**Звіт до**

**лабораторної роботи №2**

**З АСД**

Павлюченко Артем

ІПЗ-11/1

# **Умови завдання:**

Написати програму мовою C# з можливістю вибору різних алгоритмів сортування. Продемонструвати роботу (ефективність, час виконання) програм на різних структурах даних (масив, лінійний зв’язаний список), з різними умовами, що забезпечують зменшення часу виконання. Навести аналіз отриманих результатів. Реалізувати алгоритми:

* Сортування вибором;
* Сортування вставкою;

# **Аналіз задачі:**

Усі алгоритми працюватимуть з цілими числами, аби краще проаналізувати результати. Для усіх видів пошуку із масивом ми використовуємо лише один клас Program, у якому масив створюється та виконуються алгоритми над ним, а вже для зв’язного списку мені довелося створити додатковий клас Node, у якому я і реалізував створення цього списку, вивід його на екран, заповнення списку, та його ініціалізацію. Та усі алгоритми пошуків у зв’язному списку ми виконуємо у класі Program.

Робота з масивом: за допомогою оператора switch користувач обирає роботу з масивом, створює його, та за допомогою новоствореного меню вибирає ті дії і алгоритми, які хоче провести над масивом.

Робота зі зв’язним списком: за допомогою оператора switch користувач обирає роботу зі зв’язним списком, створює його, та за допомогою новоствореного меню вибирає ті дії і алгоритми, які хоче провести над списком. (у обох випадках меню не відрізнятиметься)

Для повноцінної перевірки працездатності моїх алгоритмів я перевіряв масив і список різних розмірів та отримав досить інформативні результати.

# **Структура основних вхідних та вихідних даних:**

Під час виконання кожного алгоритму пошуку вхідними даними буде виступати масив цілих чисел (або лінійний список). Вихідними даними виступатиме відсортований масив або відсортований зв’язний список.

# **Алгоритм розв'язання задачі:**

**Сортування вставкою:**

Для масиву:

static void SelectionSort(int[] items)

{

Stopwatch sWatch = new Stopwatch();

sWatch.Start();

int sortedRangeEnd = 0;

while (sortedRangeEnd < items.Length)

{

int nextIndex = FindIndexOfSmallestFromIndex(items, sortedRangeEnd);

Swap(items, sortedRangeEnd, nextIndex);

sortedRangeEnd++;

}

Console.WriteLine("Your massive after sorting: ");

OutputMassive(items);

sWatch.Stop();

Console.WriteLine("Duration " + sWatch.ElapsedMilliseconds + "ms");

}

static int FindIndexOfSmallestFromIndex(int[] items, int sortedRangeEnd)

{

int currentSmallest = items[sortedRangeEnd];

int currentSmallestIndex = sortedRangeEnd;

for (int i = sortedRangeEnd+1; i < items.Length; i++)

{

if (currentSmallest.CompareTo(items[i]) > 0)

{

currentSmallest = items[i];

currentSmallestIndex = i;

}

}

return currentSmallestIndex;

}

Для списку:

static void SelectionSort1(Node node)

{

Stopwatch sWatch = new Stopwatch();

sWatch.Start();

int sortedRangeEnd = 0;

while (sortedRangeEnd < kolvo1)

{

int nextIndex = FindIndexOfSmallestFromIndex1(node, sortedRangeEnd);

Swap1(node, sortedRangeEnd, nextIndex);

sortedRangeEnd++;

}

Console.WriteLine("Your massive after sorting: ");

node.Print();

Console.WriteLine();

sWatch.Stop();

Console.WriteLine("Duration " + sWatch.ElapsedMilliseconds + "ms");

}

static int FindIndexOfSmallestFromIndex1(Node node, int sortedRangeEnd)

{

int currentSmallest = GetElement(node, sortedRangeEnd);

int currentSmallestIndex = sortedRangeEnd;

for (int i = sortedRangeEnd + 1; i < kolvo1; i++)

{

if (currentSmallest.CompareTo(GetElement(node, i)) > 0)

{

currentSmallest = GetElement(node, i);

currentSmallestIndex = i;

}

}

return currentSmallestIndex;

}

**Сортування вибором:**

Для масиву:

static void InsertionSort(int[] items)

{

Stopwatch sWatch = new Stopwatch();

sWatch.Start();

int sortedRangeEndIndex = 1;

while (sortedRangeEndIndex < items.Length)

{

if (items[sortedRangeEndIndex].CompareTo(items[sortedRangeEndIndex - 1]) < 0)

{

int insertIndex = FindInsertionIndex(items, items[sortedRangeEndIndex]);

Insert(items, insertIndex, sortedRangeEndIndex);

}

sortedRangeEndIndex++;

}

Console.WriteLine("Your massive after sorting: ");

OutputMassive(items);

sWatch.Stop();

Console.WriteLine("Duration " + sWatch.ElapsedMilliseconds + "ms");

}

static int FindInsertionIndex(int[] items, int valueToInsert)

{

for (int i = 0; i < items.Length; i++)

{

if (items[i].CompareTo(valueToInsert) > 0)

{

return i;

}

}

throw new InvalidOperationException("Index wasn`t found");

}

static void Insert(int[] itemsArray, int indexInsertingAt, int indexInsertingFrom)

{

int temp = itemsArray[indexInsertingAt];

itemsArray[indexInsertingAt] = itemsArray[indexInsertingFrom];

for (int current = indexInsertingFrom; current > indexInsertingAt; current--)

{

itemsArray[current] = itemsArray[current - 1];

}

itemsArray[indexInsertingAt + 1] = temp;

}

Для списку:

static void InsertionSort1(Node node)

{

Stopwatch sWatch = new Stopwatch();

sWatch.Start();

int sortedRangeEndIndex = 1;

while (sortedRangeEndIndex < kolvo1)

{

if (GetElement(node,sortedRangeEndIndex).CompareTo(GetElement(node, sortedRangeEndIndex-1)) < 0)

{

int insertIndex = FindInsertionIndex1(node, GetElement(node, sortedRangeEndIndex));

Insert1(node, insertIndex, sortedRangeEndIndex);

}

sortedRangeEndIndex++;

}

Console.WriteLine("Your massive after sorting: ");

node.Print();

Console.WriteLine();

sWatch.Stop();

Console.WriteLine("Duration " + sWatch.ElapsedMilliseconds + "ms");

}

static int FindInsertionIndex1(Node node, int valueToInsert)

{

for (int i = 0; i < kolvo1; i++)

{

if (GetElement(node, i).CompareTo(valueToInsert) > 0)

{

return i;

}

}

throw new InvalidOperationException("Index wasn`t found");

}

static void Insert1(Node node1, int indexInsertingAt, int indexInsertingFrom)

{

int temp = GetElement(node1, indexInsertingAt);

GetPos(node1, indexInsertingAt).data = GetElement(node1, indexInsertingFrom);

for (int current = indexInsertingFrom; current > indexInsertingAt; current--)

{

GetPos(node1, current).data = GetElement(node1, current-1);

}

GetPos(node1, indexInsertingAt+1).data = temp;

}

Допоміжні алгоритми:

Для масиву:

static void Swap(int[] items, int left, int right)

{

if (left != right)

{

int temp = items[left];

items[left] = items[right];

items[right] = temp;

}

}

Для списку:

static void Swap1(Node node, int left, int right)

{

if (left != right)

{

int temp = GetElement(node,left);

GetPos(node,left).data = GetElement(node, right);

GetPos(node, right).data = temp;

}

}

# **Код програми:**

Увесь код програми знаходиться в репозиторії GitHub:

https://github.com/Pavich123/Programming-Algorithms

# **Набір тестів:**

1. Масив на 10 елементів:

Вхідний масив: 83 68 83 55 44 12 94 47 56 72

Вихідний масив: 12 44 47 55 56 68 72 83 83 94

1. Масив на 100 елементів:

Вхідний масив: 475 708 93 44 481 501 970 266 241 569 509 384 439 551 35 250 494 645 158 813 631 130 516 856 8 309 289 466 184 368 141 814 703 526 317 897 25 5 273 35 915 123 242 136 951 432 671 600 706 232 229 729 688 399 542 346 192 236 36 172 212 505 82 873 429 695 680 912 235 138 226 490 373 123 898 509 887 380 905 575 638 689 760 952 139 411 127 305 983 970 705 788 968 102 822 410 42 369 707 257

Вихідний масив: 5 8 25 35 35 36 42 44 82 93 102 123 123 127 130 136 138 139 141 158 172 184 192 212 226 229 232 235 236 241 242 250 257 266 273 289 305 309 317 346 368 369 373 380 384 399 410 411 429 432 439 466 475 481 490 494 501 505 509 509 516 526 542 551 569 575 600 631 638 645 671 680 688 689 695 703 705 706 707 708 729 760 788 813 814 822 856 873 887 897 898 905 912 915 951 952 968 970 970 983

1. Масив на 1000 елементів:

Вхідний масив: 4856 3707 2903 4432 2581 7137 8832 8632 822 7675 990 4601 6508 6913 8048 3142 346 7748 5948 3678 1893 2137 1892 7164 3509 4936 841 7020 3092 6270 2469 9400 8970 4900 7439 1722 3171 3364 2226 398 5954 5790 9040 7013 4965 4022 8914 3751 6028 3438 2643 3086 9544 2214 3714 2720 1815 5739 924 7644 6296 1813 5540 4552 5206 1589 5631 1609 9473 6326 9971 6981 5522 5551 7723 6102 3096 4879 2199 9487 6021 7090 992 9654 3628 9382 9856 6757 1187 4720 9906 7432 2441 4581 4101 4142 250 4489 1807 3377 8391 7306 4277 9701 3467 5661 7564 3994 4490 7611 9623 6936 3541 1437 1624 9206 822 5886 924 5823 1732 8874 422 4754 6420 2539 4540 941 1450 3582 5852 8607 3072 8821 1096 8715 2813 1290 6188 7966 1819 5863 2267 3576 5097 2971 3892 1005 2958 4895 3320 4364 3565 5984 1646 9517 6884 9523 3282 928 1122 6623 2544 ……………..3567 3896 9953 9789 1107 4038 7272 9767 6506 7023 9327 4627 203 4339 2500 5854 6863 4394 1979 3895 1353 2366 736 6592 874 2052 8959 339 1393 7445 2545 3114 8836 7895 7880 7870 5410 8215 1160 1516 8684 214 1873 4360 1714 7194 9568 4821 1014 3184 1395 2385 4778 9662 5156 1202 3160

Вихідний масив: 1 39 51 53 71 90 94 156 165 176 179 199 200 203 214 230 239 250 256 262 267 271 290 326 329 339 340 341 346 357 395 398 412 416 422 432 467 478 489 520 550 588 608 619 637 646 664 665 671 687 690 692 705 710 714 736 745 753 799 816 821 822 822 822 841 867 872 874 876 886 900 908 914 924 924 928 934 935 936 941 944 969 990 992 1005 1006 1014 1016 1025 1028 1038 1042 1042 1065 1078 1096 1107 1122 1125 1160 1173 1180 1187 1191 1202 1212 1218 1220 1228 1243 …………………. 8647 8668 8684 8686 8703 8709 8715 8731 8744 8761 8766 8772 8791 8809 8821 8828 8832 8836 8840 8870 8871 8874 8878 8904 8910 8913 8914 8958 8959 8959 8970 8973 8986 9008 9015 9027 9037 9038 9040 9058 9070 9081 9087 9106 9107 9108 9125 9131 9158 9159 9172 9183 9197 9200 9206 9211 9259 9270 9285 9302 9304 9322 9325 9327 9334 9354 9370 9382 9387 9387 9393 9400 9432 9437 9447 9465 9473 9474 9485 9486 9487 9501 9502 9517 9523 9525 9529 9532 9544 9554 9568 9593 9603 9612 9623 9637 9654 9656 9662 9679 9680 9688 9697 9701 9709 9712 9724 9728 9767 9767 9770 9789 9805 9822 9826 9833 9837 9849 9850 9854 9856 9864 9866 9869 9886 9906 9931 9931 9935 9937 9938 9953 9960 9965 9971 9972 9980 9991

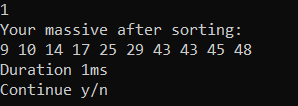
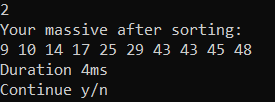
1. Масив на 100000 елементів:

Вхідний масив: 341061 921526 621355 545266 436245 105106 718203 499057 616129 97257 17590 239112 428237 990993 780924 540731 610146 460601 804462 707961 39708 210464 845522 326761 866464 351618 190024 711444 475161 702690 585472 299694 109489 111973 313474 87056 799562 170787 942837 212069 524686 636430 276307 704940 908752 985460 396982 849851 874857 620762 113889 856134 424045 …………….. 235239 544438 879540 79559 118143 808267 42224 548740 365695 843106 626306 471882 917376 278294 276737 870263 852770 50759 800093 838618 796910 306874 522176 909913 542072 306541 563593 313709 99618 390732 714540 4950 714115 477805 771765 97661 553775 603137 142432 983344 844565 258026 604639 561915 97677 551509 700079 908349 550429 285330 631686 58652 105783 808096 286750 928364 22262 969627 537487 811602 244675 728515 449122 974962 128567 621356 757766 439572 506529 179077 316488 249633 908327 816749 994052 538885 702235 960260 812236 990562

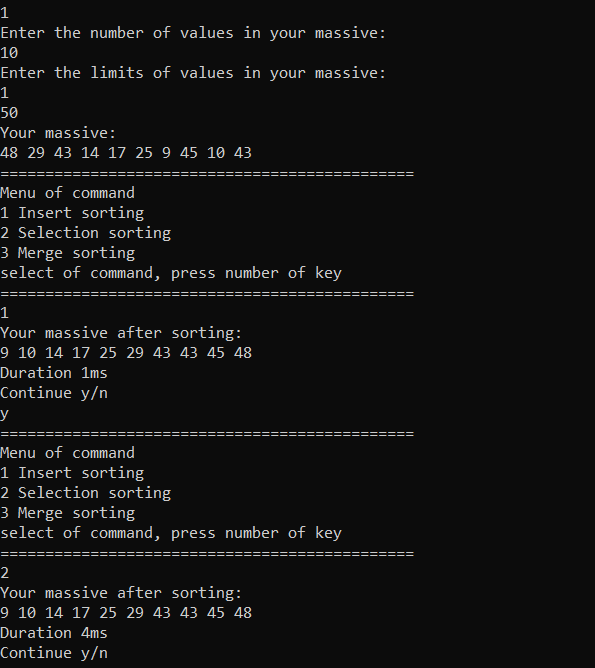
Вихідний масив: 15 17 36 36 38 41 42 54 56 66 71 73 73 87 104 106 148 162 162 163 204 209 210 213 221 230 260 269 271 285 288 302 303 306 312 317 327 359 366 379 399 400 404 421 422 435 467 469 470 477 488 516 525 536 546 553 553 592 598 599 641 644 685 718 721 722 723 730 730 739 745 747 778 786 792 803 804 804 815 827 857 866 873 883 927 945 954 965 967 969 973 996 1032 1039 1048 1054 1064 1070 1074 1078 1110 1126 1165 1170 1177 1186 1187 1203 1206 1212 1212 1214 1216 1217 1234 1245 1250 1250 1262 1266 1270 1272 1284 1292 1297 1302 1303 1310 1311 1346 1349 1350 1358 1365 1371 1413 1414 1417 1420 …………………. 999386 999396 999403 999407 999408 999432 999439 999440 999473 999495 999515 999518 999527 999527 999534 999534 999549 999550 999558 999575 999593 999627 999642 999650 999672 999685 999707 999712 999725 999729 999742 999770 999786 999787 999787 999789 999792 999817 999826 999832 999836 999856 999862 999865 999882 999887 999888 999893 999893 999929 999942 999953 999968 999976

# **Результати тестування:**

Тест №1:

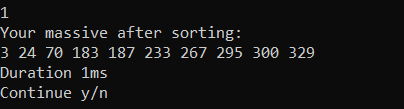
Selection Sorting: Insertion sorting:  

Selection Sorting and Insertion sorting:

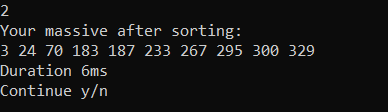


Тест №2:

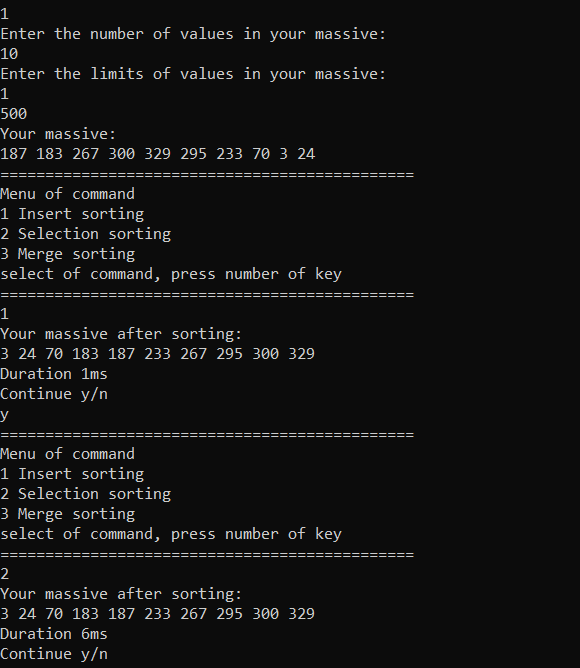
Selection Sorting:



Insertion sorting:

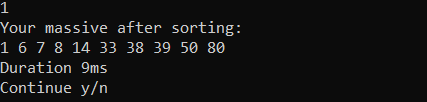


Selection Sorting and Insertion sorting:

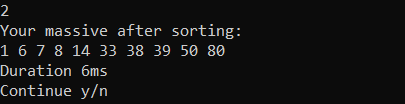


Тест №3:

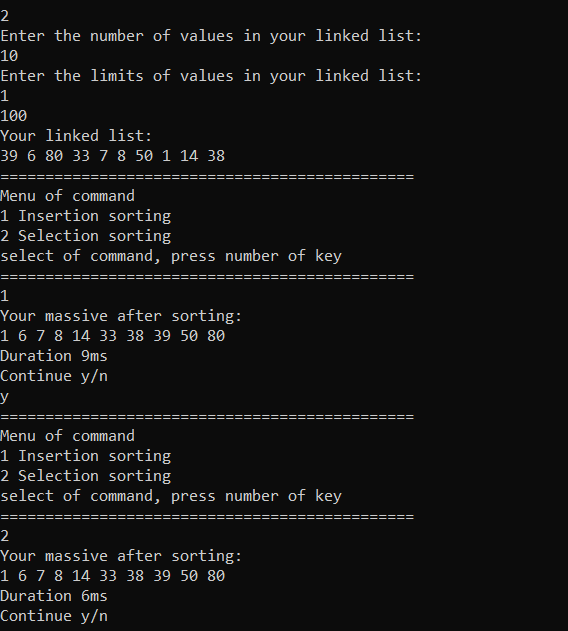
Selection Sorting:



Insertion sorting:

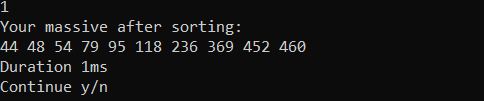


Selection Sorting and Insertion sorting:

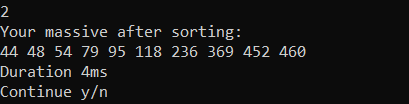


Тест №4:

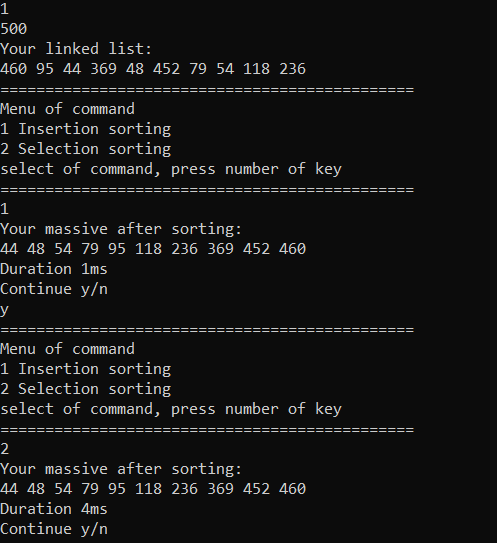
Selection Sorting:



Insertion sorting:



Selection Sorting and Insertion sorting:



У фінальному варіанті програми не виявлено жодних помилок під час тестувань, отже усі алгоритми працюють правильно, видають очікувані результати з приводу часу та ефективності.

Майже у всіх випадках алгоритм сортування вибором був значно ефективніше і швидше за алгоритм сортування вибором як при роботі з масивом, так і при роботі зі зв’язним списком, тож на мою думку, цей алгоритм має використатися під час подібних обрахунків.