Дисципліна

**«Структури даних, аналіз і алгоритми комп'ютерної обробки інформації»**

**Лабораторна робота № 1**

**«Алгоритми пошуку»**

Виконала:

Травіна Анастасія

ІПЗ-11-2

**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:

* пошуку перебором елемента масиву, що дорівнює заданому значенню.
* пошуку з бар'єром елемента масиву, що дорівнює заданому значенню.
* бінарного пошуку елемента масиву рівного заданому значенню.
* 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**

Для виконання завдання з консолі вводитиметься масив, з якого буде створено зв’язний список. За допомогою конструкції switch будемо обирати спосіб пошуку елемента, також додамо можливість генераціїї випадкових чисел. Кожен алгоритм пошуку реалізовано окремим статичним методом.

Для розрахунку ефективності кожного способу пошуку, використаємо структуру DateTime, що дозволяє обчислити час виконання алгоритму. Для зручності роботи будемо використовувати цілочисельний тип даних. Кожен вид пошуку продитиметься і для масиву, і для зв’язного списку для порівняння ефективності.

Також додамо метод зі швидким сортуванням, що знадобиться для виконання бінарного пошуку.

**Структура основних вхідних та вихідних даних**

Вхідними даними є введений з консолі масив та елемент, який потрібно знайти. Вихідними даними будуть індекс знайденого елемента при його наявності у масиві, або повідомлення про відсутність та час, затрачений на роботу алгоритму.

**Алгоритм розв’язання задачі**

Алгоритм лінійного пошуку: послідовно переглядаємо кожен елемент масиву, починаючи з нульового індексу, та порівнюємо значення з шуканим елементом. Пошук відбувається до моменту знайдення збігу, або ж до досягнення кінця масиву.

Реалізація для массиву:

int key;

int i = 0;

bool Found = false;

while ( i < MainArray.Length && !Found)

{

if (MainArray[i] == key)

{

Found = true;

}

i++;

}

Реалізація для лінійного списку:

int key;

bool Found = false;

int i = 0;

while (head != null && !Found)

{

if (head.Data == key)

{

Found = true;

}

i++;

head = head.Next;

}

Алгоритм лінійного пошуку з бар’єром: у кінець масиву додаємо елемент, який є шуканим значенням. Це дає можливість спростити умови циклу, адже співпадіння відбудеться у будь-якому разі. Послідовно переглядаємо масив, поки потрібне значення не буде знайдене.

Реалізація для массиву:

Array.Resize(ref Array, Length + 1);

Array[Length]=key;

int i = 0;

while (Array[i] != key)

{

i++;

}

Реалізація для лінійного списку:

int key;

LinkedList newNode = new LinkedList(key, null);

LinkedList temp = head;

while (temp.Next != null)

{

temp = temp.Next;

}

temp.Next = newNode;

temp = head;

int i = 0;

while (temp.Data != key)

{

temp = temp.Next;

i++;

}

Алгоритм бінарного пошуку: пошук проводиться на відсортованому масиві. Знаходиться опорний елемент посередині масива. Якщо шукане значення більше за цей елемент, то пошук проводитиметься у праві частині масиву: індекс початку перейде на індекс середини + 1, а якщо менше, то у лівій: індекс кінця перейде на індекс середнього елементу. Пошук відбуватиметься доти, доки індекс лівого елемента менше за індекс правого.

Реалізація для масиву:

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

int Mid = 0;

while (Left < Right)

{

Mid = (Left + Right) / 2;

if (key > MainArray[Mid])

{

Left = Mid + 1;

}

else

{

Right = Mid;

}

}

Реалізація для лінійного списку:

int key;

int Left = 0;

int Right = LinkedList.GetLength(head);

int Mid = 0;

LinkedList nodeLeft = head, nodeRight = null, nodeMiddle;

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

{

Mid = (Left + Right) / 2;

nodeMiddle = LinkedList.GetElement(nodeLeft, Mid - Left);

if (key > nodeMiddle.Data)

{

Left = Mid+ 1;

nodeLeft = nodeMiddle.Next;

}

else

{

Right = Mid;

nodeRight = nodeMiddle;

}

}

Алгоритм бінарного пошуку згідно з правилом золотого перерізу: проводиться як звичаний бінарний пошук, але опорний елемент обчислюється за формулою x=(x1+ **λ**\*x2)/(1+ **λ**), де **λ** = (√5+1)/2.

Реалізація для масиву:

int Left = 0, Right = MainArray.Length – 1, Mid = 0;

bool Found = false;

while (Left <= Right && !Found)

{

Mid = (int)((Left + goldenration \* Right) / (1 + goldenration));

if (key == MainArray[Mid])

{

Found = true;

}

else if (key > MainArray[Mid])

{

Left = Mid + 1;

}

else

{

Right = Mid - 1;

}

}

Реалізація для лінійного списку:

int key;

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

int Left = 0;

int Right = LinkedList.GetLength(head);

LinkedList nodeLeft = head, nodeRight = null, nodeMiddle;

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

{

int Mid = (int)((Left + goldenration \* Right) / (1 + goldenration));

nodeMiddle = LinkedList.GetElement(nodeLeft, Mid - Left);

if (key > nodeMiddle.Data)

{

Left = Mid + 1;

nodeLeft = nodeMiddle.Next;

}

else

{

Right = Mid;

nodeRight = nodeMid;

}

}

**Текст програми**

Повний код програми знаходиться на гітхабі за посиланням:

**Набір тестів**

Тест 1.

Масив з 20 елементів: 10 -5 33 25 0 7 45 -33 0 1 12 -17 20 33 29 -10 7 44 23 11

Ключ: 29

Індекс: 14 - для лінійного пошуку, 15 – для бінарного.

Тест 2.

Масив з 35 елементів: 10 56 -1 22 -5 33 45 25 0 7 8 45 -33 11 0 1 12 -5 -22 -17 20 31 33 29 50 61 -10 7 27 44 40 23 11 30 0

Ключ: 61

Індекс: 25 - для лінійного пошуку, 34 – для бінарного.

Тест 3.

Масив з 25 елементів: 70 51 10 25 0 2 7 45 -33 0 1 12 -17 20 33 -29 -10 13 44 32 18 33 50 -9 -41

Ключ: 0

Індекс: 4 - для лінійного пошуку, 6 – для бінарного.

Тест 4.

Масив з 20 елементів: 10 -5 33 25 0 7 45 -33 0 1 12 -17 20 33 29 -10 7 44 23 11

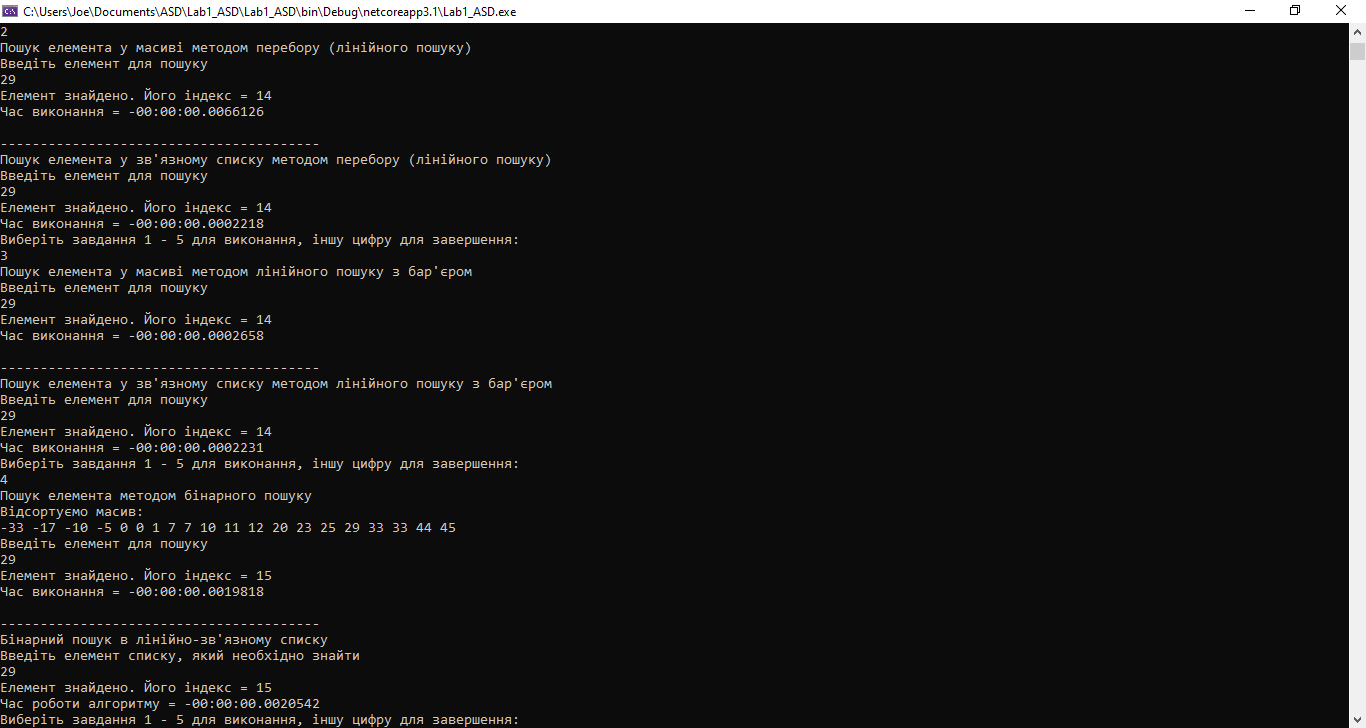
Ключ: 16

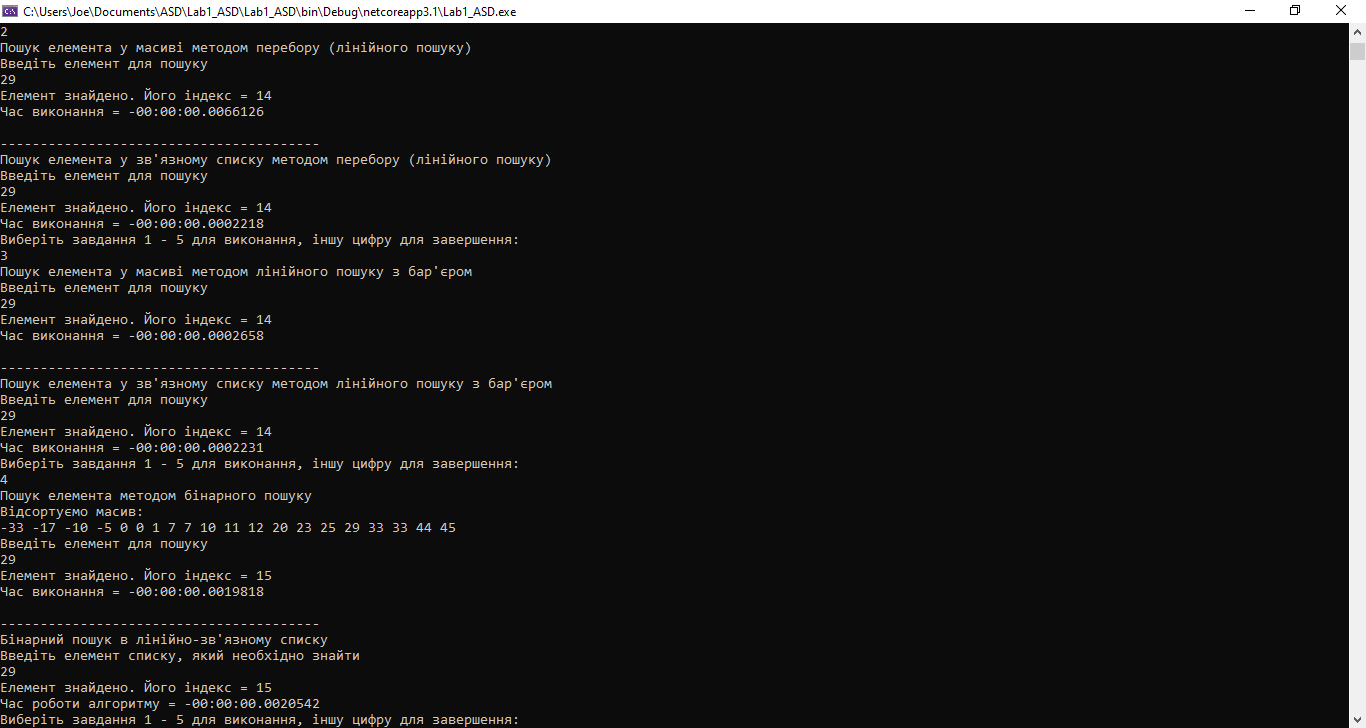
Елемент відсутній.

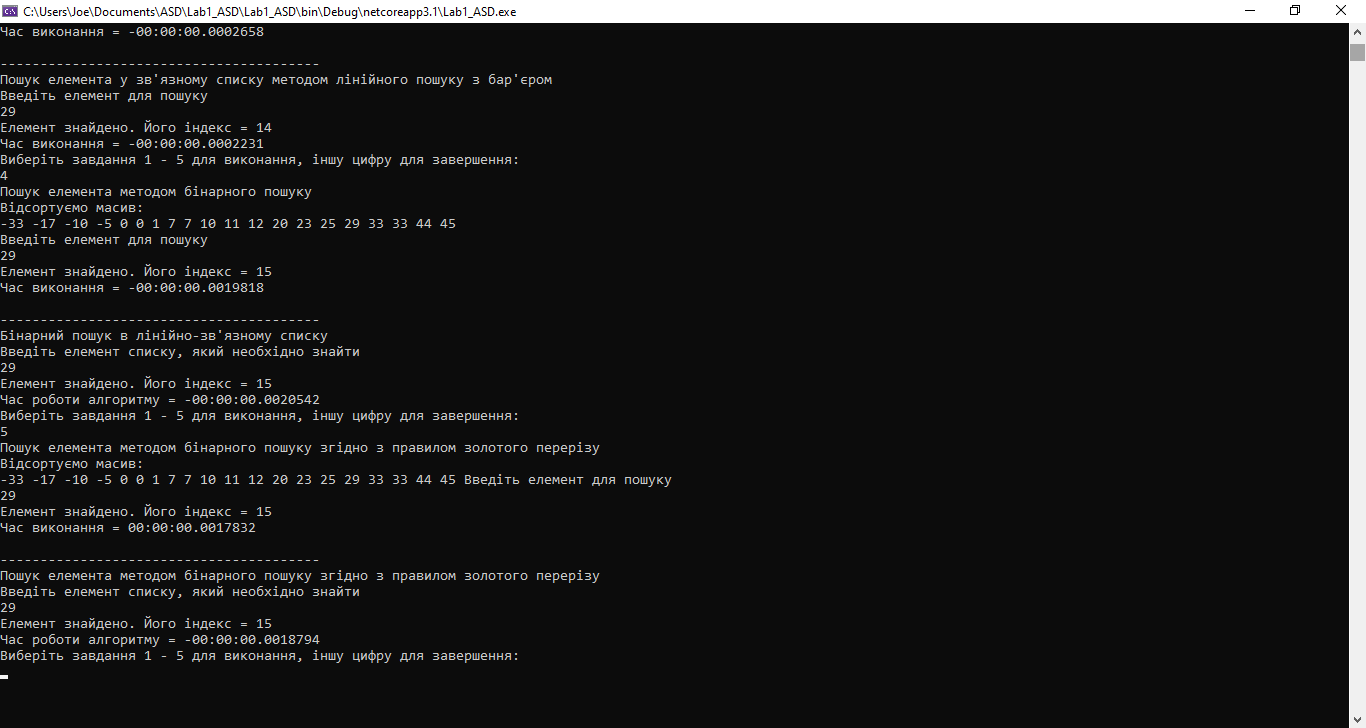
**Результати тестування програми та**

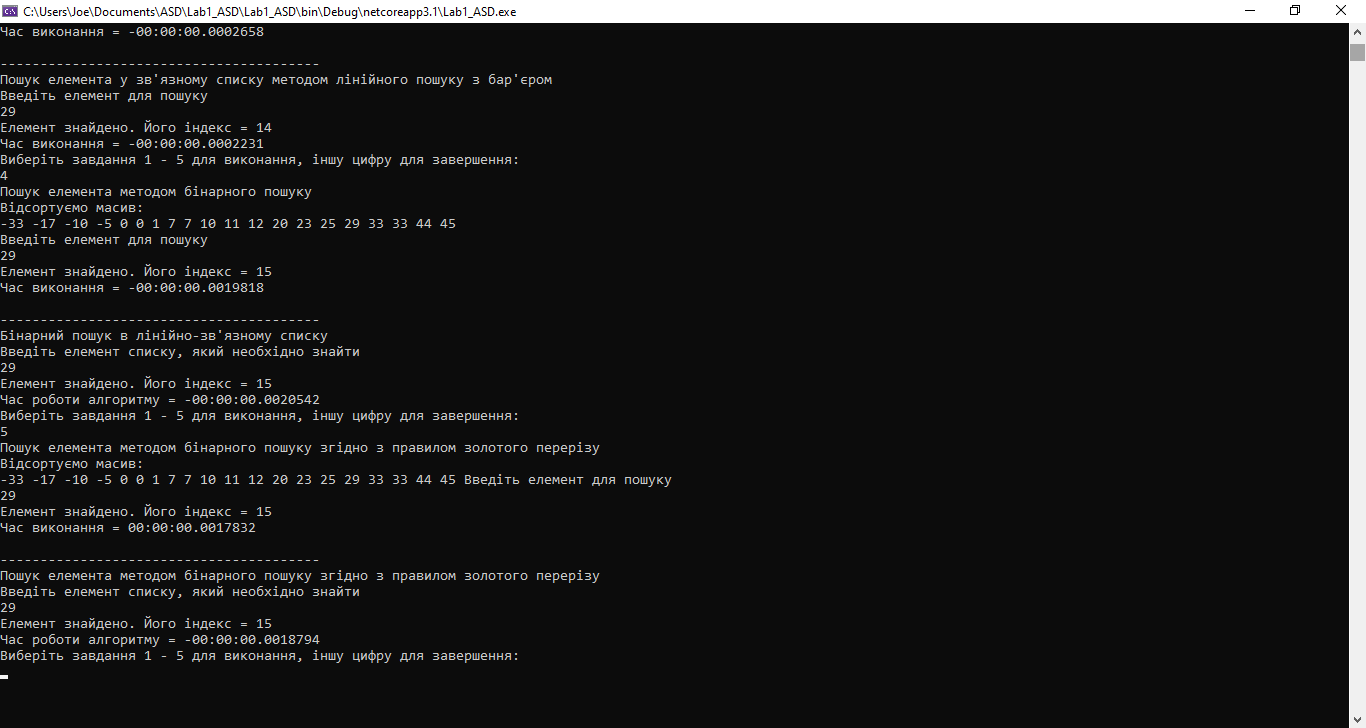
**аналіз отриманих помилок**

Тест 1

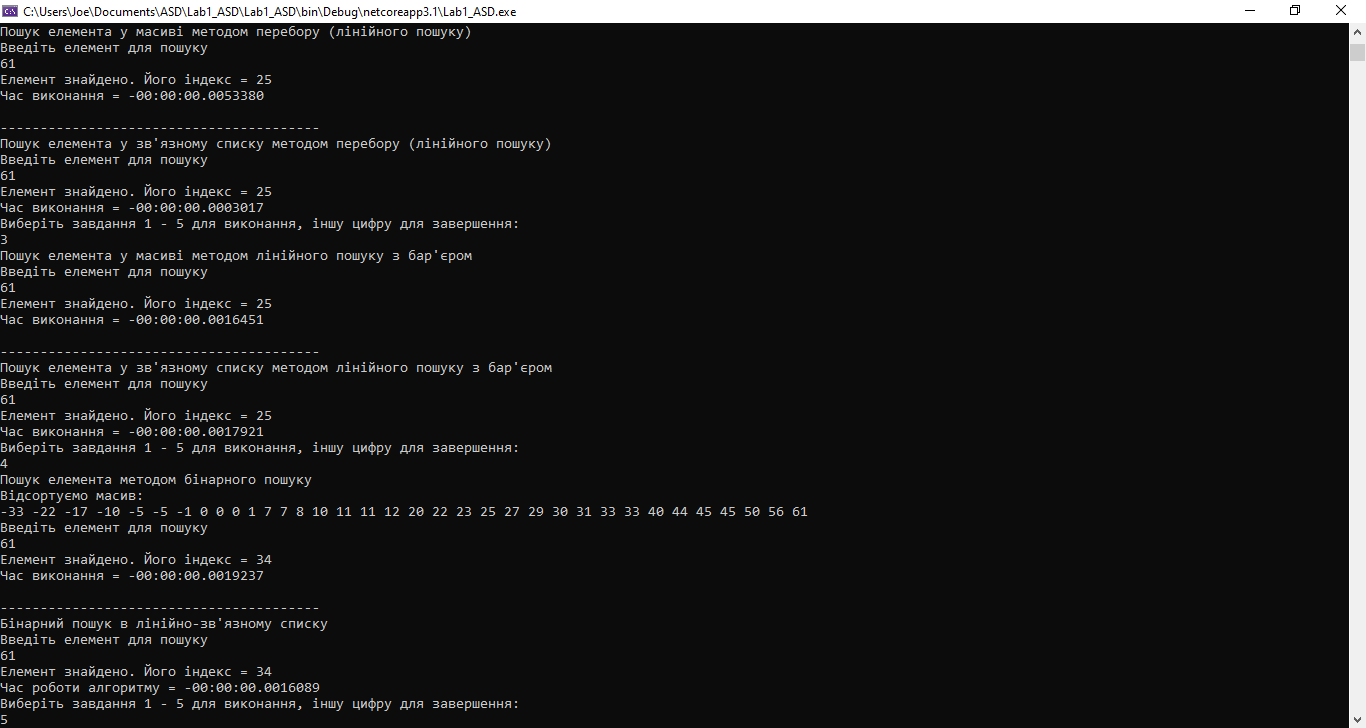


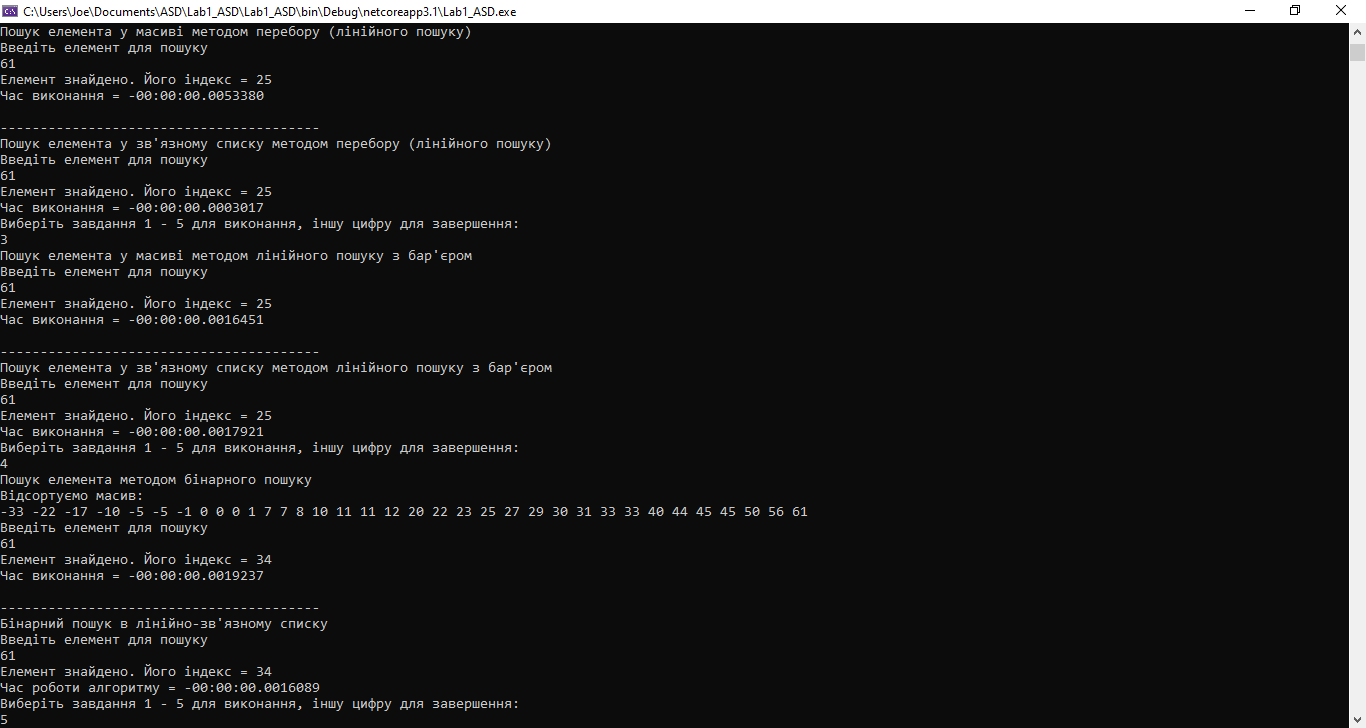


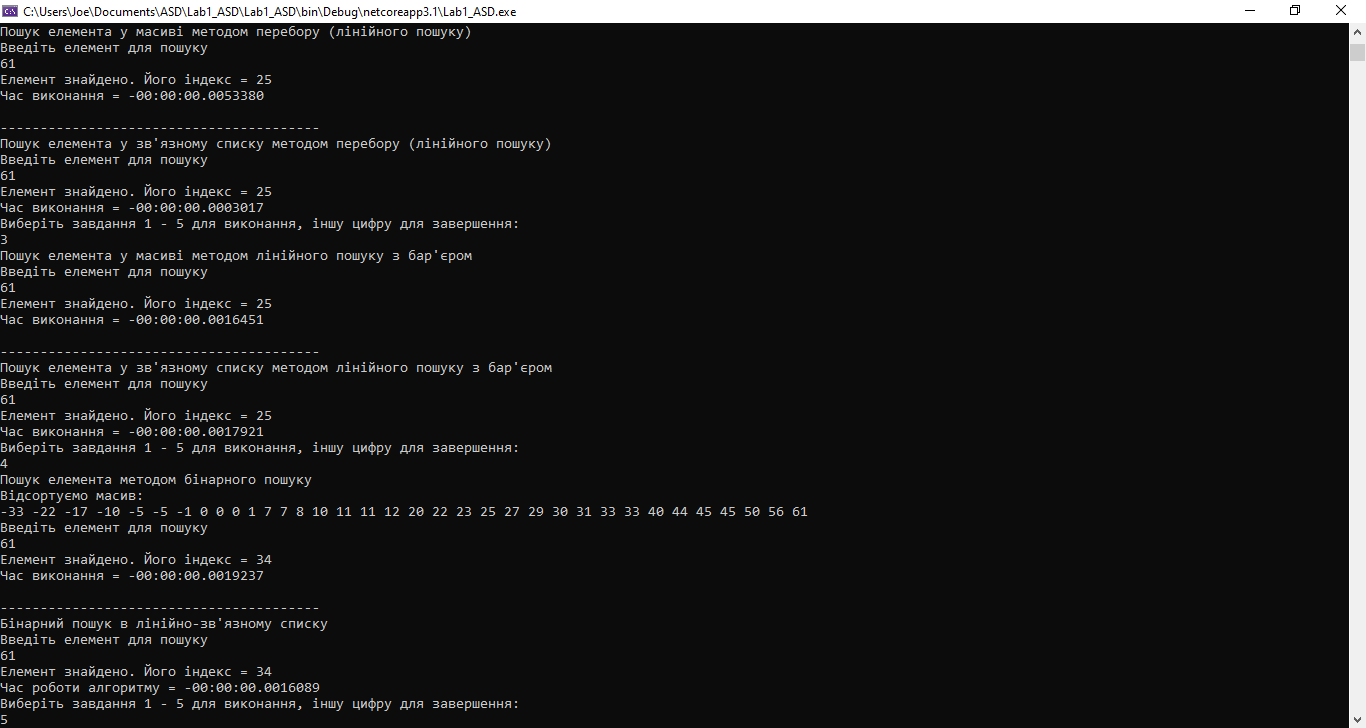


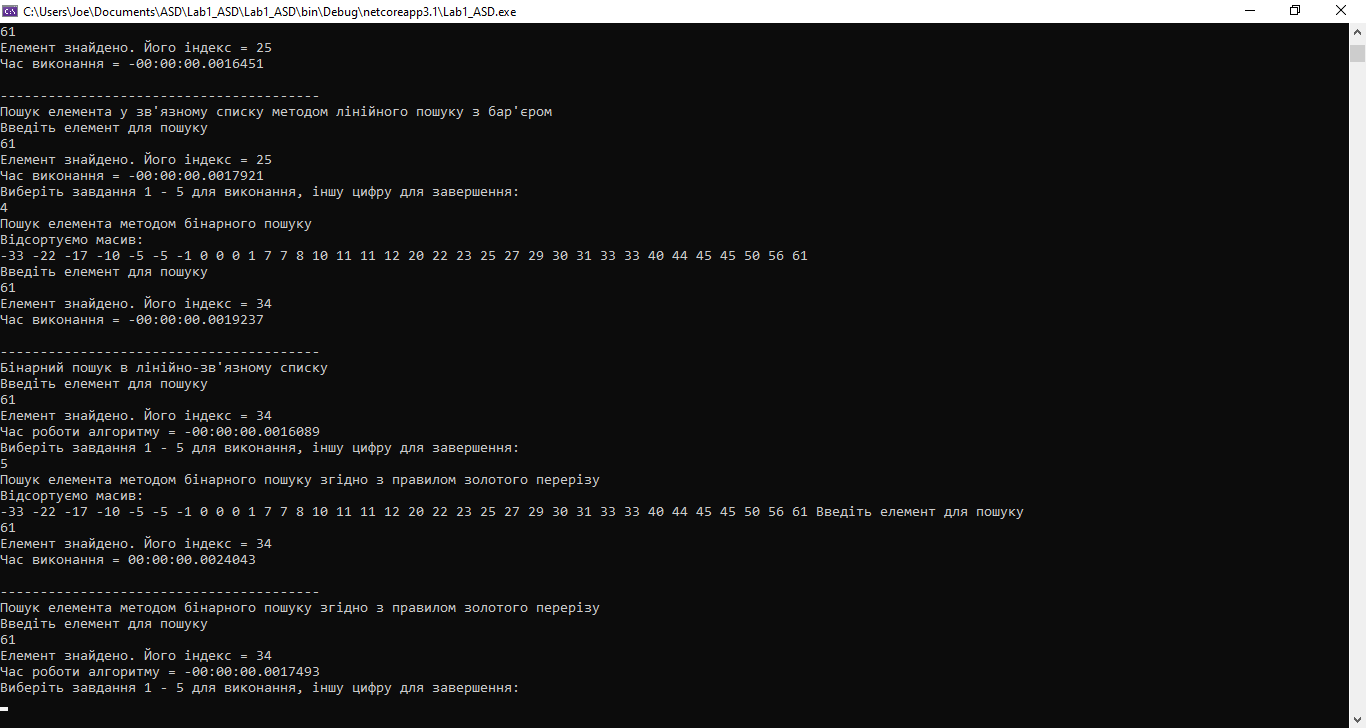


Тест 2

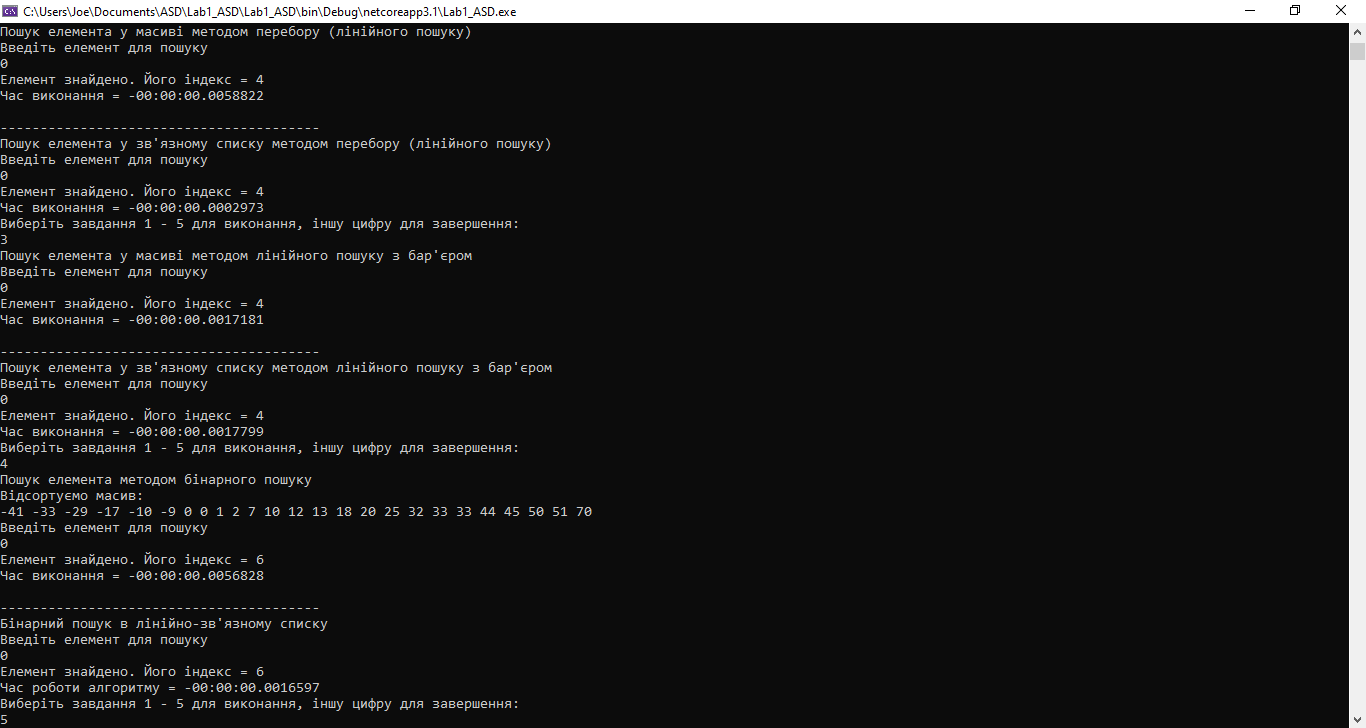


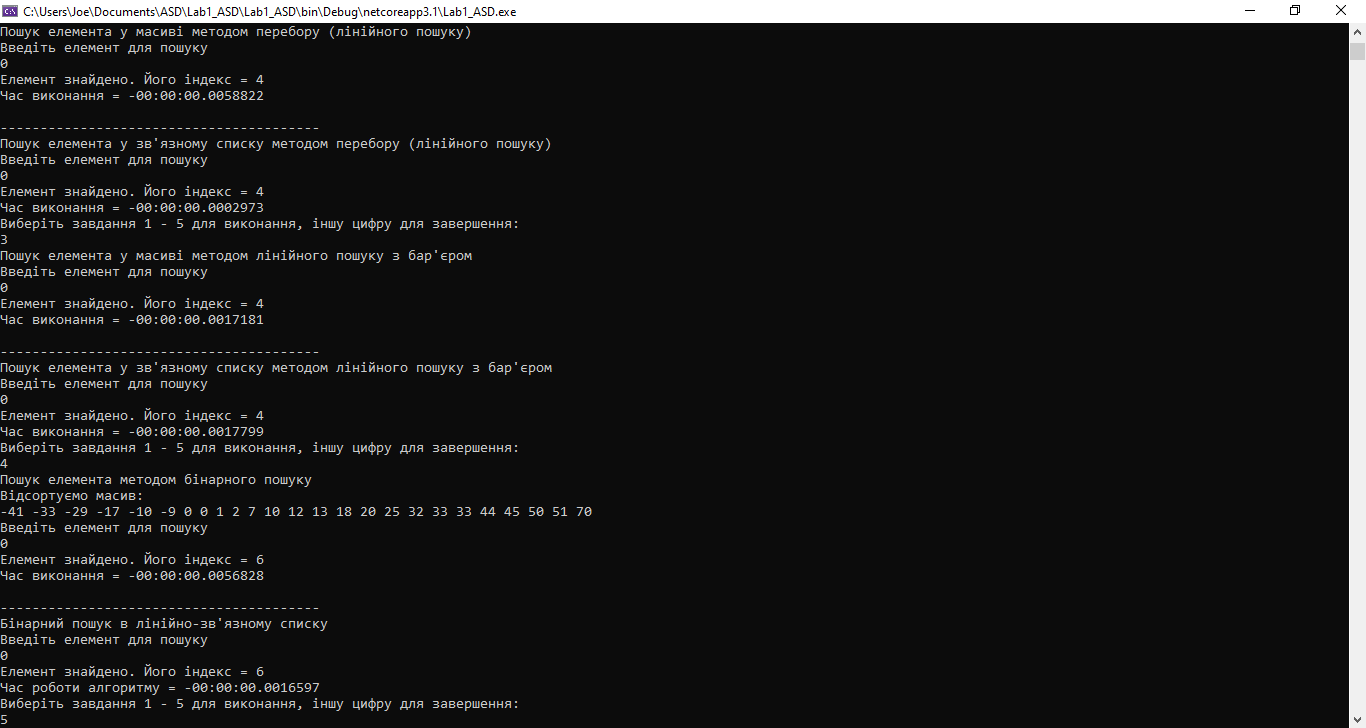


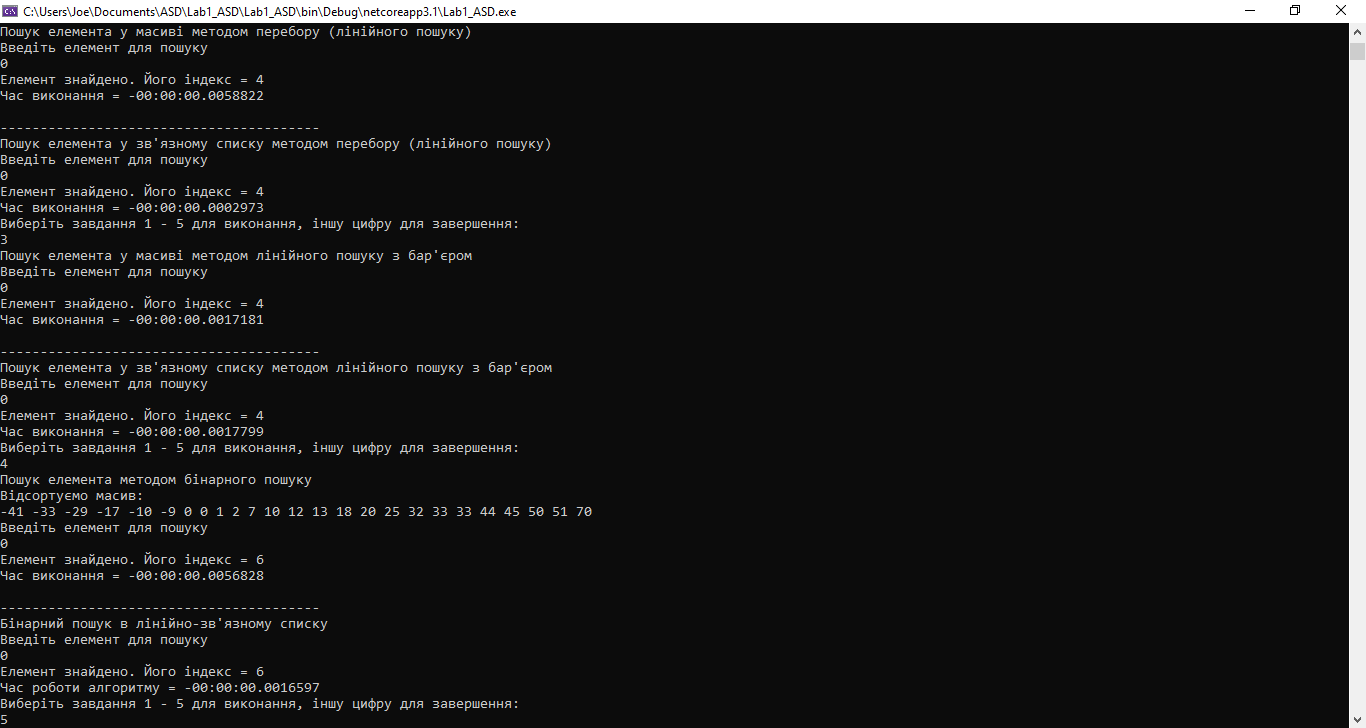


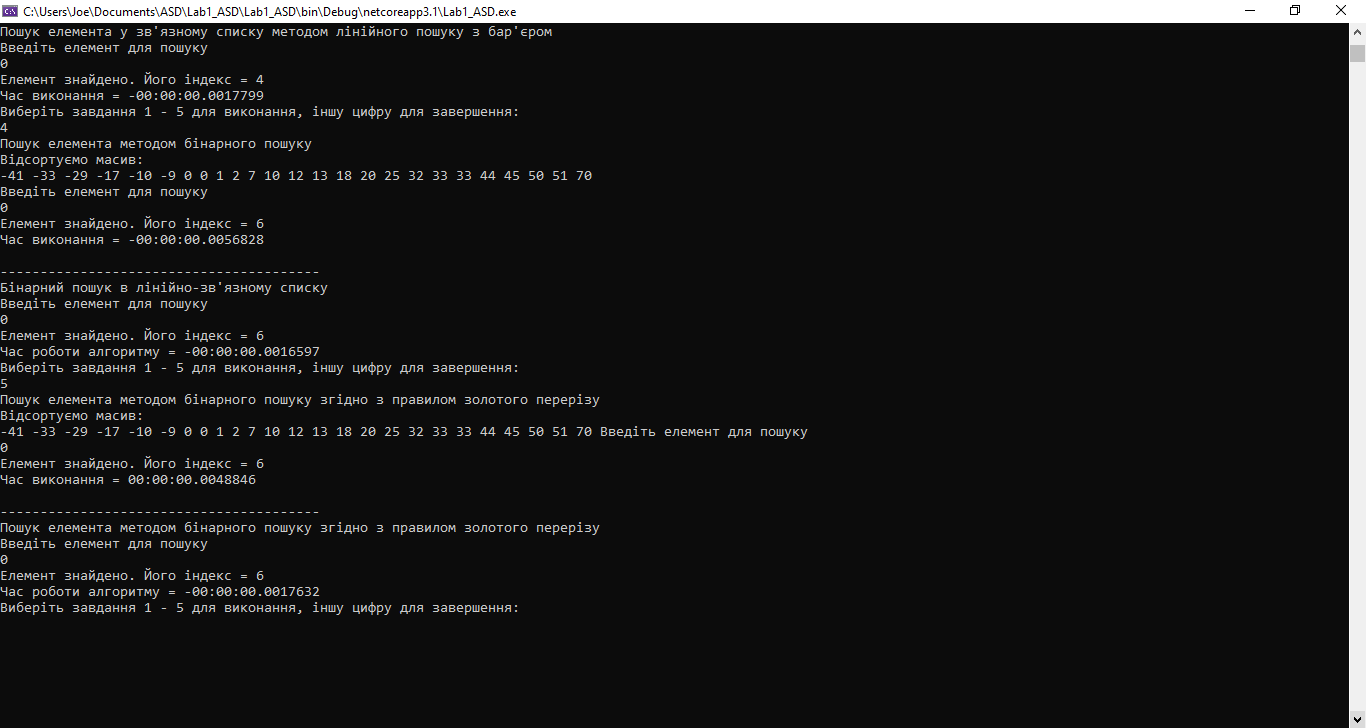


Тест 3

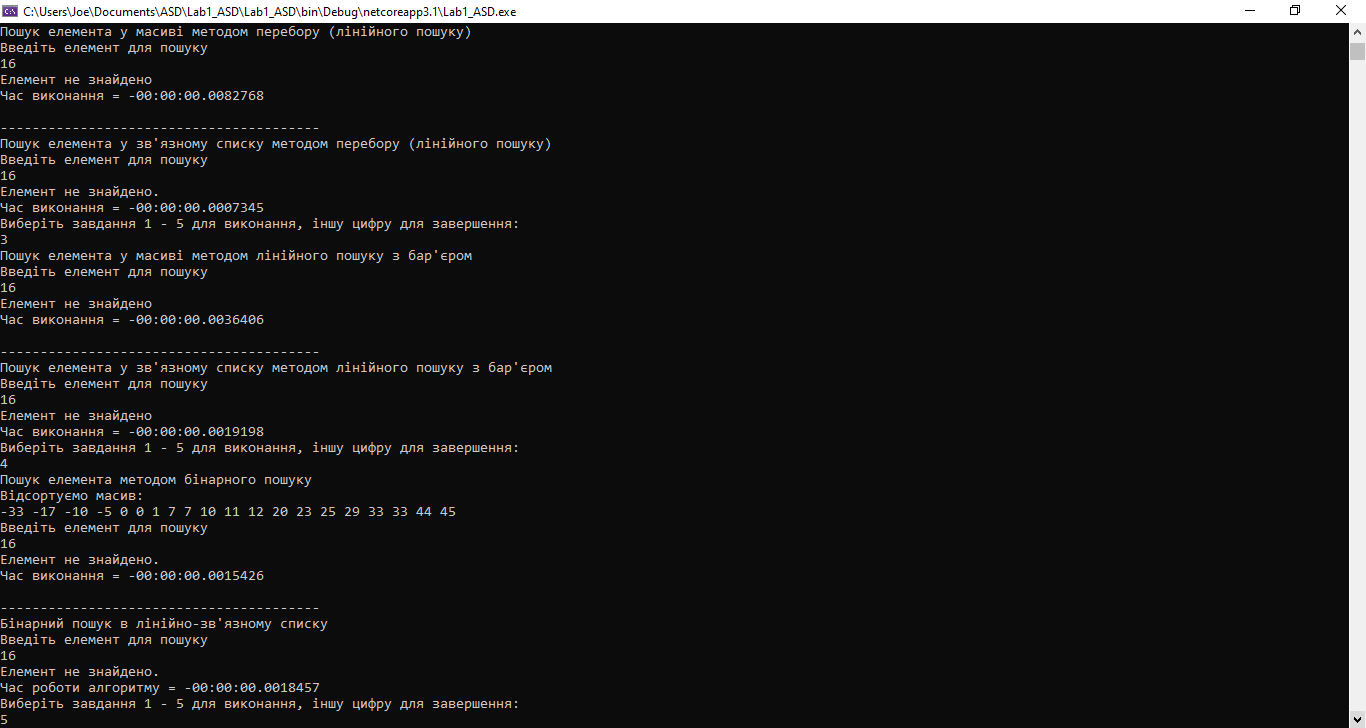


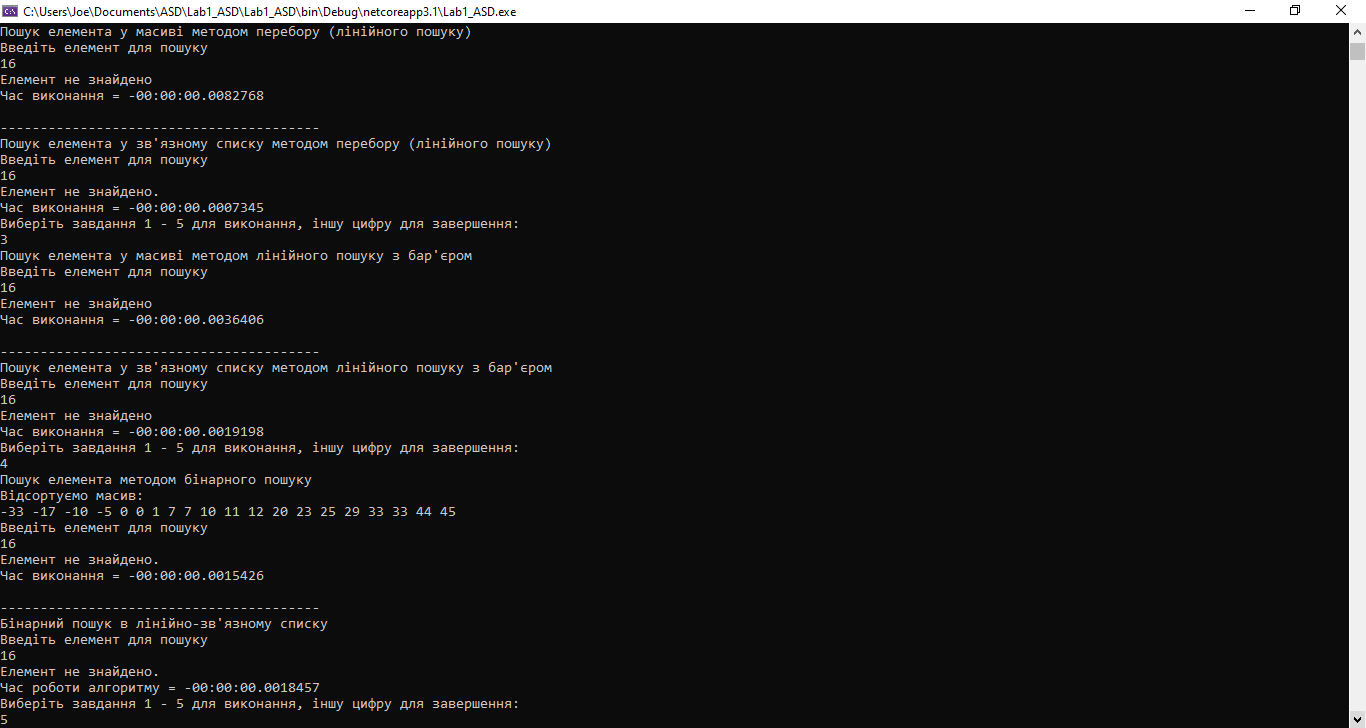


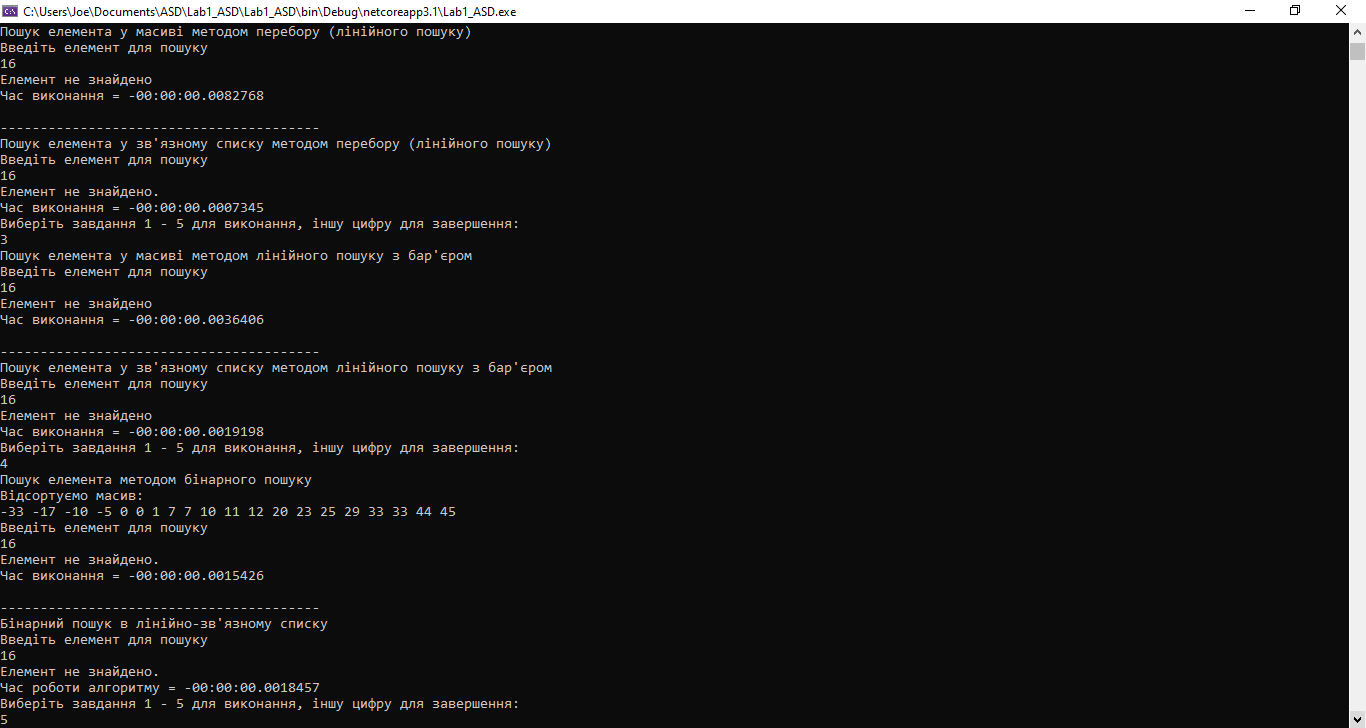


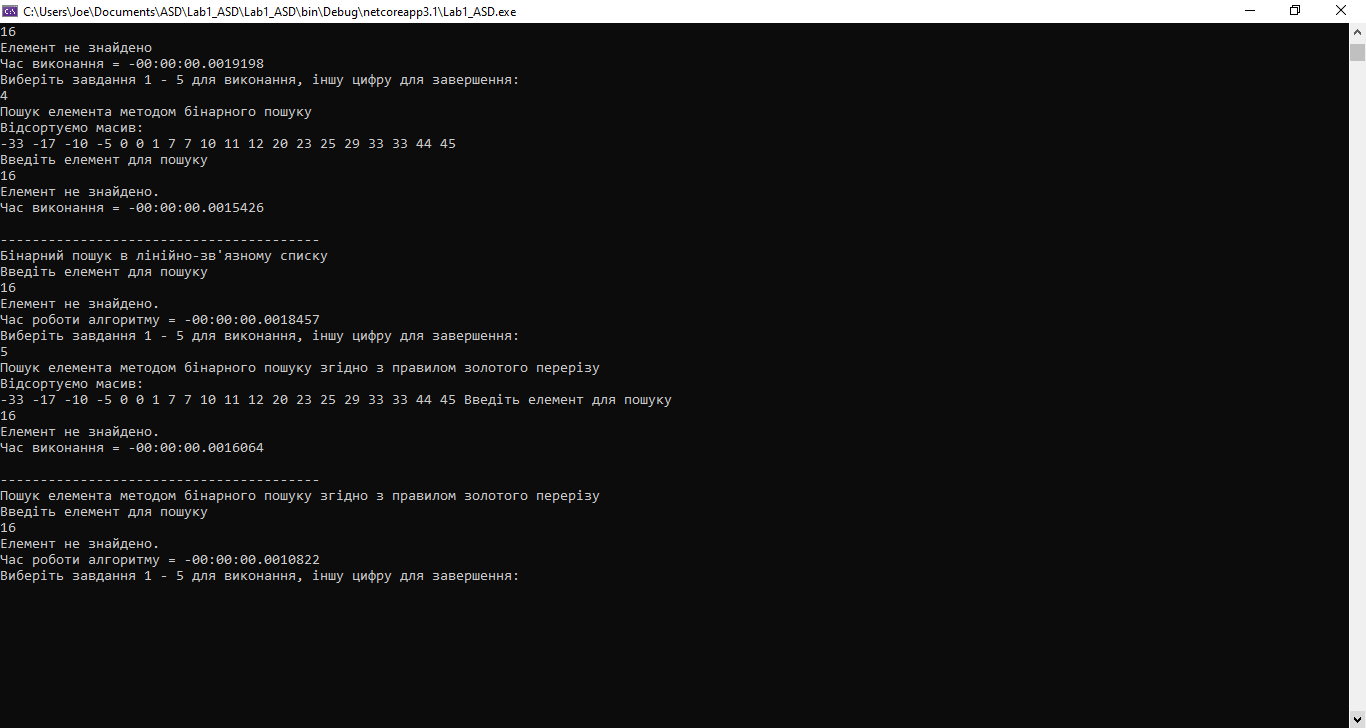


Тест 4









Порівнявши отримані результати з даними тестів, бачимо, що програма працює коректно.

У ході роботи була виправлена помилка підрахунку індексу знайденого елемента при бінарному пошуку у лінійному списку.

**Аналіз результатів**

Порівнявши результати часу, що потребується на виконання кожного алгоритму, можна помітити, що найменш ефективним є алгоритм звичайного лінійного пошуку. Швидкість бінарних пошуків та пошуку з бар’єром є приблизно однаковою, залежить від умови.

Також бачимо, що швидкість пошуку у лінійному списку є значно вищою, ніж у звичайному масиві.

**Висновок**

Було створено й протестовано програму для пошуку елементів різними методами у двох видах структур даних. Було проаналізовано ефективність різних способів.