|  |  |  |  |
| --- | --- | --- | --- |
| МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені Тараса Шевченка  ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ  **Кафедра програмних систем і технологій**  Дисципліна  **«Структури даних, аналіз і алгоритми комп’ютерної обробки інформації»**  **Лабораторна робота № 1**  **«Алгоритми пошуку»** | | | |
| **Виконав:** | Литвинчук Владислав Валерійович | **Перевірив**: | Бичков Олексій Сергійович |
| Група | ІПЗ-11(1) | Дата перевірки |  |
| Форма навчання | денна | Оцінка |  |
| Спеціальність | 121 |
| 2022 | | | |

# **Умова задачі**

Написати програму мовою C# з можливістю вибору різних алгоритмів пошуку. 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:

1. пошуку перебором елемента масиву, що дорівнює заданому значенню.
2. пошуку з бар'єром елемента масиву, що дорівнює заданому значенню.
3. бінарного пошуку елемента масиву рівного заданому значенню.
4. бінарного пошуку елемента масиву, рівного заданому значенню, в якій нове значення індексу m визначалося б не як середнє значення між L і R, а згідно з правилом золотого перерізу.

# **Аналіз задачі**

Створимо глобальний масив і лінійно-зв’язний список, які будемо передавати у функції пошуку. Спочатку розробляємо алгоритми для пошуку у звичайному статичному масиві. Надалі слід врахувати особливості лінійного списку і переписати алгоритми під нього. Будемо використовувати «перевантаження» функцій, зберігаючи назви методів.

Для швидкого створення великих масивів і списків напишемо метод, який на основі заданих діапазонів випадковим чином згенерує елементи відповідних структур даних. Створення списку розпочинається з того, що необхідно створити відповідний масив елементів і за допомогою алгоритму заповнити лінійно-зв’язний список.

Для списку буде створено клас, в якому визначені його відкриті поля і методи створення.

Програма має складатись з декількох методів, що визначають конкретний алгоритм пошуку. Для зручності і можливості за один раз переглянути усі алгоритми вводимо метод Меню, що буде реалізовано за допомогою оператора switch.

Виклик метода супроводжується введенням ключа (елемент, що необхідно знайти) і надалі спрацьовує сам алгоритм пошуку в залежності від структури даних (масив або список).

# **Структура основних вхідних та вихідних даних**

Вхідними даними для методів буде створений глобальний масив і лінійно-зв’язний список, в які занесено певні числа. Також у самих методах визначено ключ – елемент, що необхідно знайти за результатами алгоритму.

Вихідні дані представляють собою порядковий номер елемента, який було знайдено і час роботи алгоритму (від початку пошуку до виведення повідомлення), або ж рядок слів, в якому написано про те, що елемент не було знайдено.

# **Алгоритм розв’язання задачі**

## **Лінійний пошук**

### **У масиві:**

int index = 0, key;

Found = false;

while (index < Array.Length && !Found)

{

if (Array[index] == key)

{

Found = true;

}

index++;

}

### **У списку:**

int index = 0, key;

bool Found = false;

while (head != null && !Found)

{

If (head.Data == key)

{

Found = true;

}

index++;

head = head.Next;

}

## **Пошук з Барьером**

### **У масиві:**

int index = 0, key;

Array.Resize(ref nArray, ArrayLength + 1);

nArray[ArrayLength] = key;

while (nArray[index] != key)

{

index++;

}

Array.Resize(ref nArray, ArrayLength);

### **У списку:**

int index = 0, key;

LinkedList newNode = new LinkedList(key, null);

LinkedList temp = head;

while(temp.Next != null)

{

temp = temp.Next;

}

temp.Next = newNode;

temp = head;

while (temp.Data != key)

{

temp = temp.Next;

index++;

}

while (head.Next.Next != null)

{

head = head.Next;

}

head.Next = null;

## **Бінарний пошук**

### **У масиві:**

double goldenration = (Math.Sqrt(5) + 1) / 2;

double lambda = 0;

if (isGoldenRation)

{

lambda = goldenration;

}

else

{

lambda = 1;

}

int key;

int Left = 0, Right = Array.Length - 1;

int Middle = 0;

while (Left < Right)

{

Middle = (int)((Left + lambda\*Right) / (1+lambda));

if (key > Array[Middle])

{

Left = Middle + 1;

}

else

{

Right = Middle;

}

}

### **У списку:**

double goldenration = (Math.Sqrt(5) + 1) / 2;

double lambda = 0;

if (isGoldenRation)

{

lambda = goldenration;

}

else

{

lambda = 1;

}

int indexLeft = 0, indexRight = LinkedList.GetLength(head), indexMiddle = 0;

LinkedList nodeLeft = head, nodeRight = null, nodeMiddle = null;

while (nodeRight == null || nodeLeft.Data != nodeRight.Data)

{

indexMiddle = (int)((indexLeft + lambda \* indexRight) / (1 + lambda));

nodeMiddle = LinkedList.GetElement(nodeLeft, indexMiddle - indexLeft);

if (key > nodeMiddle.Data)

{

indexLeft = indexMiddle + 1;

nodeLeft = nodeMiddle.Next;

}

else

{

indexRight = indexMiddle;

nodeRight = nodeMiddle;

}

}

**Допоміжні методи:**

public static LinkedList GetElement(LinkedList head, int iterations)

{

int index = 0;

for (index = 0; index < iterations; index++)

{

head = head.Next;

}

return head;

}

public static int GetLength(LinkedList head)

{

int counter = 0;

while(head.Next != null)

{

counter++;

head = head.Next;

}

return counter;

}

# **Код програми**

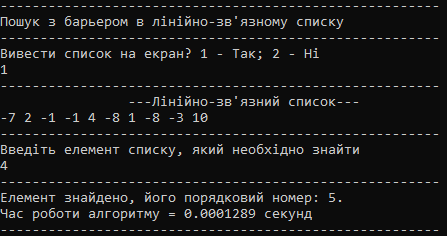
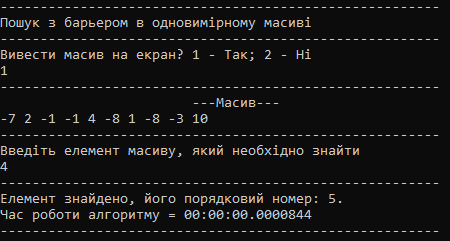
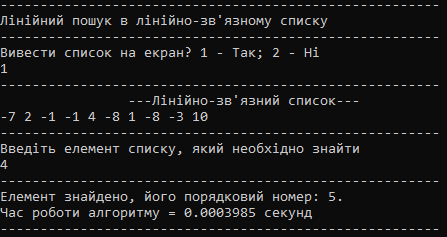
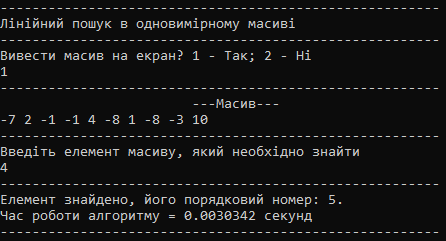
Код програми було завантажено до репозиторію на GitHub: <https://github.com/Shadeonskiy/ASD_OOP_Labs/tree/main/ASD>

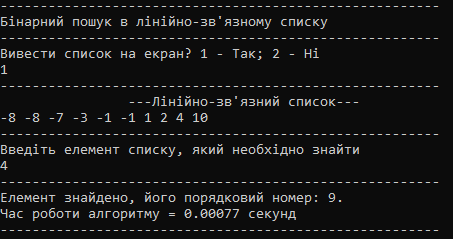
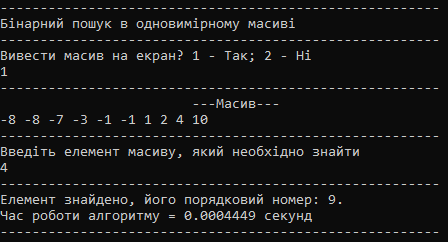
# **Набір тестів**

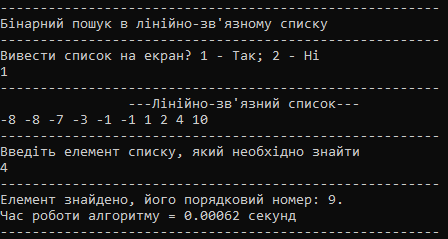
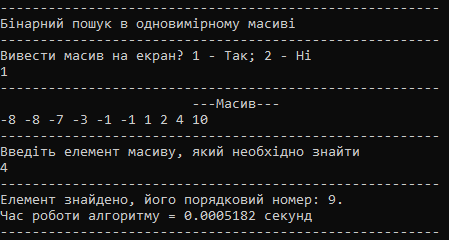
1. **Ключ:** 4 з порядковим номером 5 (9 для відсортованого);  
   **Діапазон значень для масиву:** від -10 до 10;   
   **Масив з 10 елементів:** -7 2 -1 -1 **4** -8 1 -8 -3 10
2. **Ключ:** -100, відсутній у масиві; **Діапазон значень для масиву:** від -100 до 100;   
   **Масив з 100 елементів:** 62 10 90 72 92 51 -5 71 -55 50 -2 16 -93 66 -94 -29 -59 71 82 41 91 58 -37 19 81 100 -69 4 88 -70 4 15 -79 76 -62 58 -48 -40 39 38 32 39 -87 -33 99 -78 -2 -62 11 59 94 25 -96 13 53 -96 -53 -30 92 93 20 92 83 -86 -55 84 -6 -68 28 -52 -82 82 -68 -57 -92 -48 44 97 21 58 1 93 94 -71 -63 80 10 9 -78 -67 11 83 -32 46 -82 41 56 100 -79 -86
3. **Ключ:** 571 з порядковим номером 542 (791 для відсортованого); **Діапазон значень для масиву:** від -1000 до 1000;   
   **Масив з 1000 елементів:** 815 -241 231 -627 -974 17 -109 239 -875 862 609 -17 779 927 977 -746 -17 -5 -928 -704 582 736 731 -614 542 -753 858 -753 -604 -40 152 226 -588 859 840 328 462 512 -155 648 765 39 -853 473 384 -314 -799 -34 -854 -448 507 -104 -138 16 -299 -922 28 -155 -168 779 159 -356 -158 166 -290 -617 -429 921 -913 -352 -208 471 -850 -575 -469 -457 589 -743 2 -145 -954 -108 -900 845 453 -745 -637 396 157 761 -701 -383 -320 66 -512 120 -803 -18 -238 1 -885 -720 -122 498 761 -965 -254 -564 -516 -842 -510 -229 843 977 732 -733 -457 -3 713 -545 -981 175 -236 -673 -651 -825 -209 84 938 412 -655 -394 495 -1000 -261 767 -987 -397 -916 417 510 928 -88 -2 270 529 -226 -297 334 -780 -424 200 269 -693 -18 -59 517 -449 148 585 245 662 -502 73 813 885 276 842 238 966 -746 941 -87 -704 -55 -909 -737 766 57 -179 401 -912 750 718 -164 146 338 187 19 799 -750 462 455 -501 -828 848 429 839 186 386 -748 -67 466 369 966 -364 -714 -27 362 -110 -322 751 494 -673 -816 156 912 -219 -763 -333 -454 -911 -177 439 716 -208 -515 -586 -877 97 -337 -576 -420 672 -431 -532 -377 -619 752 -801 -141 -636 825 -871 120 -501 969 128 -686 16 936 -352 602 -481 -161 -837 -891 -974 654 174 -849 873 -151 -735 -774 -746 171 823 759 717 -467 531 29 -963 808 -818 -736 -307 -681 216 -176 357 -901 108 162 -985 -178 -939 -168 -594 -642 -404 -272 -422 -952 -13 515 560 903 -134 329 -855 369 -403 -518 -595 619 564 -289 -343 899 417 708 -563 -651 -206 773 742 103 -789 432 109 -9 353 359 937 -197 -550 990 -179 -333 -296 -210 453 887 -321 988 503 -375 -243 -604 258 350 -826 -493 -59 -111 -710 -771 254 214 774 -544 692 435 -781 155 -984 239 -455 602 168 575 891 991 195 -374 -746 -450 671 -194 -730 117 -654 815 -827 760 -184 -154 -583 47 -487 -779 -264 607 -106 -753 99 -983 667 525 972 -737 -921 155 229 684 459 -816 312 -685 -366 740 -442 -553 -56 -344 -890 878 261 -540 -662 171 -179 498 88 -54 -161 -717 97 -53 528 -763 -117 -854 -165 -994 37 192 -415 -510 -700 -369 534 -899 -587 -227 664 -712 -337 -763 -863 221 -243 207 186 -200 442 -419 -934 -718 -155 175 88 -635 -212 -603 -144 -588 613 -63 619 -82 292 -229 -841 708 638 -281 85 -686 283 -450 621 -141 -190 -694 480 677 -40 -966 -435 -544 125 -697 645 125 543 169 314 -985 918 251 -324 726 -382 219 930 986 47 479 163 -278 496 753 -1 -706 555 -771 555 -519 -83 180 453 578 -628 163 794 -11 126 34 -836 -531 406 70 -603 -968 875 895 -760 591 -623 495 -371 482 -128 843 -40 373 304 351 **571** 314 614 -168 98 -263 799 99 354 456 425 -59 -140 -987 314 -307 317 -574 355 -33 -580 660 989 964 -896 423 -449 971 -295 -470 -797 -579 685 776 464 -150 -144 -427 973 660 -766 -225 -459 -215 -834 436 -365 -384 -833 -821 -475 -466 -983 -663 -70 911 325 650 -273 675 -814 828 -606 -176 252 3 257 84 -451 -537 838 -257 -547 695 -268 646 119 204 798 -331 -212 935 803 -475 -996 670 -597 347 -154 553 525 206 847 -703 474 -594 -619 717 533 163 -821 551 -934 705 -658 783 -920 -677 606 285 -208 -879 853 -942 -113 -749 -975 869 -180 583 -401 910 -762 -3 -62 -369 -103 -844 -780 -674 264 402 672 -365 -511 238 869 -902 -817 -779 589 -920 742 561 -239 404 354 788 411 222 -619 -487 -103 950 -437 -731 312 70 -234 711 -115 925 -898 279 -980 390 -550 217 570 649 -617 928 685 -402 994 -481 -832 -322 -764 535 278 110 -255 -1 946 -250 -495 722 -929 -781 927 -202 333 -527 336 665 978 -538 541 371 -46 -864 -783 762 -161 -547 -172 -701 956 -956 -900 -366 -166 232 -568 776 179 104 334 270 -115 -272 146 350 723 585 -405 212 262 -671 -459 707 137 -135 -420 141 -108 -16 -841 492 922 -794 767 -973 -882 293 965 -900 -960 561 -514 -126 128 -730 -515 -774 -9 -132 -961 255 859 617 37 627 503 -606 497 969 -349 292 -116 -805 -56 843 349 -321 -39 118 231 -571 621 -506 171 702 843 667 -422 -133 -620 -647 -850 -976 -56 -95 633 -695 169 -861 -476 725 797 996 450 -251 -323 -709 930 286 -78 806 265 873 637 610 634 365 -554 336 784 836 816 -371 589 -703 141 -140 251 -961 -791 43 511 -178 -406 507 -367 626 45 -579 25 552 738 293 946 -426 89 -723 -612 335 271 -733 859 833 -644 -312 910 167 -594 152 608 817 40 891 77 879 -705 -557 279 -870 -24 -992 -680 -976 -689 -769 -902 -478 -317 15 209 -137 -528 917 -232 942 185 678 -535 906 597 466 219 894 -186 208 -488 848 -784 -933 -722 -354 -1 -742 -642 263 948 835 522 -876 -410 388 -516 392 943 -256 345 123 122 135 -307 -383 978 373 -467 510 -211 -573 211 104 720 -749 -694 -385 -264 -437 865 -786 559 506 995 350 -844 -659 316 -791 -18 828 951 -931 863 389 -275 81 374 662 -332 627 725 -152 -527

# **Результати тестування**

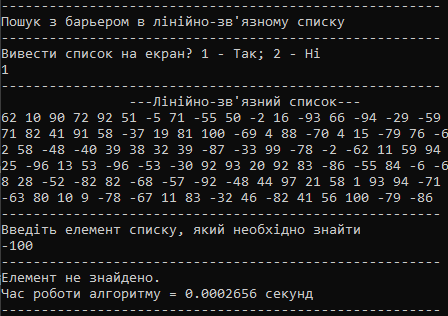
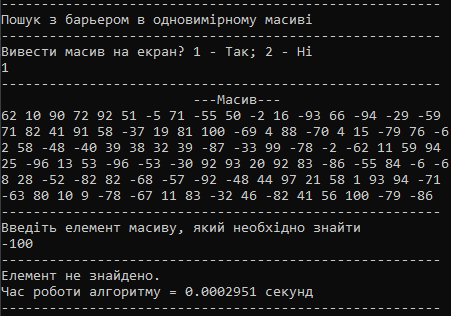
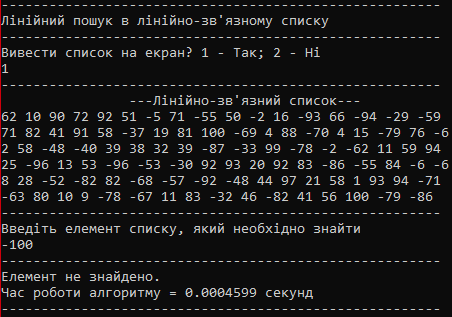
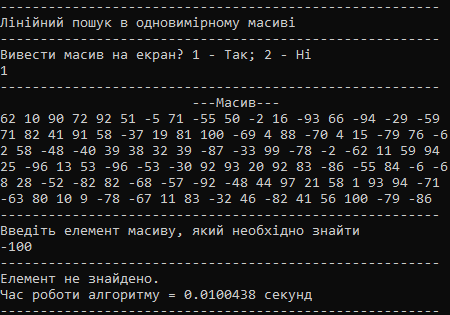
## **Тест 1:**



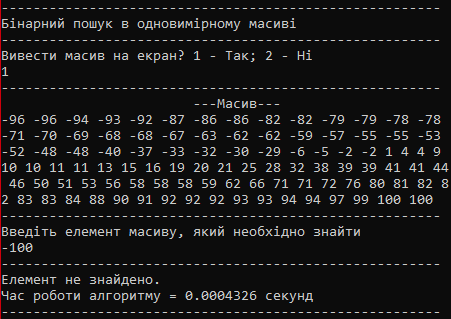
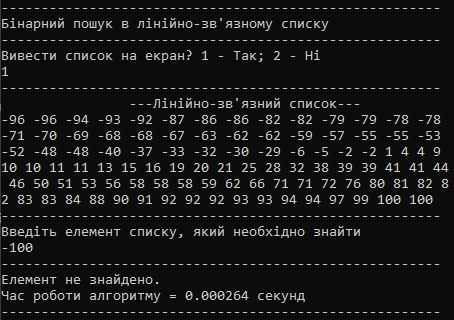
Звичайний бінарний пошук:

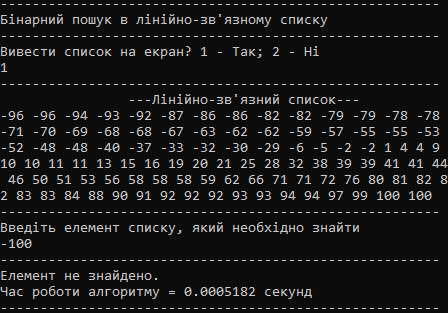
Бінарний пошук методом золотого перетину:

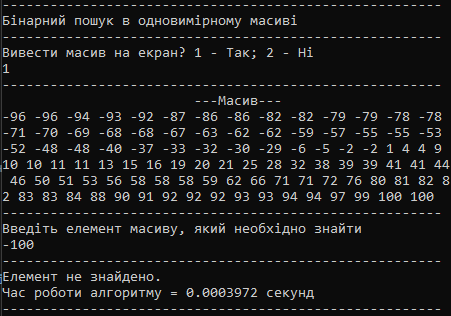
## **Тест 2:**



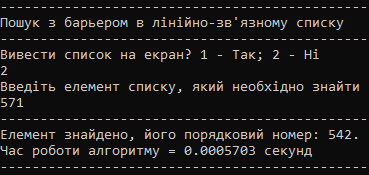
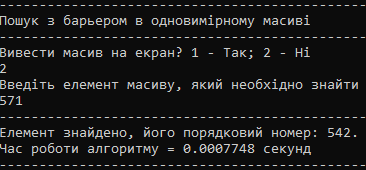
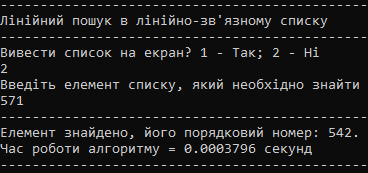
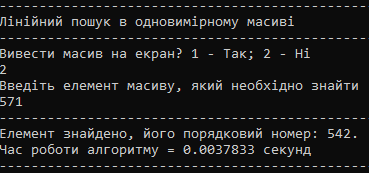
Звичайний бінарний пошук:



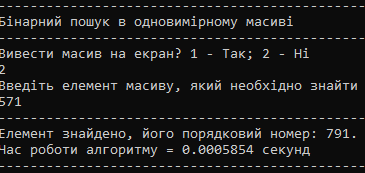
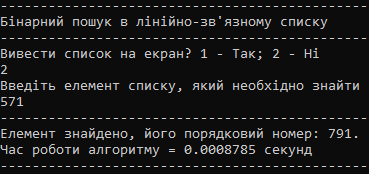
Бінарний пошук методом золотого перетину:



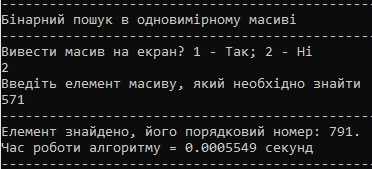
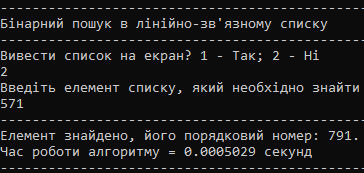
## **Тест 3:**



Звичайний бінарний пошук:



Бінарний пошук методом золотого перетину:



# **Аналіз результатів**

З результатів роботи програми можемо побачити, що помилок немає, адже усі тести виконані успішно. Під час розробки застосунку зіштовхнувся з проблемою написання алгоритму для бінарного пошуку з золотим перетином, яка виникала через недостатнє розуміння золотого перетину і формули ділення відрізка у заданому відношені.

Цю помилку добре видно з формули, адже був забутий множник для знаменнику, через що рішення могло зациклитись на одному місці – не виконується вихід з циклу:

де – відношення ділення відрізка.

# **Тестування**

Проведемо тестування на масиві і лінійному списку розміром у **1000** елементів, що буде згенеровано випадковим чином:

## **Лінійний пошук:**

### **Масив:**

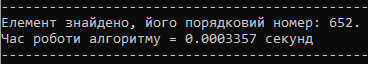


### **Список:**



## **Пошук з бар’єром:**

### **Масив:**



### **Список:**



## **Звичайний бінарний пошук:**

### **Масив:**



### **Список:**



## **Бінарний пошук методом золотого перетину:**

### **Масив:**



### **Список:**



Проведемо тестування на масиві і лінійному списку розміром у **1000000** елементів, що буде згенеровано випадковим чином:

## **Лінійний пошук:**

### **Масив:**



### **Список:**



## **Пошук з бар’єром:**

### **Масив:**

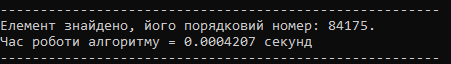


### **Список:**



## **Звичайний бінарний пошук:**

### **Масив:**

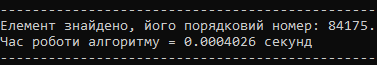


### **Список:**



## **Бінарний пошук методом золотого перетину:**

### **Масив:**



### **Список:**



# **Висновок**

Виконано завдання першої лабораторної роботи. Реалізовані всі поставлені в умові алгоритми пошуку на різних структурах даних (масив, лінійно-зв’язний список). Для звичайного масиву найбільш ефективним виявився бінарний пошук методом золотого перетину, а для лінійно-зв’язного списку – пошук з бар’єром.