

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.....	12
3.4.2	Приклад роботи.....	14
3.5	ТЕСТУВАННЯ АЛГОРИТМУ	17
3.5.1	Часові характеристики оцінювання.....	18
3.5.2	Графіки залежності часових характеристик оцінювання від розмірності масиву.....	20
	ВИСНОВОК	22
	КРИТЕРІЇ ОЦІНЮВАННЯ	23

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	Стійкий
«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.

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 $O(n \log n)$
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

«Бульбашка»

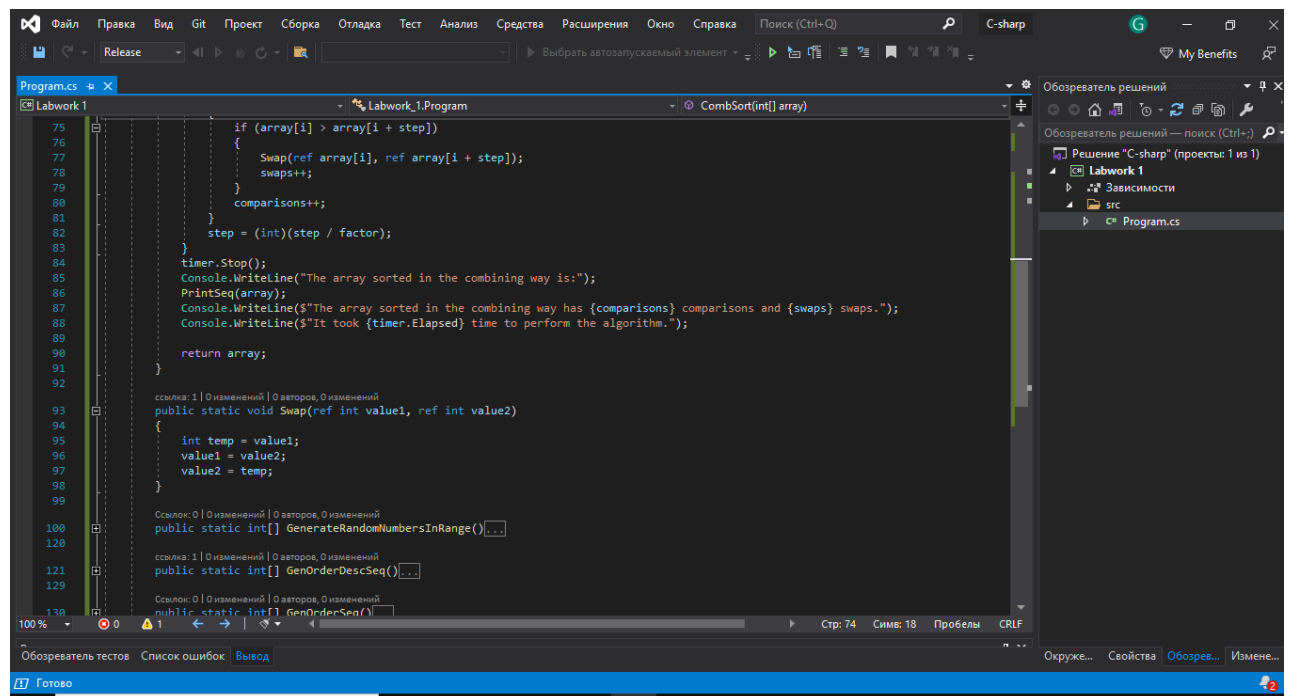
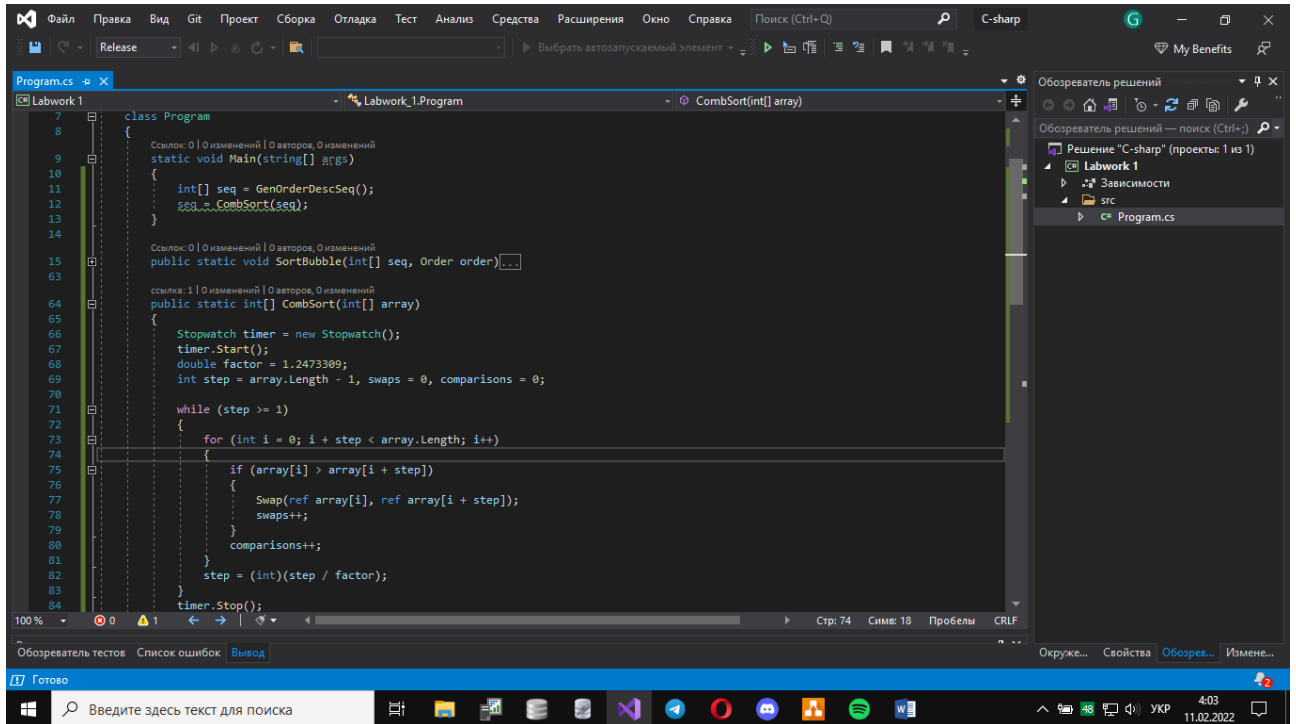
This screenshot shows the Visual Studio IDE with a C# file named Program.cs. The code implements the SortBubble method for ascending order. The method signature is `public static void SortBubble(int[] seq, Order order)`. It starts by creating a Stopwatch, starting it, and initializing comparison and swap counters. It then enters a loop for the order of sorting. For ascending order, it uses a nested loop to compare adjacent elements and swap them if they are in the wrong order. The code is currently at line 36, where a swap is performed.

```
18  ссылка: 1 | О изменениях | О авторе, О изменениях
19  public static void SortBubble(int[] seq, Order order)
20  {
21      Stopwatch timer = new Stopwatch();
22      timer.Start();
23      long comparison = 0, swaps = 0;
24      int temp = 0;
25
26      if (order is Order.Ascending)
27      {
28          for (int y = 0; y < seq.Length - 1; y++)
29          {
30              for (int i = 1; i < seq.Length; i++)
31              {
32                  if (seq[i - 1] > seq[i])
33                  {
34                      temp = seq[i];
35                      seq[i] = seq[i - 1];
36                      seq[i - 1] = temp;
37                      swaps++;
38                  }
39                  comparison++;
40              }
41          }
42      }
43      else if (order is Order.Descending)
44      {
45          for (int y = 0; y < seq.Length - 1; y++)
46          {
47              for (int i = 1; i < seq.Length; i++)
48              {
49                  if (seq[i - 1] < seq[i])
```

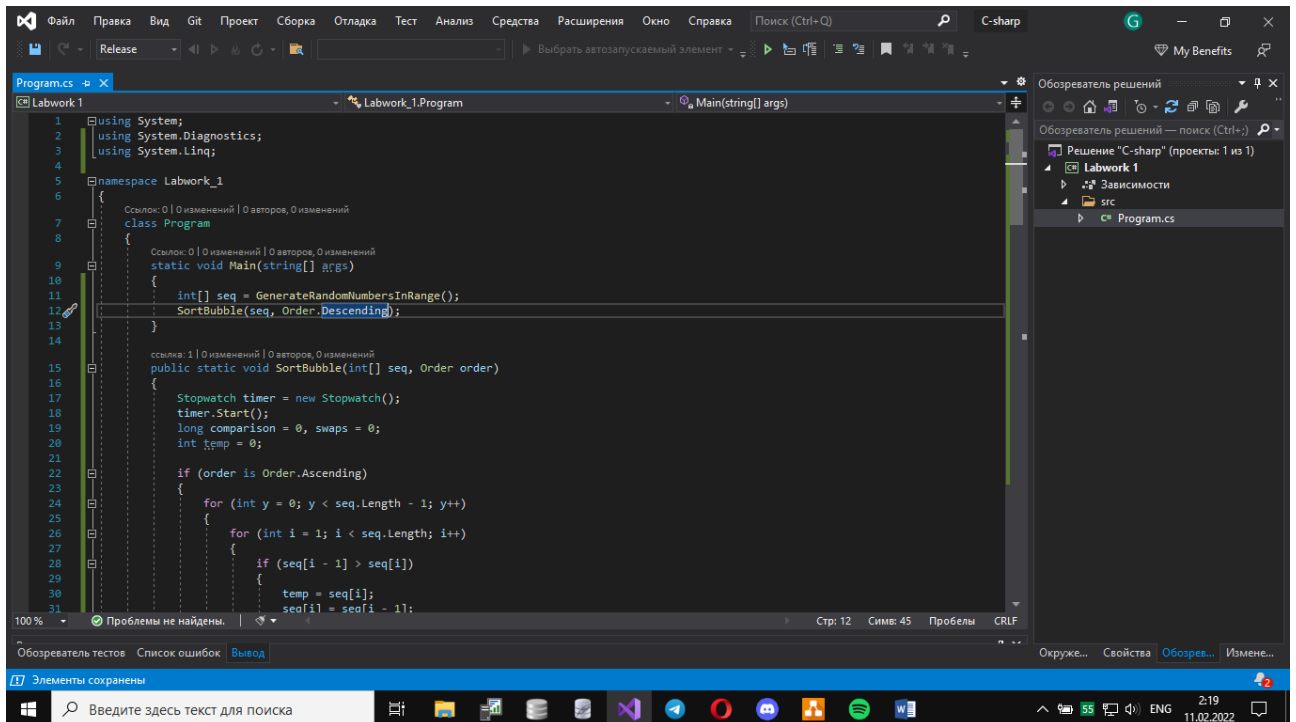
This screenshot shows the continuation of the SortBubble method and the main program logic. The method completes its sorting loop and stops the timer. It then prints the sorted sequence and the number of comparisons and swaps. The main program logic generates a random sequence of numbers and calls the SortBubble method.

```
50                  temp = seq[i];
51                  seq[i] = seq[i - 1];
52                  seq[i - 1] = temp;
53                  swaps++;
54              }
55              comparison++;
56          }
57      }
58      timer.Stop();
59      Console.WriteLine("The new sorted sequence:");
60      PrintSeq(seq);
61      Console.WriteLine($"The bubble-sorted sequence with the next" +
62      $" count of the comparisons is - {comparison}; swaps - {swaps} is: ");
63      Console.WriteLine($"It takes {timer.Elapsed} time to perform the algorithm.");
64  }
65
66  Ссылка: 0 | О изменениях | О авторе, О изменениях
67  public static int[] GenerateRandomNumbersInRange()
68  {
69      int n = 0;
```

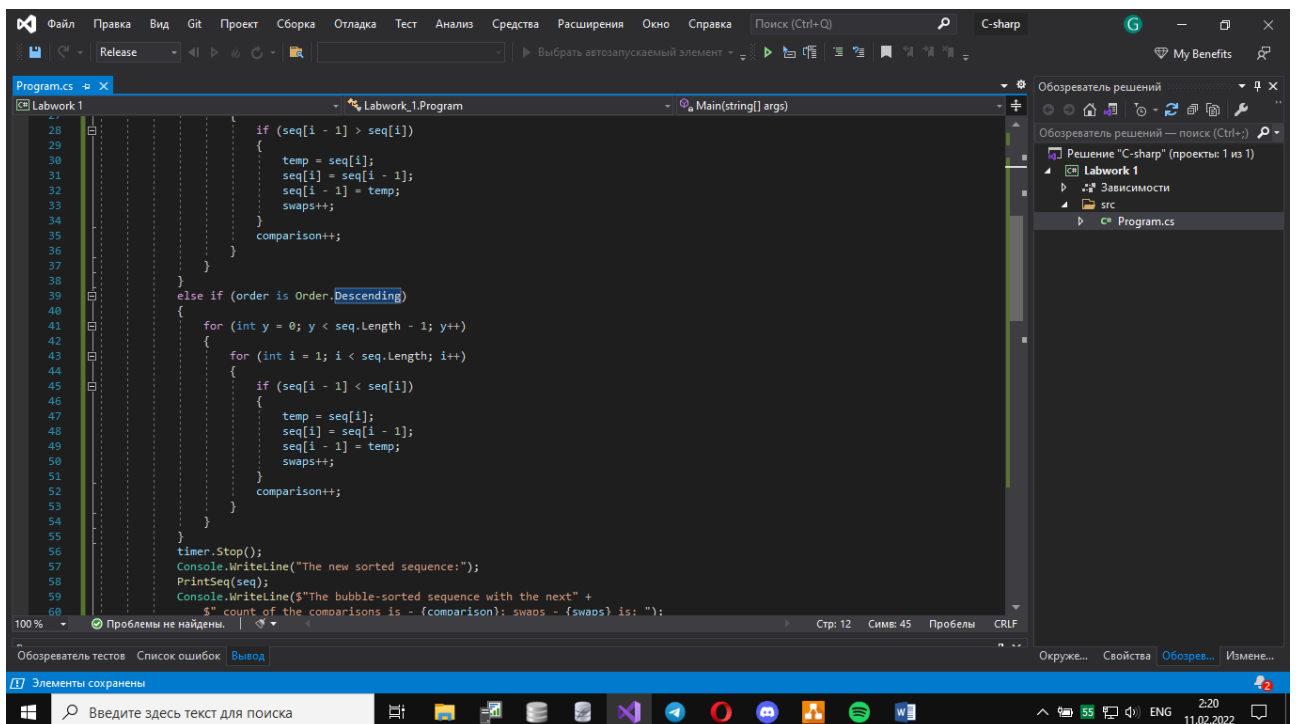
«Гребінець»



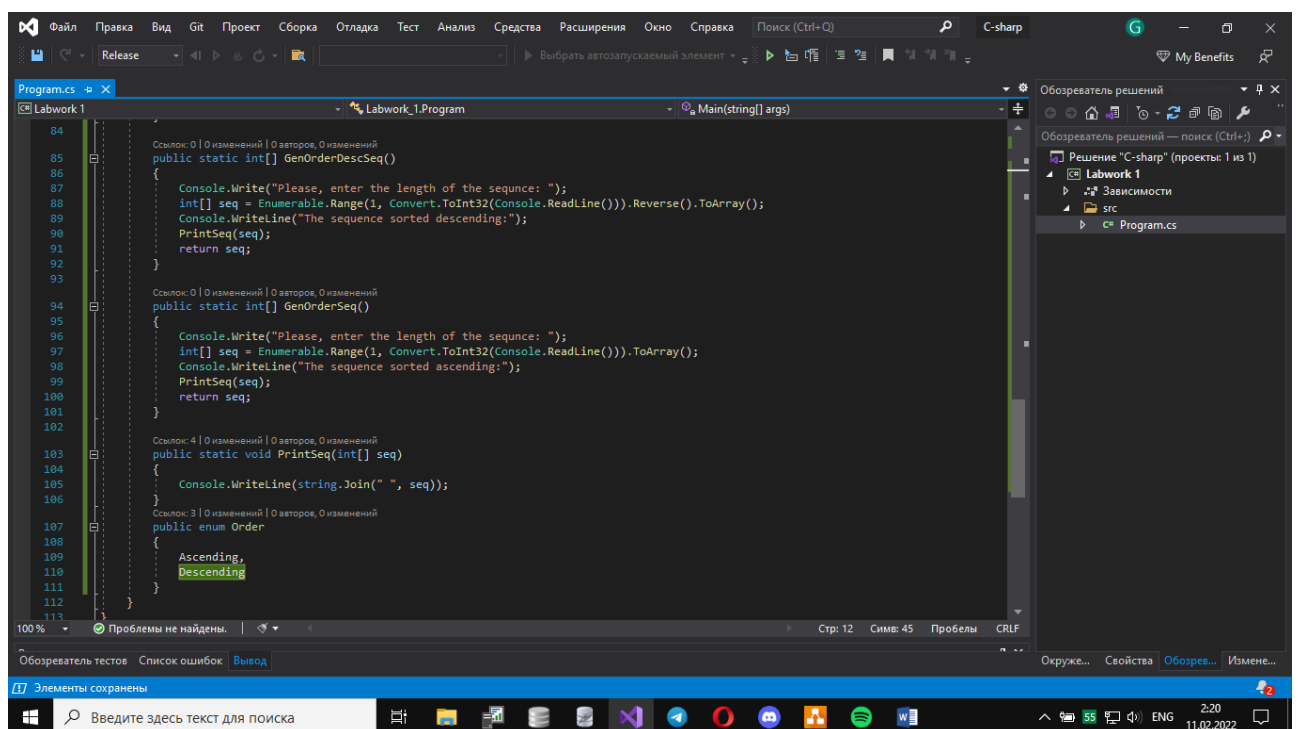
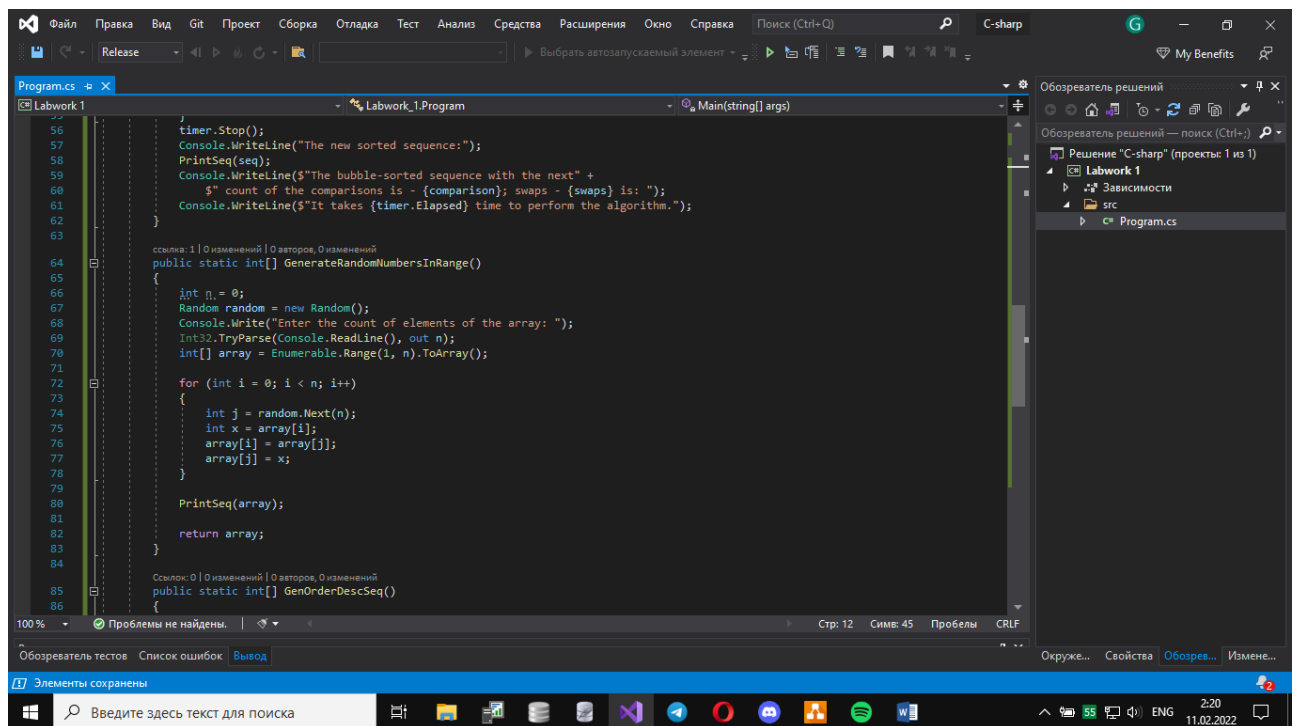
1.4.1 Вихідний код

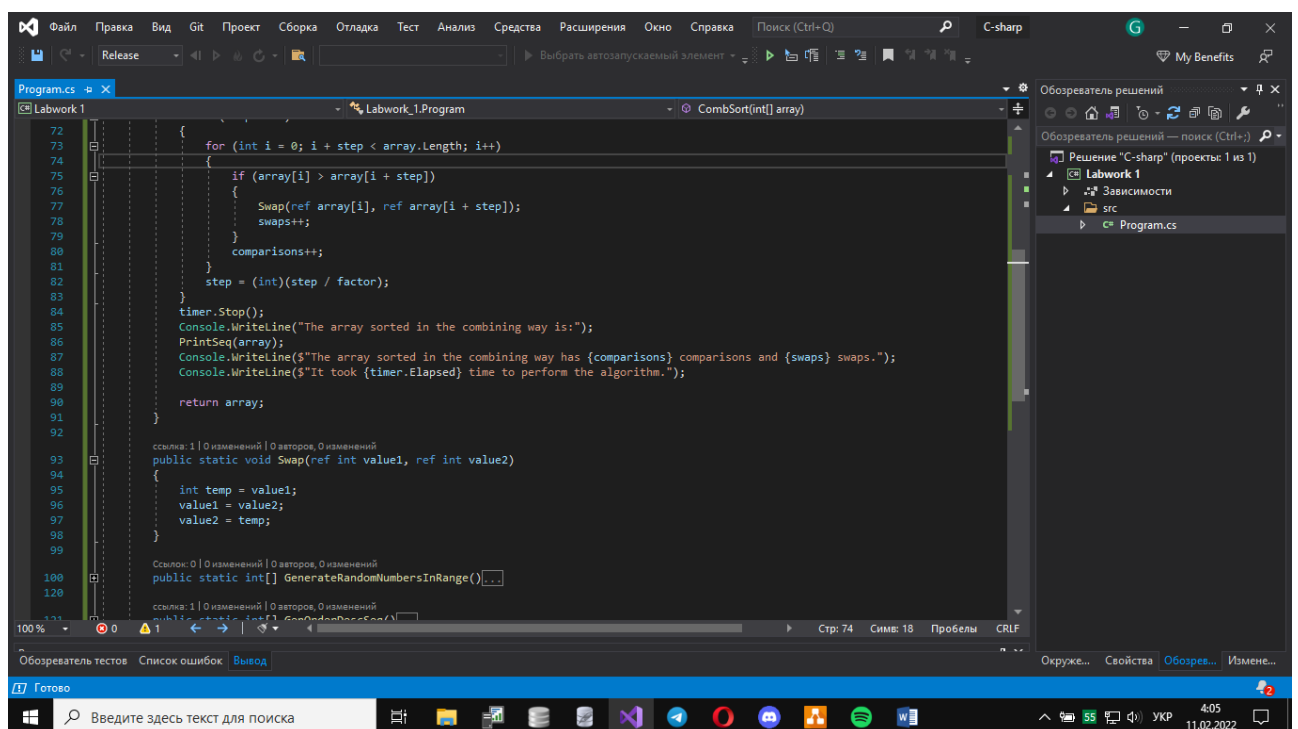
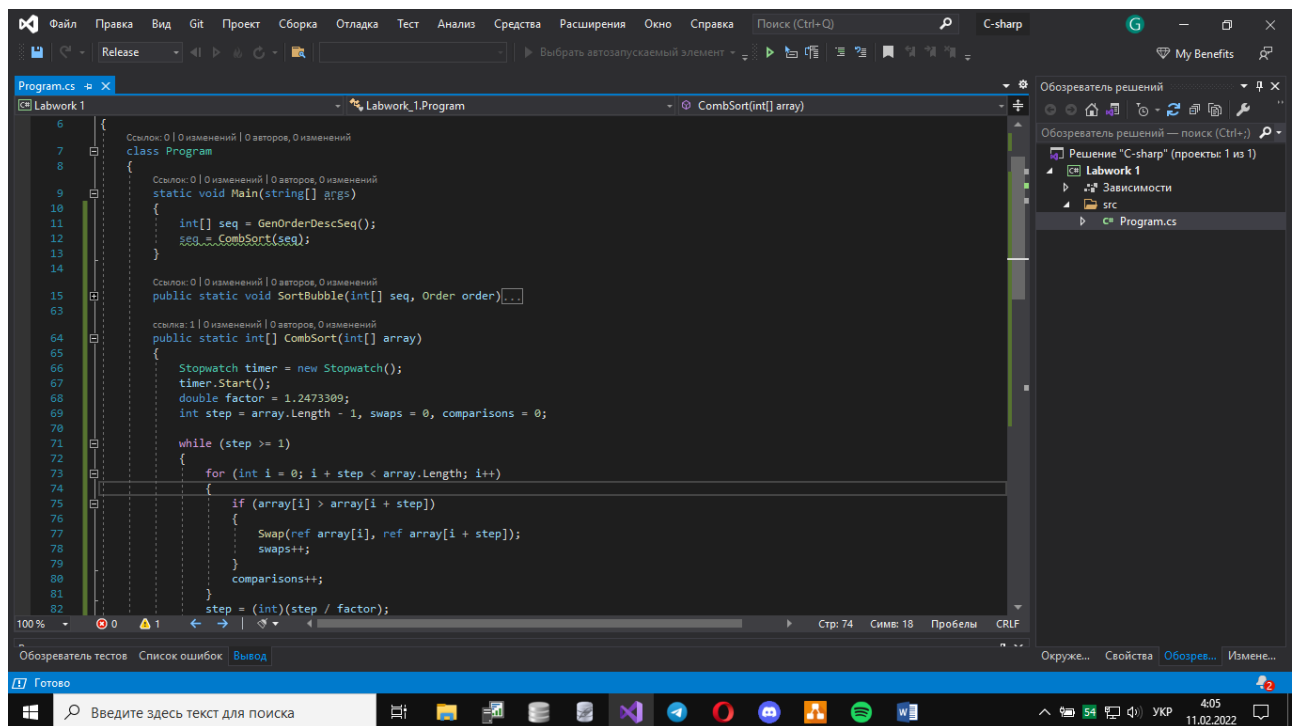


```
1 using System;
2 using System.Diagnostics;
3 using System.Linq;
4
5 namespace Labwork_1
6 {
7     //Ссылка: 0 | Изменений | 0 авторов, 0 изменений
8     class Program
9     {
10         //Ссылка: 0 | Изменений | 0 авторов, 0 изменений
11         static void Main(string[] args)
12         {
13             int[] seq = GenerateRandomNumbersInRange();
14             SortBubble(seq, Order.Descending);
15         }
16
17         //ссылка: 1 | Изменений | 0 авторов, 0 изменений
18         public static void SortBubble(int[] seq, Order order)
19         {
20             Stopwatch timer = new Stopwatch();
21             timer.Start();
22             long comparison = 0, swaps = 0;
23             int temp = 0;
24
25             if (order is Order.Ascending)
26             {
27                 for (int y = 0; y < seq.Length - 1; y++)
28                 {
29                     for (int i = 1; i < seq.Length; i++)
30                     {
31                         if (seq[i - 1] > seq[i])
32                         {
33                             temp = seq[i];
34                             seq[i] = seq[i - 1];
35                             seq[i - 1] = temp;
36                             swaps++;
37                         }
38                         comparison++;
39                     }
40                 }
41             }
42             timer.Stop();
43             Console.WriteLine("The new sorted sequence:");
44             PrintSeq(seq);
45             Console.WriteLine($"The bubble-sorted sequence with the next" +
46                             $" count of the comparisons is - {comparison}; swaps - {swaps} is: ");
47         }
48     }
49 }
```



```
28         if (seq[i - 1] > seq[i])
29         {
30             temp = seq[i];
31             seq[i] = seq[i - 1];
32             seq[i - 1] = temp;
33             swaps++;
34         }
35         comparison++;
36     }
37 }
38
39 else if (order is Order.Descending)
40 {
41     for (int y = 0; y < seq.Length - 1; y++)
42     {
43         for (int i = 1; i < seq.Length; i++)
44         {
45             if (seq[i - 1] < seq[i])
46             {
47                 temp = seq[i];
48                 seq[i] = seq[i - 1];
49                 seq[i - 1] = temp;
50                 swaps++;
51             }
52             comparison++;
53         }
54     }
55 }
56 timer.Stop();
57 Console.WriteLine("The new sorted sequence:");
58 PrintSeq(seq);
59 Console.WriteLine($"The bubble-sorted sequence with the next" +
60                 $" count of the comparisons is - {comparison}; swaps - {swaps} is: ");
61 }
```





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»

```
Program
Enter the count of elements of the array: 100
44 87 93 15 8 1 31 60 49 30 92 84 40 51 18 39 56 3 48 74 32 99 85 10 35 11 98 46 22 59 97 47 52 58 28 37 27 96 79 7 86 7
512 68 5 81 29 14 54 20 53 57 38 45 95 12 64 9 88 26 69 66 34 77 24 94 71 76 75 21 6 82 80 13 78 73 50 67 23 25 65 2 62 36
52 4 19 63 43 90 100 41 61 33 91 55 70 89 17 16 42 83
53 The new sorted sequence:
54 100 99 98 97 96 95 94 93 92 91 90 89 88 87 86 85 84 83 82 81 80 79 78 77 76 75 74 73 72 71 70 69 68 67 66 65 64 63 62 61
55 60 59 58 57 56 55 54 53 52 51 50 49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21
56 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
57 The bubble-sorted sequence with the next count of the comparisons is - 9801; swaps - 2566 is:
58 It takes 00:00:00.0000334 time to perform the algorithm.
59
60 C:\Users\Дима\Desktop\Studying\Labs\algorithms2term\C-sharp\Labwork 1\bin\Release\netcoreapp3.1\Labwork 1.exe (процесс 8
61 1452) завершил работу с кодом 0.
62 Нажмите любую клавишу, чтобы закрыть это окно...
63
64
65
66
67
68
69
70
71
72
73
74
75
76
100 %
Вывод
Показать выходные данные из: Сборка
Обозреватель тестов Список ошибок... Вывод
Готово
Введите здесь текст для поиска
2:22 11.02.2022
```

«Гребінець»

```
Program.cs
Labwork 1
using System;
using System.Diagnostics;
using System.Linq;
namespace
{
    class Program
    {
        static void Main(string[] args)
        {
            Enter the count of elements of the array: 100
            16 46 7 38 53 85 76 9 100 8 72 22 84 12 69 26 11 96 71 70 57 23 39 34 28 36 47 62 45 55 98 44 35 37 50 31 1 90 73 24 91
            6 19 56 52 32 64 54 63 42 13 58 30 88 51 18 87 75 80 49 81 14 27 2 99 33 29 82 17 68 92 94 20 74 97 59 21 65 77 83 43 25
            10 4 86 48 93 95 41 40 5 79 89 3 15 78 61 66 67 60
            The array sorted in the combining way is:
            1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 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
            The array sorted in the combining way has 1233 comparisons and 235 swaps.
            It took 00:00:00.0000109 time to perform the algorithm.
            C:\Users\Дима\Desktop\Studying\Labs\algorithms2term\C-sharp\Labwork 1\bin\Release\netcoreapp3.1\Labwork 1.exe (процесс 3
            P816) завершил работу с кодом 0.
            Нажмите любую клавишу, чтобы закрыть это окно...
        }
    }
}
100 %
Вывод
Показать выходные данные
Обозреватель тестов Список ошибок... Вывод
Сборка успешно завершена
Введите здесь текст для поиска
4:06 11.02.2022
```


Рисунок 3.2 – Сортунання масиву на 1000 елементів

«Бульбашка»

```
Консоль отладки Microsoft Visual Studio
39 740 810 99 44 683 490 854 458 967 980 238 138 1 111 627 50 850 689 249 287 908 61 464 910 195 592 330 151 868 607 547 260 873 193 374 298 531 186 359 424 53 442 756
696 277 656 9 746 550 907 289 48 545 324 519 983 419 335 96 137 608 881 954 14 729 720 311 391 386 664 148 700 706 806 666 771 649 651 17 125 697 46 832 505 177 185 708
343 860 947 319 792 727 48 846 881 898 996 712 811 506 659 674 871 251 369 273 736 242 358 418 768 630 218 824 406 643 722 113 673 134 961 973 452 430 274 828 432 766
309 624 655 641 440 974 129 415 136 483 764 845 27 829 657 999 862 387 826 253 132 839 611 437 481 514 804 964 261 892 214 204 175 675 245 310 212 122 402 232 331 256 3
78 690 642 323 35 866 78 79 938 445 903 911 58 376 585 423 19 905 384 192 931 757 217 646 470 606 351 86 223 180 705 203 142 403 552 563 139 49 434 566 344 587 844 788
849 741 300 24 72 943 714 231 963 942 208 329 197 390 842 946 348 889 221 236 234 308 366 444 576 677 13 334 620 945 960 492 541 512 622 542 357 382 476 962 36 784 67 5
95 108 489 317 4 614 278 880 586 888 226 8 392 496 864 621 658 896 305 128 179 623 781 33 749 176 356 209 562 396 539 379 52 388 540 989 315 723 698 414 790 953 913 758
861 495 951 269 775 680 950 31 887 457 990 32 467 393 958 70 284 777 372 663 281 182 560 886 957 926 795 588 774 34 639 271 216 902 213 572 615 422 462 150 820 821 730
425 266 660 394 477 149 2 51 684 686 968 373 786 930 328 975 616 84 110 131 528 685 466 769 979 981 742 626 767 117 428 793 173 709 81 782 25 275 733 69 498 924 235 47
4 915 985 75 944 952 671 417
The new sorted sequence:
1000 999 998 997 996 995 994 993 992 991 990 989 988 987 986 985 984 983 982 981 980 979 978 977 976 975 974 973 972 971 970 969 968 967 966 965 964 963 962 961 960 959
958 957 956 955 954 953 952 951 950 949 948 947 946 945 944 943 942 941 940 939 938 937 936 935 934 933 932 931 930 929 928 927 926 925 924 923 922 921 920 919 918 917
916 915 914 913 912 911 910 909 908 907 906 905 904 903 902 901 900 899 898 897 896 895 894 893 892 891 890 889 888 887 886 885 884 883 882 881 880 879 878 877 876 875
874 873 872 871 870 869 868 867 866 865 864 863 862 861 860 859 858 857 856 855 854 853 852 851 850 849 848 847 846 845 844 843 842 841 840 839 838 837 836 835 834 833
832 831 830 829 828 827 826 825 824 823 822 821 820 819 818 817 816 815 814 813 812 811 810 809 808 807 806 805 804 803 802 801 800 799 798 797 796 795 794 793 792 791
790 789 788 787 786 785 784 783 782 781 780 779 778 777 776 775 774 773 772 771 770 769 768 767 766 765 764 763 762 761 760 759 758 757 756 755 754 753 752 751 750 749
748 747 746 745 744 743 742 741 740 739 738 737 736 735 734 733 732 731 730 729 728 727 726 725 724 723 722 721 720 719 718 717 716 715 714 713 712 711 710 709 708 707
706 705 704 703 702 701 700 699 698 697 696 695 694 693 692 691 690 689 688 687 686 685 684 683 682 681 680 679 678 677 676 675 674 673 672 671 670 669 668 667 666 665
664 663 662 661 660 659 658 657 656 655 654 653 652 651 650 649 648 647 646 645 644 643 642 641 640 639 638 637 636 635 634 633 632 631 630 629 628 627 626 625 624 623
622 621 620 619 618 617 616 615 614 613 612 611 610 609 608 607 606 605 604 603 602 601 600 599 598 597 596 595 594 593 592 591 590 589 588 587 586 585 584 583 582 581
580 579 578 577 576 575 574 573 572 571 570 569 568 567 566 565 564 563 562 561 560 559 558 557 556 555 554 553 552 551 550 549 548 547 546 545 544 543 542 541 540 539
538 537 536 535 534 533 532 531 530 529 528 527 526 525 524 523 522 521 520 519 518 517 516 515 514 513 512 511 510 509 508 507 506 505 504 503 502 501 500 499 498 497
496 495 494 493 492 491 490 489 488 487 486 485 484 483 482 481 480 479 478 477 476 475 474 473 472 471 470 469 468 467 466 465 464 463 462 461 460 459 458 457 456 455
454 453 452 451 450 449 448 447 446 445 444 443 442 441 440 439 438 437 436 435 434 433 432 431 430 429 428 427 426 425 424 423 422 421 420 419 418 417 416 415 414 413
412 411 410 409 408 407 406 405 404 403 402 401 400 399 398 397 396 395 394 393 392 391 390 389 388 387 386 385 384 383 382 381 380 379 378 377 376 375 374 373 372 371
370 369 368 367 366 365 364 363 362 361 360 359 358 357 356 355 354 353 352 351 350 349 348 347 346 345 344 343 342 341 340 339 338 337 336 335 334 333 332 331 330 329
328 327 326 325 324 323 322 321 320 319 318 317 316 315 314 313 312 311 310 309 308 307 306 305 304 303 302 301 300 299 298 297 296 295 294 293 292 291 290 289 288 287
286 285 284 283 282 281 280 279 278 277 276 275 274 273 272 271 270 269 268 267 266 265 264 263 262 261 260 259 258 257 256 255 254 253 252 251 250 249 248 247 246 245
244 243 242 241 240 239 238 237 236 235 234 233 232 231 230 229 228 227 226 225 224 223 222 221 220 219 218 217 216 215 214 213 212 211 210 209 208 207 206 205 204 203
202 201 200 199 198 197 196 195 194 193 192 191 190 189 188 187 186 185 184 183 182 181 180 179 178 177 176 175 174 173 172 171 170 169 168 167 166 165 164 163 162 161
160 159 158 157 156 155 154 153 152 151 150 149 148 147 146 145 144 143 142 141 140 139 138 137 136 135 134 133 132 131 130 129 128 127 126 125 124 123 122 121 120 119
118 117 116 115 114 113 112 111 110 109 108 107 106 105 104 103 102 101 100 99 98 97 96 95 94 93 92 91 90 89 88 87 86 85 84 83 82 81 80 79 78 77 76 75 74 73 72 71 70 69
9 68 67 66 65 64 63 62 61 60 59 58 57 56 55 54 53 52 51 50 49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 1
3 12 11 10 9 8 7 6 5 4 3 2 1
The bubble-sorted sequence with the next count of the comparisons is - 998001; swaps - 261610 is:
It takes 00:00:00.0019363 time to perform the algorithm.

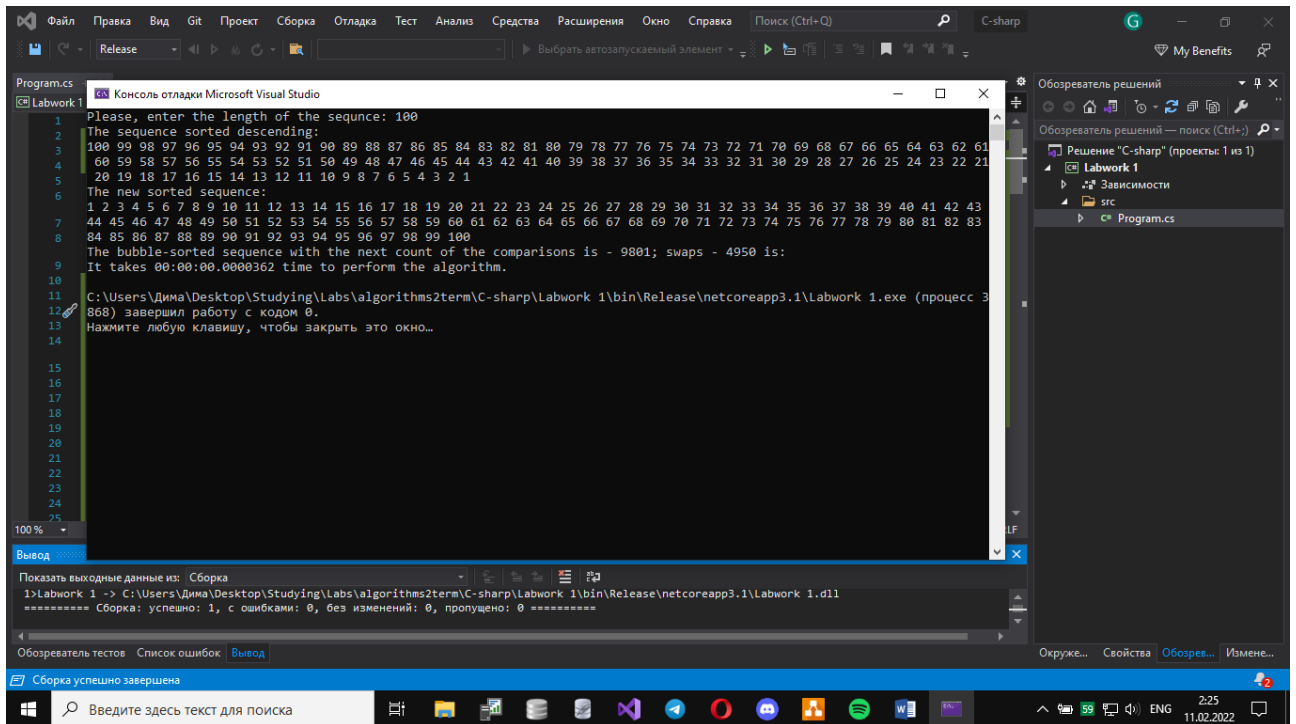
C:\Users\Дима\Desktop\Studying\Labs\algorithms2term\C-sharp\Labwork 1\bin\Release\netcoreapp3.1\Labwork 1.exe (процесс 7080) завершил работу с кодом 0.
Нажмите любую клавишу, чтобы закрыть это окно...
```

«Гребінець»

```
Консоль отладки Microsoft Visual Studio
914 988 564 334 719 877 546 414 666 481 548 995 533 145 570 758 867 769 136 756 68 26 833 640 734 851 455 419 603 217 151 638 960 403 887 201 512 131 283 923 143 795 9
79 984 696 965 842 551 863 767 296 723 754 445 852 200 483 671 451 989 884 105 506 55 271 254 875 844 744 48 657 574 499 29 677 934 517 265 14 7 503 673 598 304 122 27
981 740 579 194 891 23 79 425 344 466 475 468 449 786 110 423 204 883 215 709 843 632 417 495 119 378 280 413 882 620 223 866 712 805 431 383 459 918 268 428 311 514 34
3 398 85 463 150 112 776 397 659 434 587 141 158 400 208 563 783 132 511 110 310 309 308 307 306 305 304 303 302 301 300 299 298 297 296 295 294 293 292 291 290 289 288 287
826 645 153 972 190 552 914 547 929 944 310 295 217 792 402 694 651 649 249 464 435 722 238 300 693 942 513 890 959 897 59 985 4 80 870 39 780 810 903 751 604 115 868 815 102 912 973
5 930 60 369 660 940 60 879 70 149 212 945 775 929 544 257 606 994 477 820 104 167 794 904 605 836 600 353 650 670 33 480 30 780 810 903 751 604 115 868 815 102 912 973
567 195 921 646 704 747 630 924 480 940 954 91 812 705 40 186 316 698 676 697 474 855 741 620 518 860 363 360 70 412 350 987 615 811 830 28 444 814 526 127 974 785 421 256
802 409 362 727 38 585 636 685 951 356 777 69 922 996 889 298 892 197 276 839 53 876 893 952 982 178 969 729 297 799 168 538 240 446 846 789 537 886 355 730 129 607 49
3 718 58 642 525 745 251 713 683 501 433 742 764 926 199 138 230 768 13 978 643 76 700 191 1 681 258 664 841 658 436 796 73 759 728 350 226 865 216 86 625 593 361 57
2 991 771 170 373 822 72 561 648 448 321 589 473 134 95 966 820 245 40 327 126 631 274 43 365 998 505 884 180 990 790 81 927 404 21 624 959 92 424 637 993 787 943 206 2
85 67 993 736 528 885 45 367 779 123 172 162 807 778 177 320 31 452 101 71 340 396 305 202 94 616 437 782 715 687 87 743 491 152 832 622 353 96 163 682 314 688 749 183
970 83 148 187 281 142 408 708 388 535 192 610 487 997 845 160 50 602 971 211 529 5 717 269 963 932 131 580 560 371 37 44 731 36 838 986 975 467 732 242 233 614 667 405
309 896 862 347 292 869 322 219 229 947 54 63 901 555 453 430 75 368 378 672 523 738 710 708 166 690 326 189 808 165 860 898 725 188 54 246 160 12 293 213 854 586 950
571 155 720 956 164 803 22 612 380 730 864 243 389 827 679 496 315 562 289 907 176 707 144 917 480 301 56 385 272 100 447 823 89 273 351 962 978 524 733 847 878 93
7 949 955 930 957 999 443 817 545 763 565 919 653 566 47 618 781 899 837 941 64 253 479 370 895 678 66 484 627 220 140 57 441 329 135 931 469 250 319 498 765 964 910 37
2 34 983 692 325 584 222 908
The array sorted in the combining way is:
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 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
The array sorted in the combining way has 22034 comparisons and 4170 swaps.
It took 00:00:00.0001017 time to perform the algorithm.
```


1.5 Тестування алгоритму

«Бульбашка»



```
Program.cs
Labwork1
1 Please, enter the length of the sequence: 100
2 The sequence sorted descending:
3 100 99 98 97 96 95 94 93 92 91 90 89 88 87 86 85 84 83 82 81 80 79 78 77 76 75 74 73 72 71 70 69 68 67 66 65 64 63 62 61
4 60 59 58 57 56 55 54 53 52 51 50 49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21
5 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
6 The new sorted sequence:
7 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43
8 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
9 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
10 The bubble-sorted sequence with the next count of the comparisons is - 9801; swaps - 4950 is:
11 It takes 00:00:00.0000362 time to perform the algorithm.
12
13 C:\Users\Дима\Desktop\Studying\Labs\algorithms2term\C-sharp\Labwork 1\bin\Release\netcoreapp3.1\Labwork 1.exe (процесс 3
14 868) завершил работу с кодом 0.
15 Нажмите любую клавишу, чтобы закрыть это окно...
16
17
18
19
20
21
22
23
24
25
26
27
28
29
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
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
251
```

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 за спаданням бульбашки для 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 3.2.2 – Характеристики оцінювання алгоритму гребінця 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 за зростанням 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 бульбашки 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

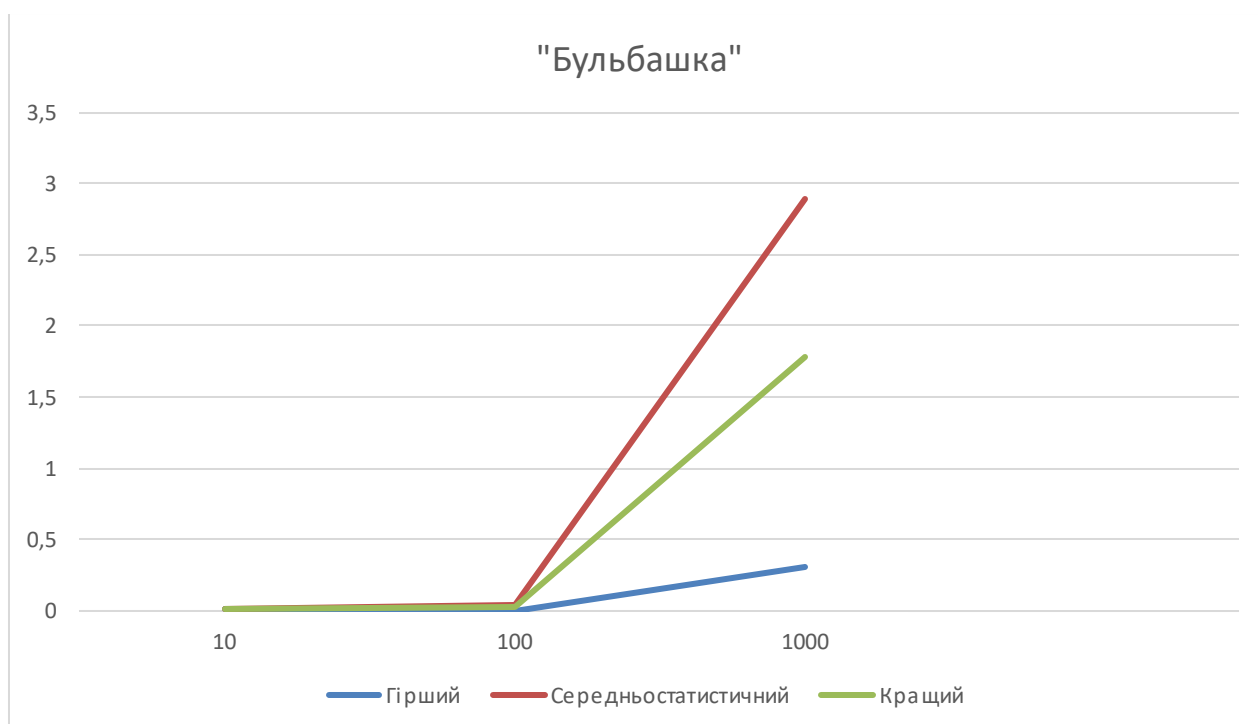
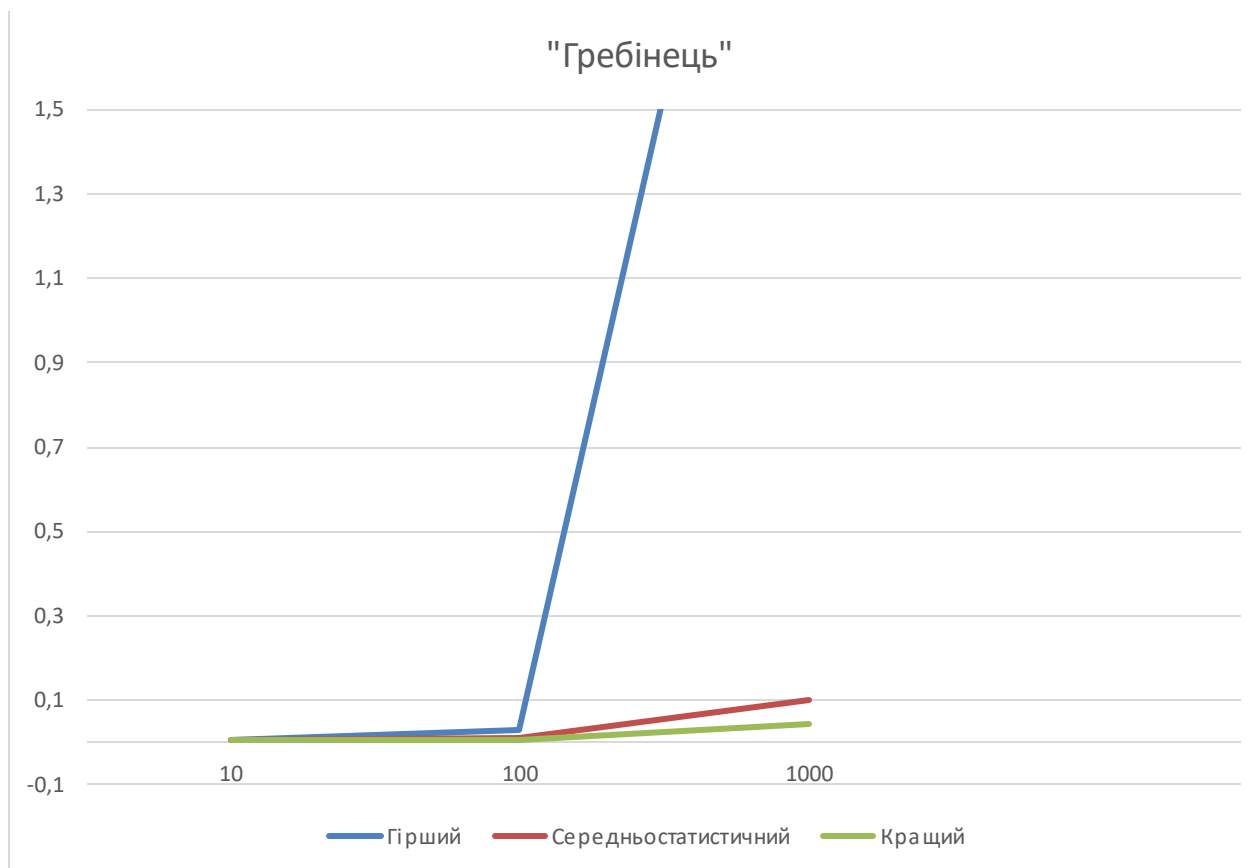
Таблиця 3.4.2 – Характеристика оцінювання алгоритму сортування для випадкової послідовності елементів у масиві.

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 (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



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 проведене тестування алгоритму «бульбашки» та «гребінець» шляхом підстановки різних значень, які представляють кількість елементів послідовності.

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%.