**Звіт**

**з 1 лабораторної роботи**

**з дисципліни**

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

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.

**Масиви**

0)Основа програми

Щоб почати писати алгоритми пошуку, нам треба створити масиви.

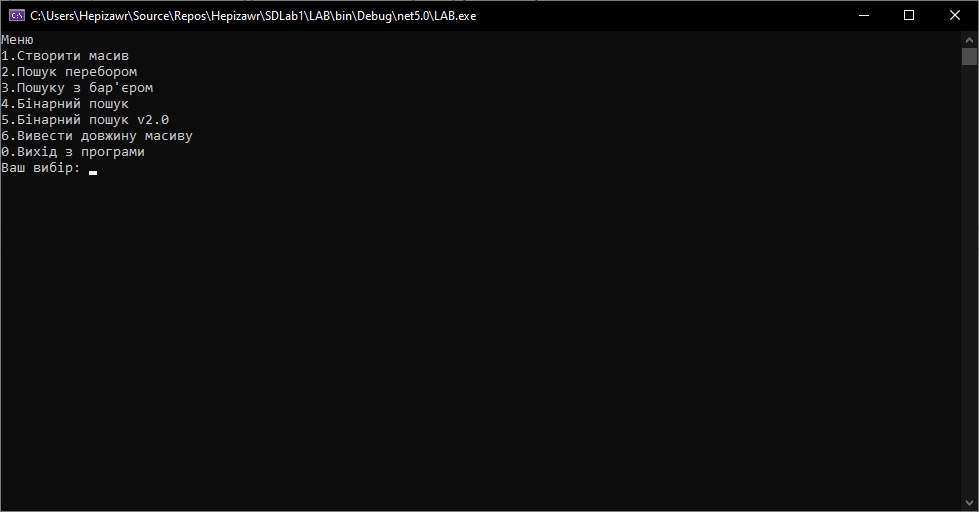
Для цього я написав 3 методи:

1.Заповнення масив вписуючи кожен елемент самостійно(Program.MasHand())

2.Заповнення масиву псевдо-випадковими числами(Program.MasRnd())

3.Заповнення масиву псевдо-випадковими числами в порядку зростання (Program.MasRndSort())

Для зручності визову методів з консолі також було реалізовано меню:

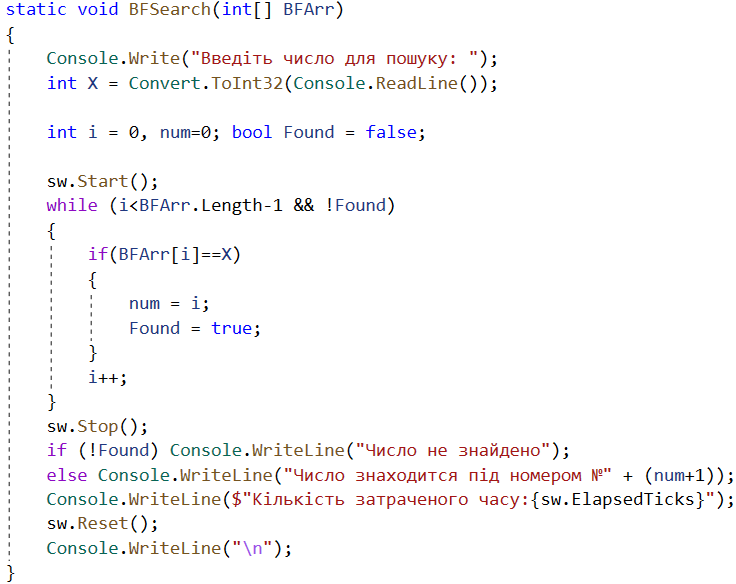


1)Пошук перебором

Пошук перебором – найпростіший в розумінні алгоритм.

При написанні цього алгоритму я урахував замітки з методички.

Ніяких проблем при написанні невиникла.



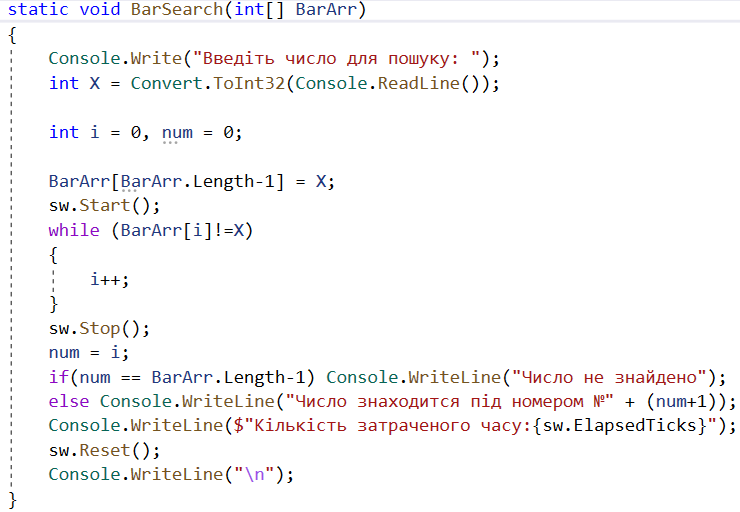
Ефективність:

Використовуючи метод ElapsedTicks класу StopWatch, при створенні масиву з 1000 елементів з діапазоні [-100;99] і завданні знайти число 100, алгоритм перебере весь масив і зрозуміє що такого елементу не має приблизно за 42 тіки. Стандартний же пошук існуючого числа займає приблизно від 11(елемент знаходиться не далеко від початку масиву) до 35 тіків (елемент знаходиться десь далі ніж середина масиву).

2)Пошук з бар’єром

Ще простіший в реалізації алгоритм ніж пошук перебором.

Але треба враховувати, що при створенні масиву треба не забути ще ініціалізувати додатковий останній пустий елемент масиву, який і буде слугувати бар’єром.



Ефективність:

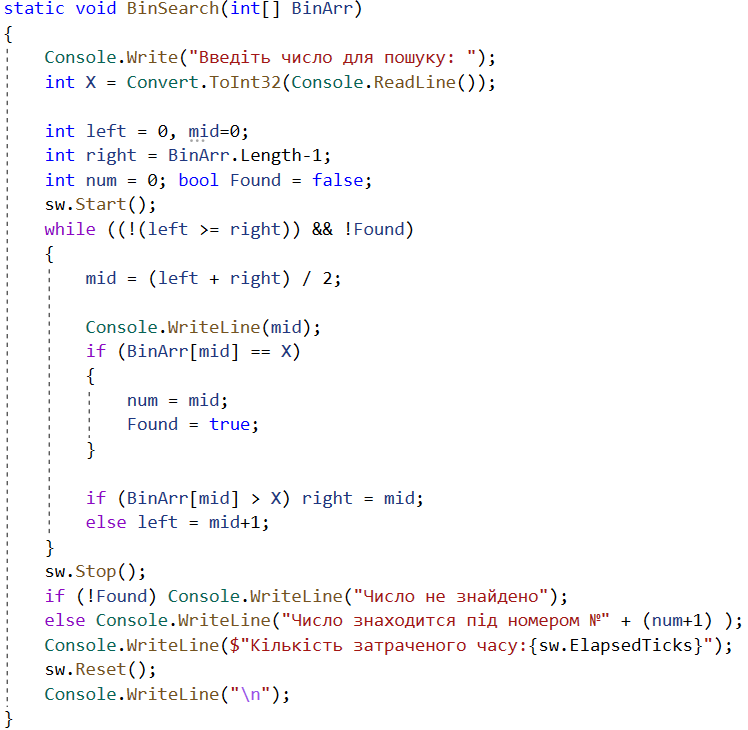
При спробі знайти неіснуючий елемент в масиві з 1000 елементів алгоритм закінчує свою роботу десь за 30 тіків. Стандартний же пошук існуючого числа займає приблизно від 8(елемент знаходиться не далеко від початку масиву) до 24 тіків (елемент знаходиться десь далі ніж середина масиву)

Можемо зробити висновок що алгоритм пошуку з бар’єром є ефективнішим за пошук перебором.

3)Бінарний пошук

Являє собою найскладніший в розумінні та написанні алгоритм відносно минулих алгоритмів.

Саме для цього алгоритму я написав метод створення псевдо-випадкового відсортованого масиву. Адже суть цього алгоритму полягає в порівняні величини поточного числа та шуканого.



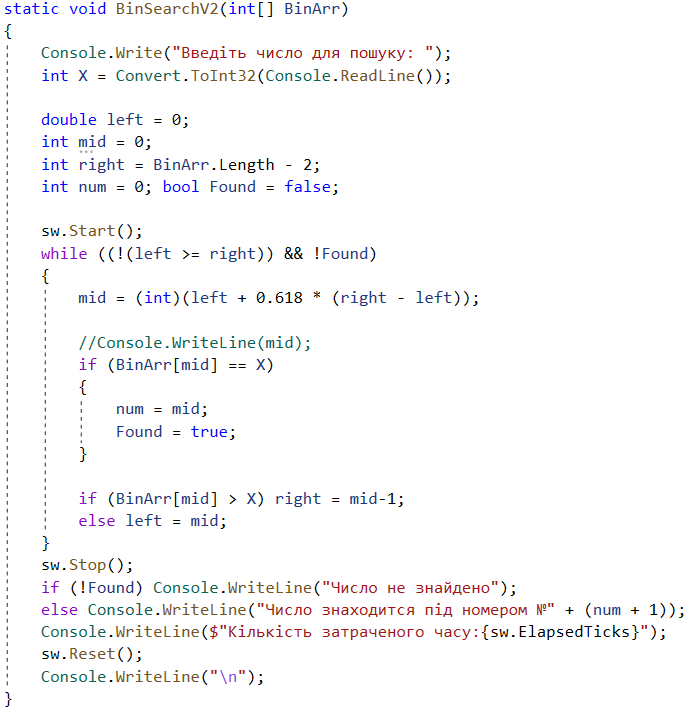
Ефективність:

При спробі знайти неіснуючий елемент в масиві з 1000 елементів алгоритм закінчує свою роботу десь за 10 тіків. Стандартний же пошук існуючого числа займає приблизно від 8 до 14 тіків.

Бінарний пошук куди ефективніший за пошук перебором та пошук з бар’єром так як для знаходження числа в будь-якому місці масиву він затрачає приблизно однаковий час

4)Бінарний пошук(з правилом золотого переріз)

При стандартному бінарному пошуку поточний елемент знаходиться сумою границь діленою на два. Для правила золотого перерізу треба ділити на 1.61803… , а точніше я використовував формулу: left + 0.618 \* (right - left);



Ефективність:

При спробі знайти неіснуючий елемент в масиві з 1000 елементів алгоритм закінчує свою роботу десь за 13 тіків. Стандартний же пошук існуючого числа займає приблизно від 13 до 15 тіків.

Ми видимо що правило Бінарний пошук з правилом золотого переріз є приблизно однаковий за ефективністю порівнюючи зі звичайним Бінарним пошуком, навіть іноді буває менш ефективнішим.