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

Виконав:	Якубець Михайло Вікторович	**Перевірив**:	Бичков Олексій Сергійович
Група	ІПЗ-12	Дата перевірки	
Форма навчання	денна	Оцінка	
Спеціальність	121		
2022			

Мета: дослідити ефективність та швидкість виконання алгоритмів пошуку на різних структурах даних.

Умова завдання:
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:

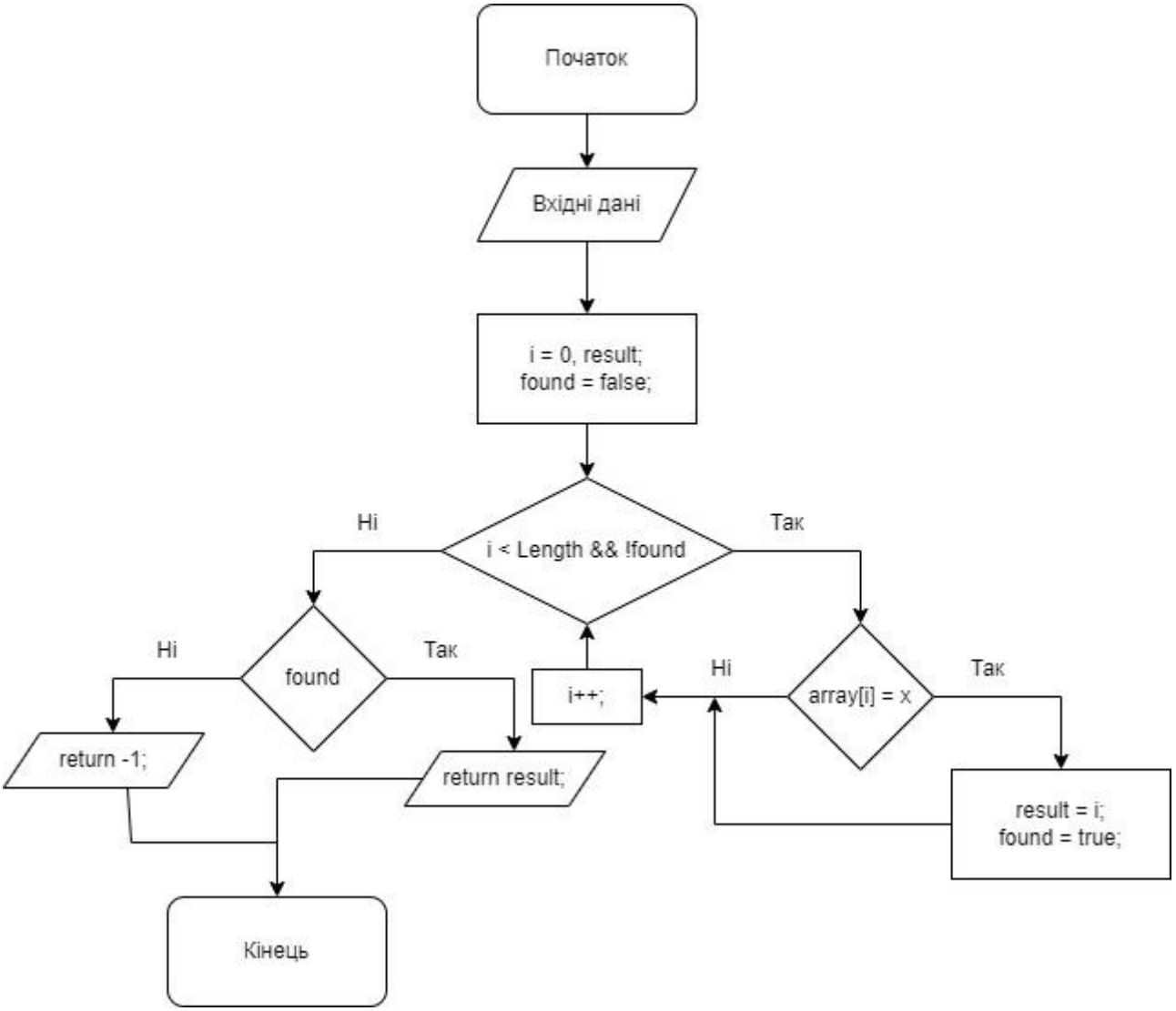
- пошуку перебором елемента масиву, що дорівнює заданому значенню.
- пошуку з бар'єром елемента масиву, що дорівнює заданому значенню.
- бінарного пошуку елемента масиву рівного заданому значенню.
- бінарного пошуку елемента масиву, рівного заданому значенню, в якій нове значення індексу m визначалося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

Аналіз завдання:
Для того, щоб дослідити роботу різних структур даних (масив, лінійний зв’язаний список) в алгоритмах пошуку, створимо класи Show для відображення результатів дослідження, LinkedList для використання як структуру даних лінійний зв’язаний список, Item для використання як вузла в списку, SearchInArray для зберігання всіх методів пошуку в масиві, та використаємо класи Random для генерації структур даних й Stopwatch для виміру часу виконання.

Структури даних:
Для справедливого виміру ефективності алгоритмів будемо використовувати однакові дані для обох структур. Також, для економії нашого часу, створимо методи генерації масиву у вказаному з клавіатури розмірі для 1 та 2 алгоритмів пошуку. Потім всі згенеровані випадкові дані перекинемо в лінійний зв’язаний список. Однак для 3 та 4 алгоритмів пошуку зробимо генерацію послідовних значень, від меншого до більшого, оскільки 3 та 4 алгоритми пошуку потребують мати відсортовану структуру даних. І останнє, для того, щоб не вибирати значення зі зненерованого масиву вручну, введемо вибір елемента пошуку як індекс цього елемента.

Алгоритми:

1. **Лінійний пошук**
   Блок-схема:



Код алгоритма для масиву:

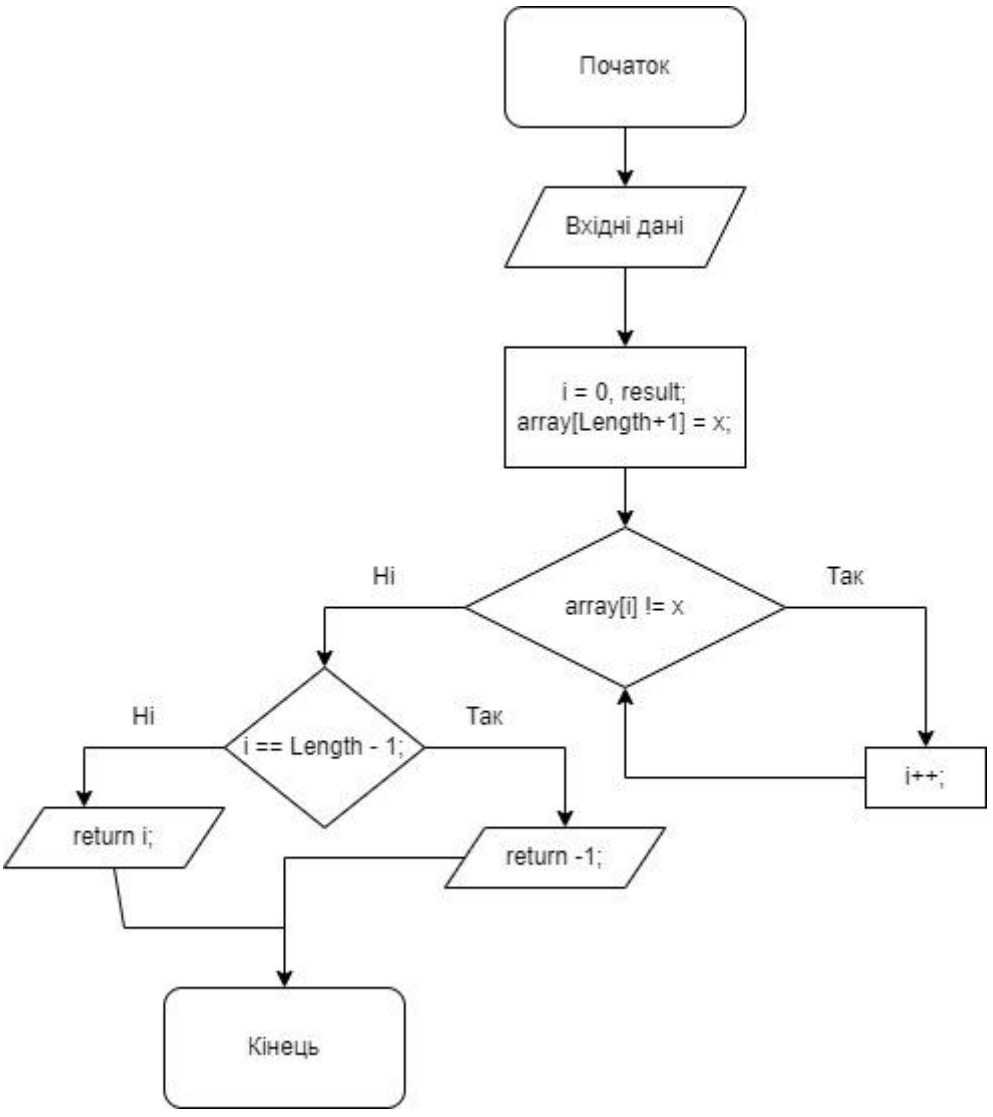
```
public int LinearSearch(int[] array, int elementToSearch)
{
    int i = 0, resultIndex = 0;
    bool found = false;

    while (i < array.Length && !found)
    {
        if (array[i] == elementToSearch)
        {
            resultIndex = i;
            found = true;
        }

        i++;
    }

    if (!found)
        return -1;

    return resultIndex;
}
```

Код алгоритма для лінійного зв’язаного списку:

```
public int LinearSearch(T elementToSearch)
{
    int i = 0, resultIndex = 0;
    bool found = false;

    foreach (var el:T in this){...}

    if (!found)
        return -1;

    return resultIndex;
}
```

2. **Пошук з бар'єром**

Блок-схема:



Код алгоритма для масиву:

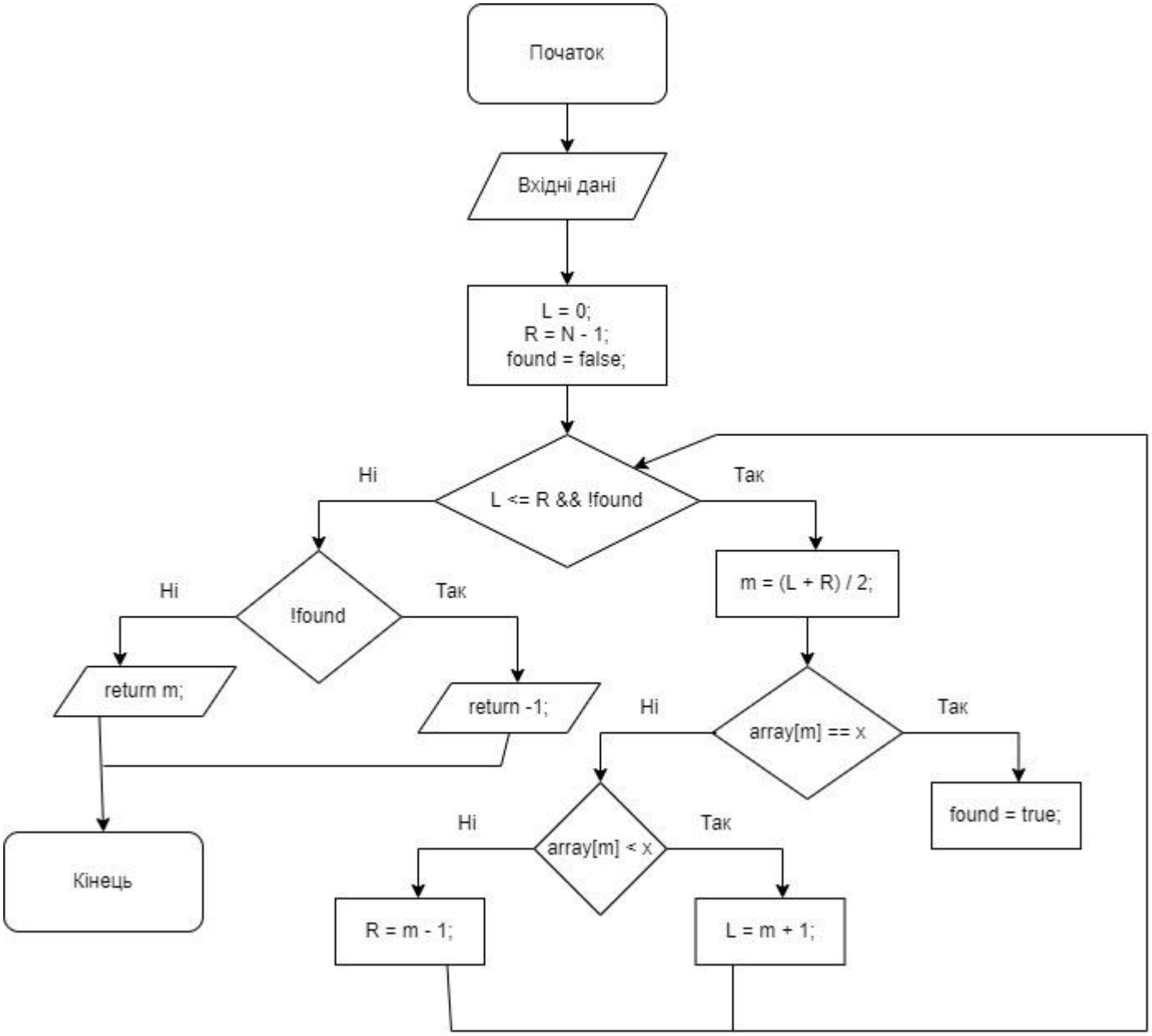
```
public int BarrierSearch(int[] array, int elementToSearch)
{
    Array.Resize(ref array, newSize: array.Length + 1);
    array[array.Length-1] = elementToSearch;

    int i = 0;

    while (array[i] != elementToSearch)
        i++;

    if (i == array.Length-1)
        return -1;

    return i;
}
```

Код алгоритма для лінійного зв’язаного списку:

```
public int BarrierSearch(T elementToSearch)
{
    AddLast(elementToSearch);

    int i = 0;

    foreach (var el:T in this)
    {
        if (!el.Equals(elementToSearch))
            i++;
        else
            break;
    }

    if (i == Count-1)
        return -1;

    return i;
}
```

3. **Бінарний пошук**

   Блок-схема:



Код алгоритма для масиву:

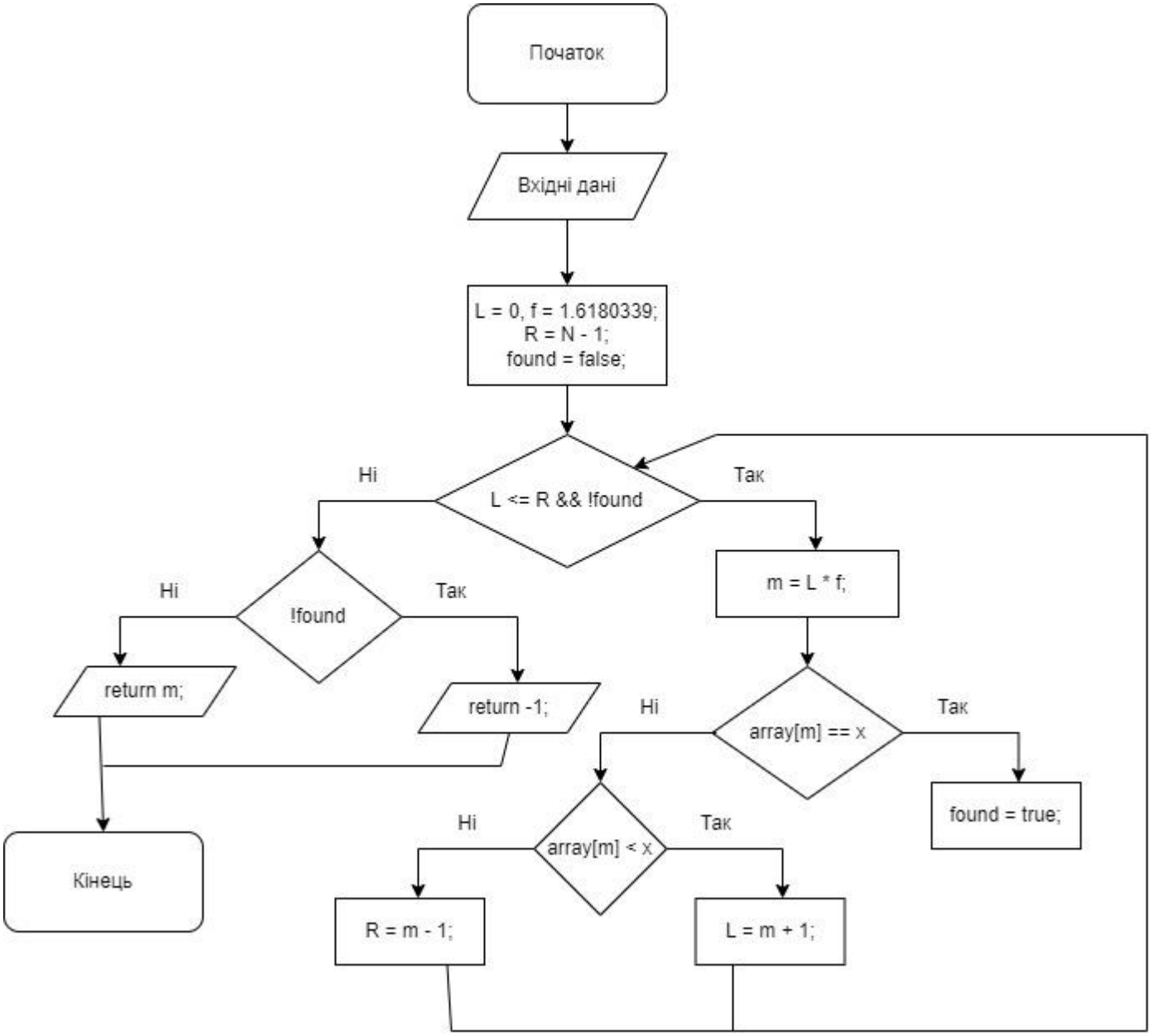
```
public int BinarySearch(int[] array, int elementToSearch, out int operationsAmount)
{
    int left = 0, m = -1;
    int right = array.Length - 1;
    bool found = false;

    operationsAmount = 0;

    while (left <= right && !found)
    {
        m = Convert.ToInt32((left + right) / 2);
        operationsAmount++;

        if (array[m] == elementToSearch)
            found = true;
        else if (array[m] < elementToSearch)
            left = m + 1;
        else
            right = m - 1;
    }

    if (!found)
        return -1;

    return m;
}
```

Код алгоритма для лінійного зв’язаного списку:

```
public int BinarySearch(T elementToSearch, out int operationsAmount)
{
    int left = 0;
    int right = Count - 1;
    bool found = false;

    int m = Convert.ToInt32(right / 2);
    Item<T> current = Get(m);
    int currentM = m;

    operationsAmount = m + 1;

    while (left <= right && !found)
    {
        int element = (int) (object) current.Data;

        if (element.Equals((int)(object)elementToSearch))
            found = true;
        else if (element < (int) (object) elementToSearch)
        {
            left = m + 1;
            m = Convert.ToInt32((left + right) / 2);

            for (int i = 0; i < m - currentM; i++)
            {
                current = current.Next;
```

```
                operationsAmount++;
            }
        }
        else
        {
            right = m - 1;
            m = Convert.ToInt32((left + right) / 2);

            current = Get(m);
            operationsAmount += m + 1;
        }

        currentM = m;
        operationsAmount++;
    }

    if (!found)
        return -1;

    return m;
}
```

4. **Бінарний пошук з правилом золотого перерізу**
   Блок-схема:



Код алгоритма для масиву:

```
public int BinarySearchWithGoldenRatio(int[] array, int elementToSearch, out int operationsAmount)
{
    int left = 0, m = -1;
    int right = array.Length - 1;
    bool found = false;

    operationsAmount = 0;

    while (left <= right && !found)
    {
        m = operationsAmount == 0 ? Convert.ToInt32(left * F) : Convert.ToInt32((left + right) / 2);

        operationsAmount++;

        if (array[m] == elementToSearch)
            found = true;
        else if (array[m] < elementToSearch)
            left = m + 1;
        else
            right = m - 1;
    }

    if (!found)
        return -1;

    return m;
}
```

Код алгоритма для лінійного зв’язаного списку:

```
public int BinarySearchWithGoldenRatio(T elementToSearch, out int operationsAmount)
{
    int left = 0;
    int right = Count - 1;
    bool found = false;

    int m = Convert.ToInt32(left * F);
    Item<T> current = Get(m);
    int currentM = m;

    operationsAmount = m + 1;

    while (left <= right && !found)
    {
        int element = (int) (object) current.Data;

        if (element.Equals((int)(object)elementToSearch))
            found = true;
        else if (element < (int) (object) elementToSearch)
        {
            left = m + 1;
            m = Convert.ToInt32((left + right) / 2);

            for (int i = 0; i < m - currentM; i++)
            {
                current = current.Next;
                operationsAmount++;
```

```
            }
        }
        else
        {
            right = m - 1;
            m = Convert.ToInt32((left + right) / 2);

            current = Get(m);
            operationsAmount += m + 1;
        }

        currentM = m;
        operationsAmount++;
    }

    if (!found)
        return -1;

    return m;
}
```

Результати:

Розміри структур даних		10			10^7		
Індекси шуканих елементів		0	3	9	0	$3*10^6$	$10^7 - 1$
Лінійний пошук	Масив	t = 0.8749 N = 1	t = 0.0081 N = 4	t = 0.0002 N = 10	t = 0.0004 N = 1	t = 13.2332 N = 3000001	t = 40.4061 N = 10000000

	Список	t = 9.1945 N = 1	t = 0.0141 N = 4	t = 000.001 3 N = 10	t = 0.0035 N = 1	t = 64.5366 N = 3000001	t = 202.5007 N = 10000000
Пошук з бар'єром	Масив	t = 7.0040 N = 1	t = 0.0213 N = 4	t = 0.0025 N = 10	t = 26.6240 N = 1	t = 25.9456 N = 3000001	t = 48.2667 N = 10000000
	Список	t = 0.6957 N = 1	t = 0.0044 N = 4	t = 0.0018 N = 10	t = 0.0021 N = 1	t = 69.7322 N = 3000001	t = 208.0286 N = 10000000
Бінарний пошук	Масив	t = 0.6376 N = 3	t = 0.0081 N = 4	t = 0.0005 N = 4	t = 0.0017 N = 23	t = 0.0023 N = 22	t = 0.0021 N = 24
	Список	t = 6.5371 N = 11	t = 0.0029 N = 13	t = 0.0010 N = 14	t = 42.4704 N = 10000015	t = 163.0543 N = 38933348	t = 44.7466 N = 10000024
Бінарний пошук з правилом золотого перерізу	Масив	t = 2.4778 N = 1	t = 0.0088 N = 4	t = 0.0006 N = 5	t = 0.0057 N = 1	t = 0.0032 N = 24	t = 0.0026 N = 25
	Список	t = 1.0986 N = 2	t = 0.0028 N = 14	t = 0.0011 N = 15	t = 0.0009 N = 2	t = 171.5153 N = 41933363	t = 43.5570 N = 10000025

де t – час виконання алгоритму в мс, N – кількість операцій, а клітинки з чорним фоном – це тести, де лінійний зв’язаний список швидше за масив.

Висновок.

В цій лабораторній роботі було досліджено ефективність та швидкість виконання алгоритмів пошуку на різних структурах даних (масив, лінійний зв’язаний список). Як виявилось, масив значно швидший за лінійний зв’язаний список в ≈67% через те, що в нього операція зчитування – O(1), на відміну від списку, де та сама операція займає O(n) в найгіршому випадку. Однак стосовно пошуку з бар'єром не все так однозначно, тому що масив потребує переписати всі значення в нову частину пам'яті, а це – O(n) в будь-якому випадку, але лінійний зв’язаний список легко додає новий елемент, а це – O(1).