |

## 2.2 Псевдокод алгоритму

**Підпрограма** sort(arr, size)

**Початок**

factor:=1.247

step:=size/factor

**повторити поки step>1**

**повторити для і від 0 до size-step**

j:=i+step

**якщо** arr[j] > arr[j + 1] **то**

temp: = arr[i]

arr[i]: = arr[j]

arr[j]: = temp

**все якщо**

**все повторити**

step:=step/factor

**якщо** flag== 0 **то**

break

**все якщо**

**все повторити**

**Кінець**

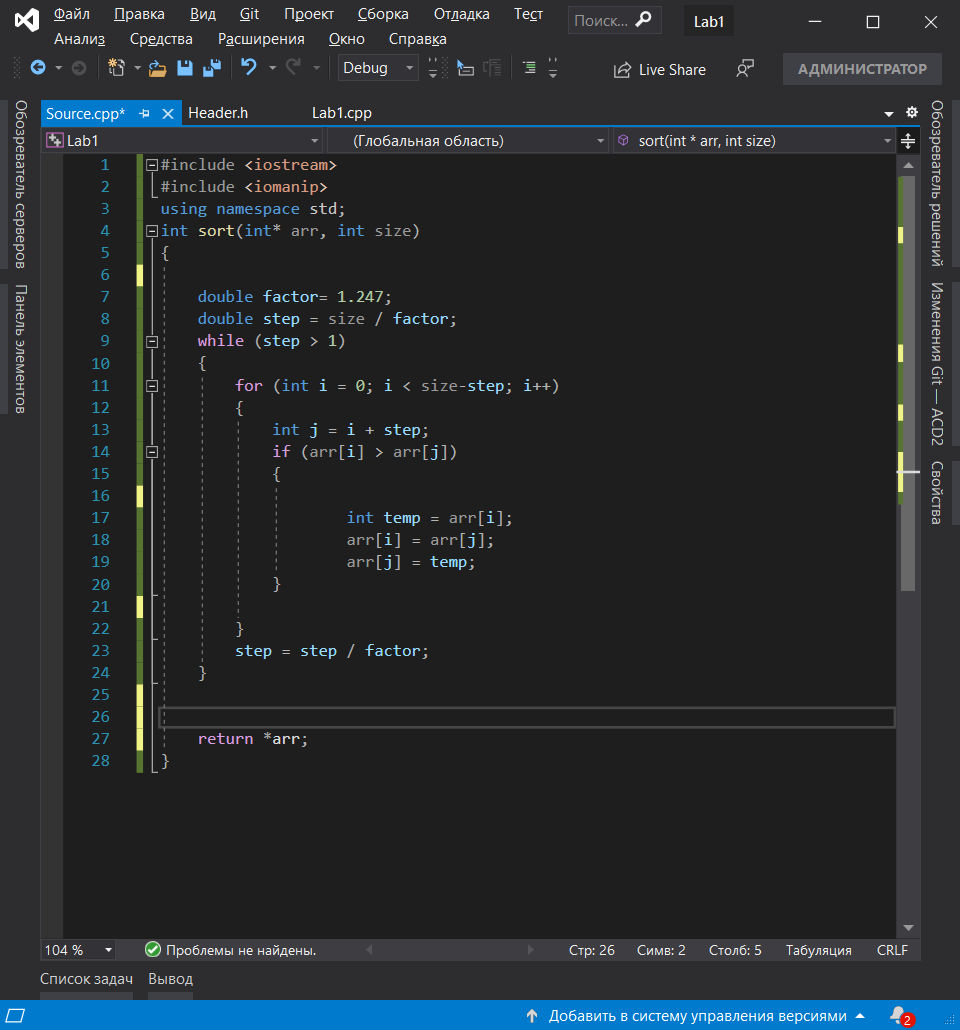
## 2.3 Аналіз часової складності

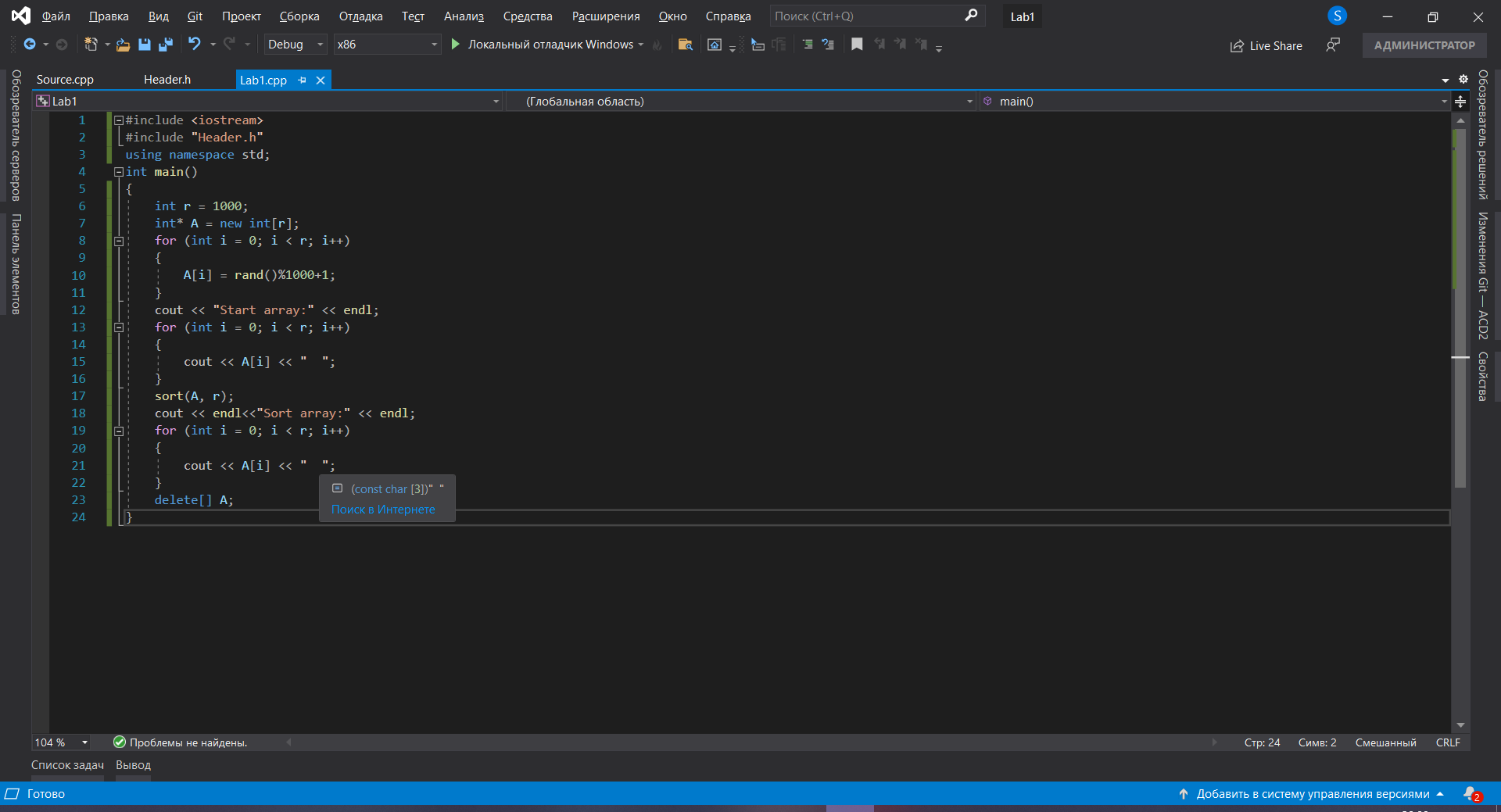
Найгірший випадок алгоритму: O(n^2)

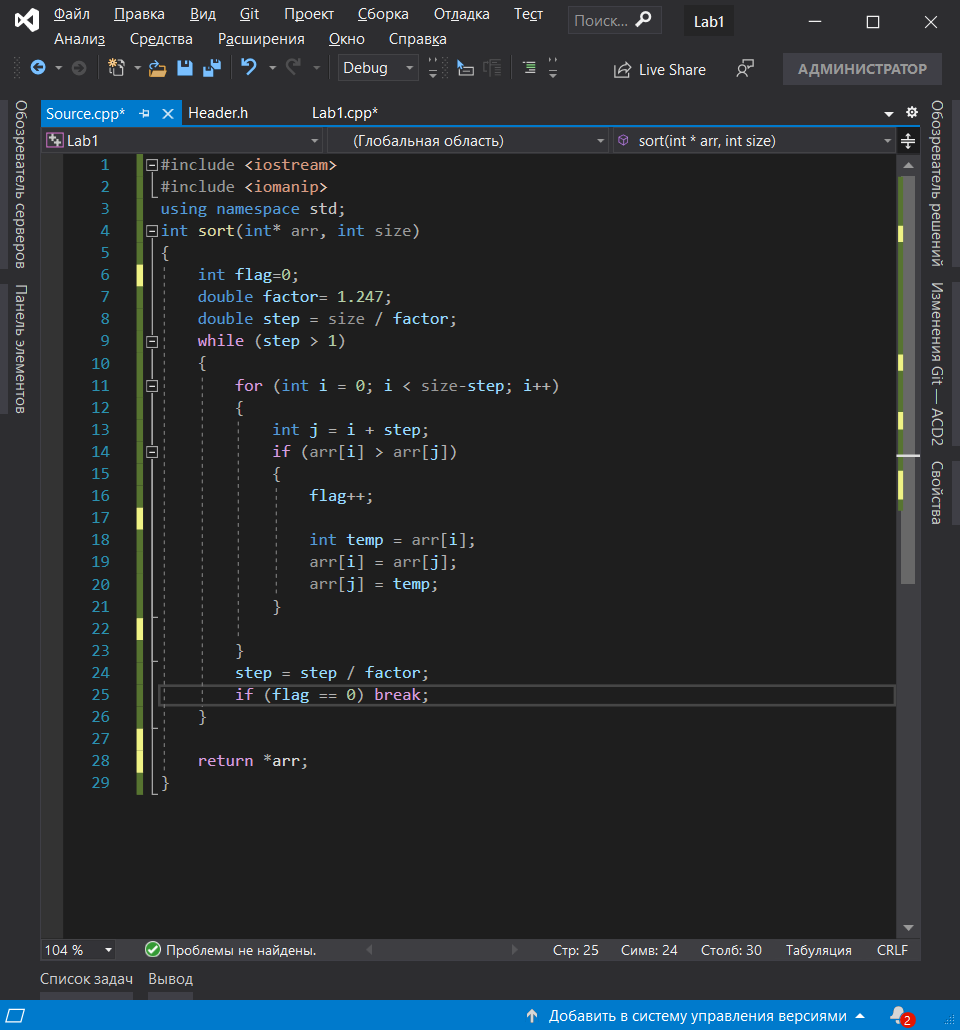
Найкращий випадок: O(n \* log(n))

Середній випадок: O(n \* log(n))

## 2.4 Програмна реалізація алгоритму



2.5 Вихідний код



2.6 Приклад роботи

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

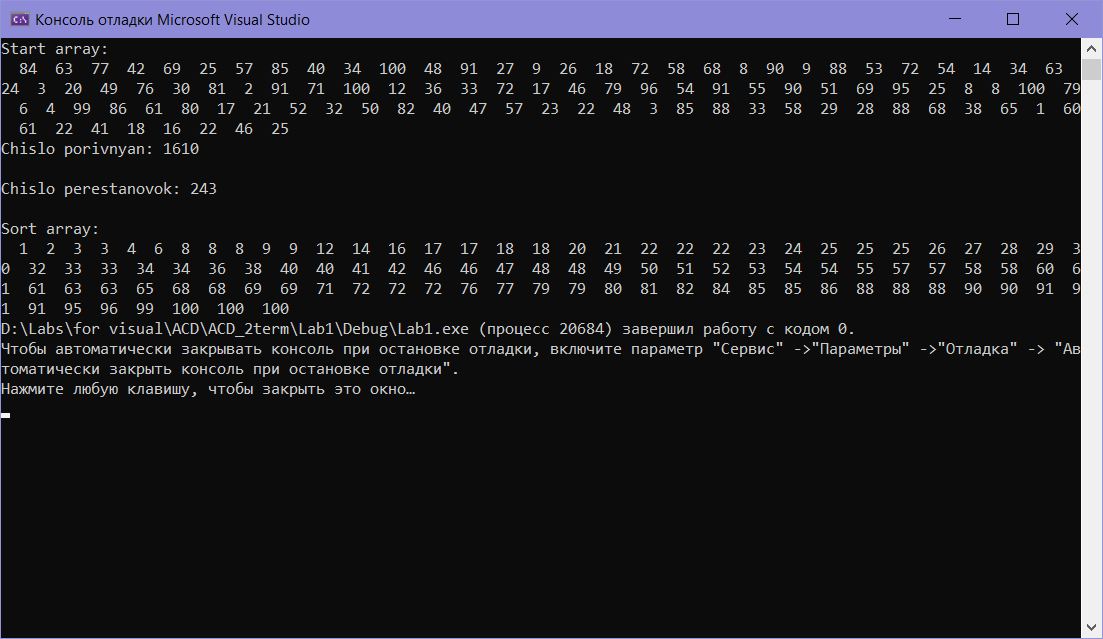
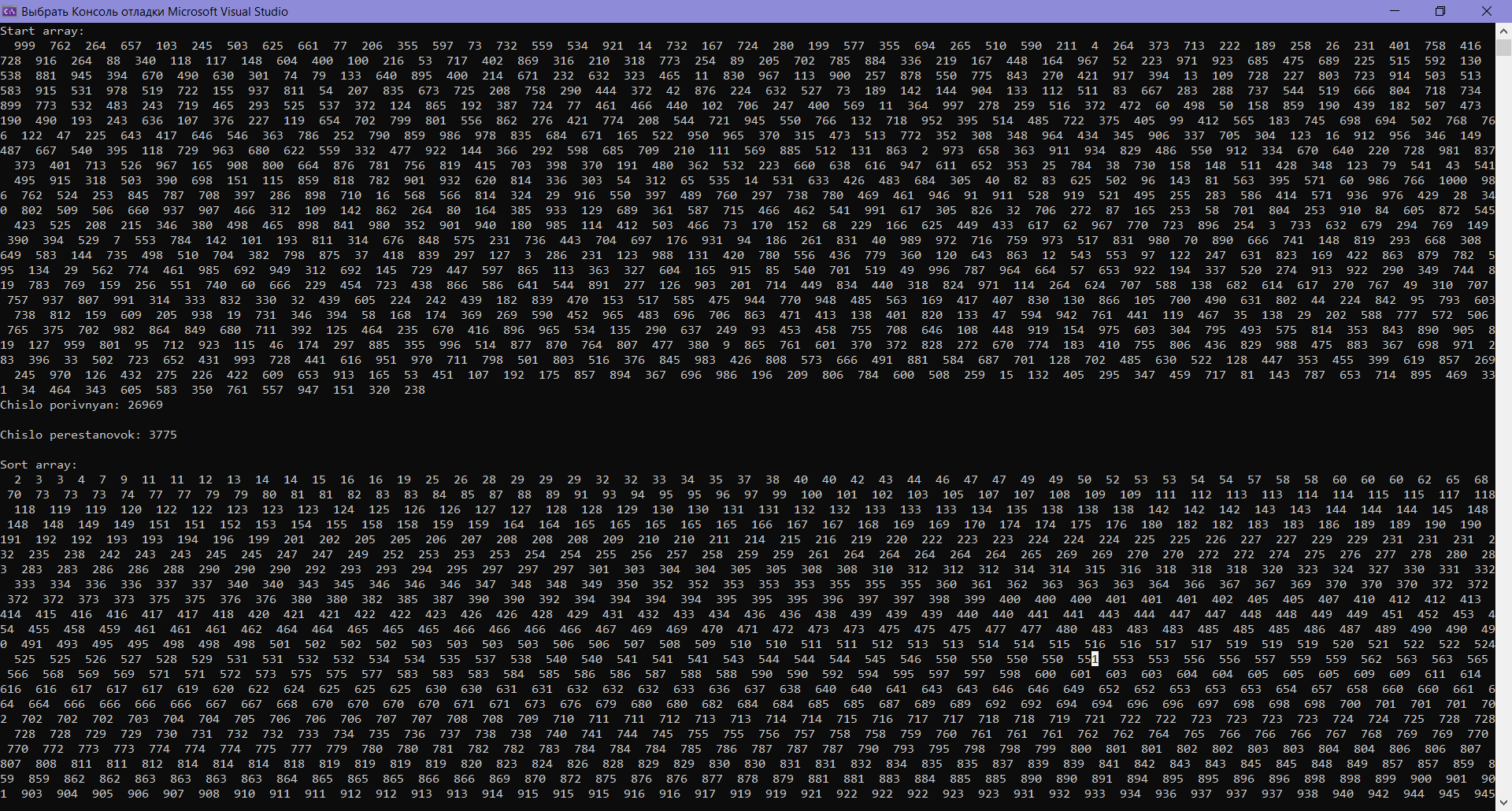
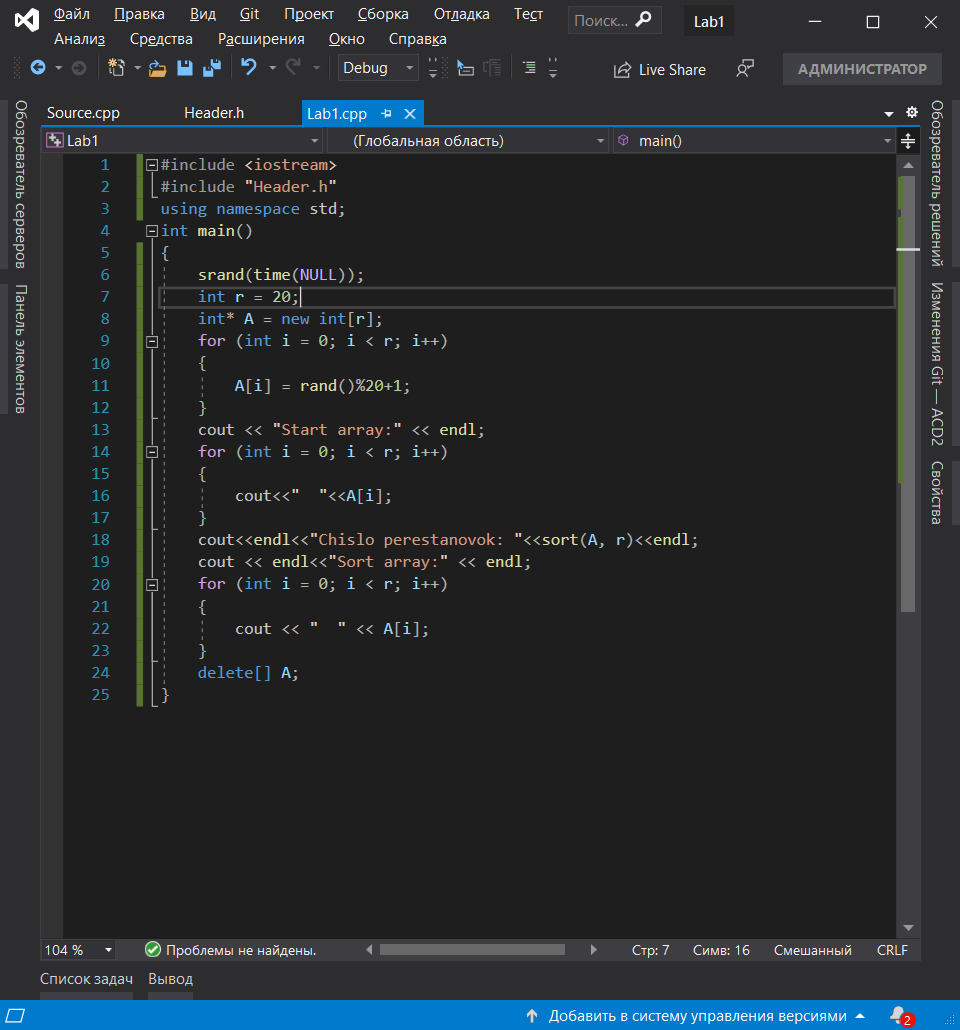
 Рисунок 3.1 – Сортування масиву на 100 елементів

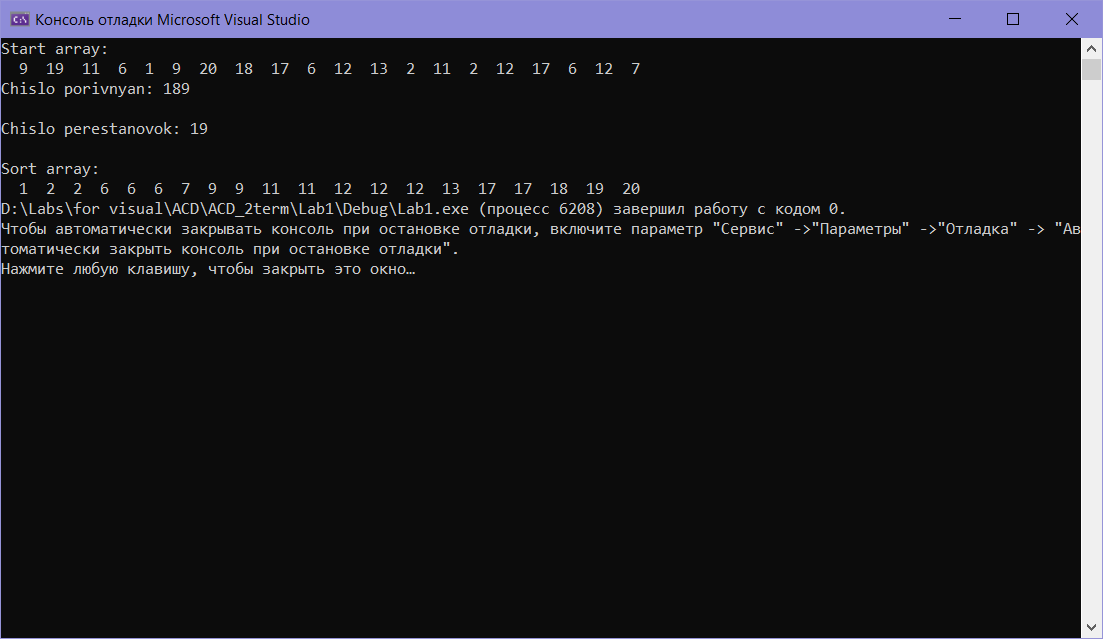
Рисунок 3.2 – Сортування масиву на 1000 елементів

2.7 Тестування алгоритму

Додатково протестуємо алгоритм сортування гребінцем на випадково згенерованому масиві з розмірністю 20 елементів (елементи можуть повторюватись).

Код: Результат роботи:





Як бачимо, алгоритм коректно відсортував випадково згенерований масив. Отже, наш алгоритм сортування гребінцем працює правильно.

2.8 Часові характеристики оцінювання

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

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

|  |  |  |
| --- | --- | --- |
| Розмірність масиву | Число порівнянь | Число перестановок |
| 10 | 2 | 0 |
| 100 | 20 | 0 |
| 1000 | 199 | 0 |
| 5000 | 991 | 0 |
| 10000 | 1981 | 0 |
| 20000 | 3962 | 0 |
| 50000 | 9904 | 0 |

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

|  |  |  |
| --- | --- | --- |
| Розмірність масиву | Число порівнянь | Число перестановок |
| 10 | 67 | 7 |
| 100 | 1610 | 108 |
| 1000 | 26969 | 1474 |
| 5000 | 169778 | 8732 |
| 10000 | 369539 | 18812 |
| 20000 | 799052 | 40786 |
| 50000 | 2247599 | 108878 |

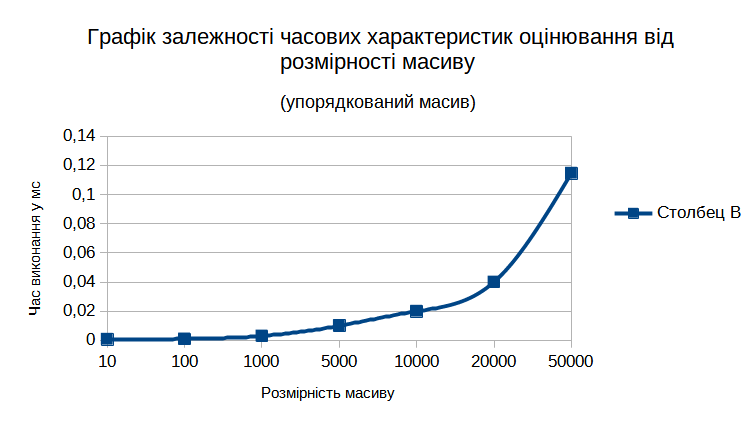
У таблиці 3.4 наведені характеристики оцінювання числа порівнянь і числа перестановок алгоритму сортування гребінцем для масивів різної розмірності, масиви містять випадкову послідовність елементів.

Таблиця 3.4 – Характеристика оцінювання алгоритму сортування гребінцем для випадкової послідовності елементів у масиві.

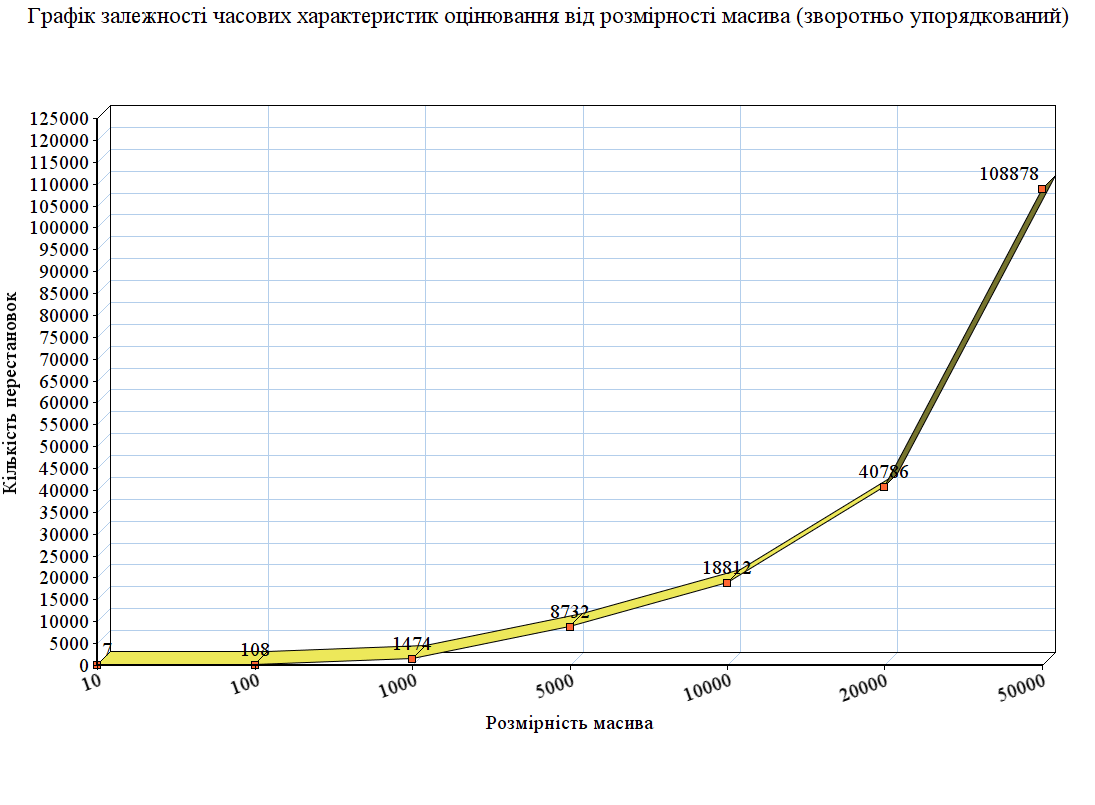
|  |  |  |
| --- | --- | --- |
| Розмірність масиву | Число порівнянь | Число перестановок |
| 10 | 67 | 10 |
| 100 | 1610 | 218 |
| 1000 | 26969 | 3943 |
| 5000 | 169778 | 25311 |
| 10000 | 369539 | 56774 |
| 20000 | 799052 | 124309 |
| 50000 | 2247599 | 340961 |

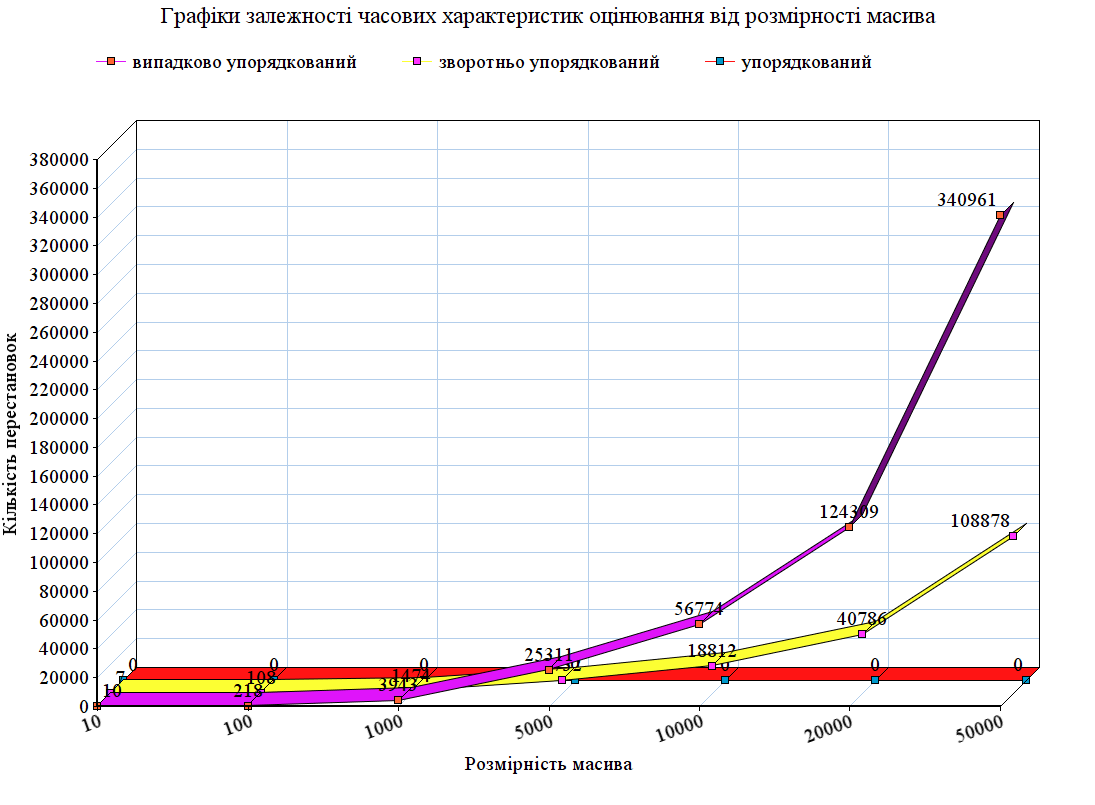
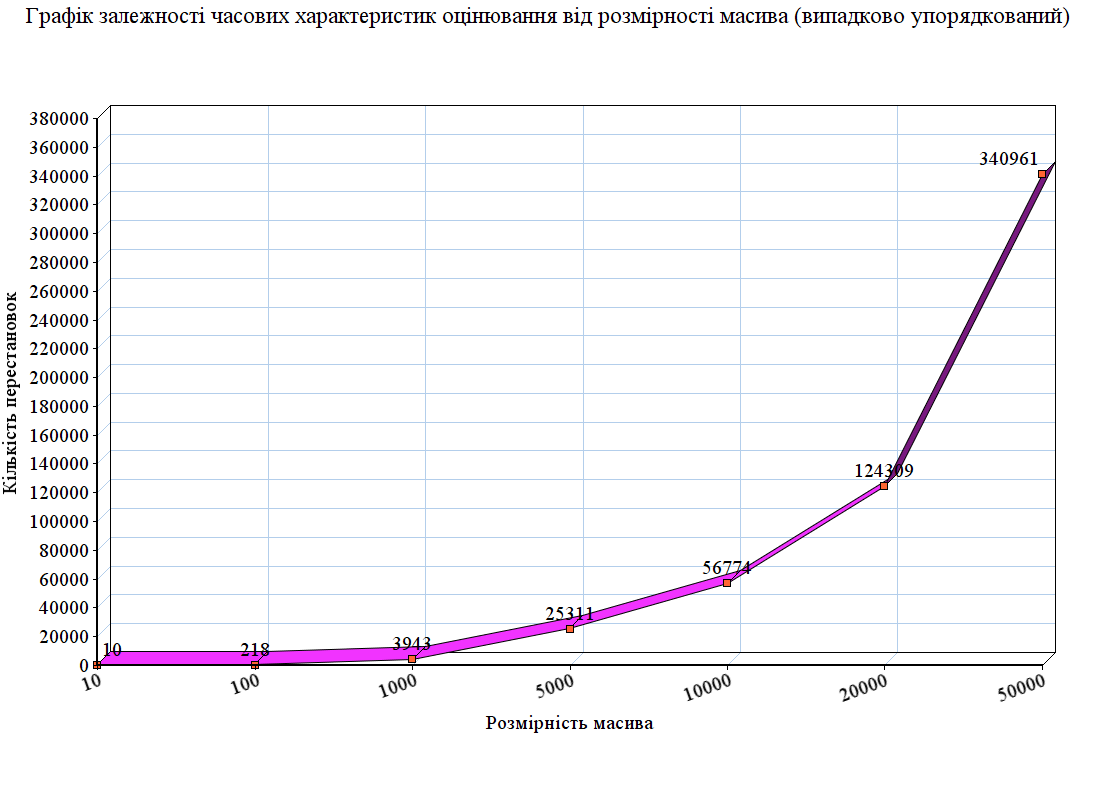
### Графіки залежності часових характеристик оцінювання від розмірності масиву

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

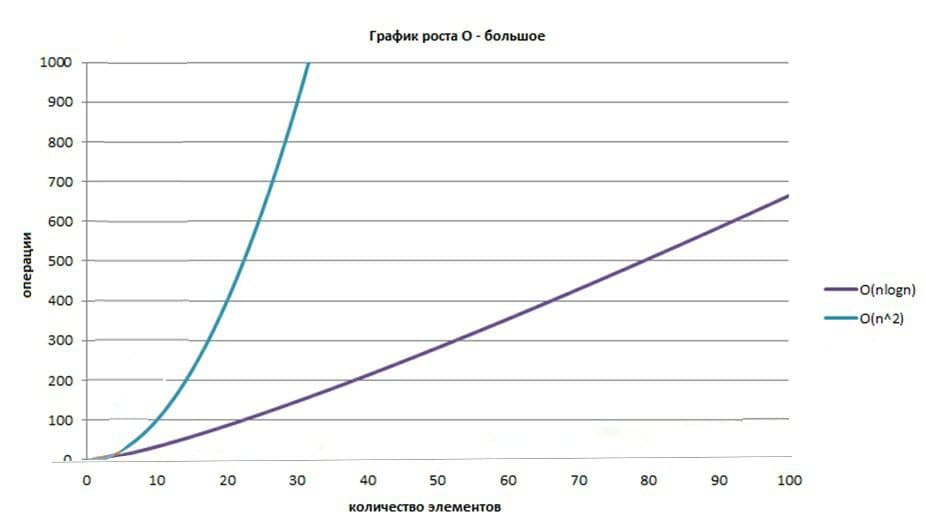
Рисунок 3.3 – Графіки залежності часових характеристик оцінювання







Асимптотична оцінка в гіршому і кращому випадках



Висновок

У ході лабораторної роботи ми вивчили основні методи аналізу обчислювальної складності алгоритмів внутрішнього сортування, застосувалиці знання на практичному завданні і оцінили поріг ефективності кожного методу. При виконанні порівняльного аналізу алгоритмів сортування бульбашкою і гребінцем, ми зробили висновок, що сортування гребінцем є більш швидким алгоритмом. Переконатися у цьому можна, подивившись на графіки залежності часових характеристик оцінювання від розмірності масиву.

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%.