Politechnika Śląska w Gliwicach
Wydział Matematyki Stosowanej

Dokumentacja projektu:

Mediana w zbiorze liczb rzeczywistych

Szpak Kamil
gr. 3H

Spis treści

Repozytorium GitHub

https://github.com/Presoon/Egzamin-AiSD-Mediana-Liczb-Rzeczywistych

1. Teoria matematyczna

Co to jest mediana liczb?

Mediana – to wartość środkowa w uporządkowanym ciągu liczb.

Jak obliczamy medianę w zbiorze?

Nasze rozważania należy rozbić na dwa przypadki:

1. Mediana w uporządkowanym ciągu, w którym jest nieparzysta liczba elementów:

Dany ciąg uporządkowany rosnąco: 2, 5, 7, **8**, 9, 10, 32.
Zatem Mediana **M = 8**, ponieważ jest to środkowy wyraz ciągu liczbowego. Jest to czwarty element niezależnie, czy liczymy od początku, czy od końca tego ciągu liczb.

2. Mediana w ciągu uporządkowanym rosnąco o parzystej liczbie elementów tego ciągu.

Dany ciąg uporządkowany rosnąco: 2, 3, 4, 5, 6, 7. Jeśli ciąg posiada parzystą ilość elementów, wówczas bierzemy sumę dwóch środkowych elementów i wyliczamy ich średnią arytmetyczną. W tym przypadku dwa wyrazy są środkowe, czyli równoodległe od początku i końca ciągu. Medianą będzie średnia arytmetyczna tych dwóch środkowych liczb:

$$M = \frac{4+5}{2} = \frac{9}{2} = 4,5$$

2. Budowa programu, opis kodu

Program został napisany w języku C# przy użyciu frameworka .NET.
W implementacji zostały dodane komentarze które mogą zostać wykorzystane do generowania dokumentacji kodu.
W programie zostały zaimplementowane 3 klasy:

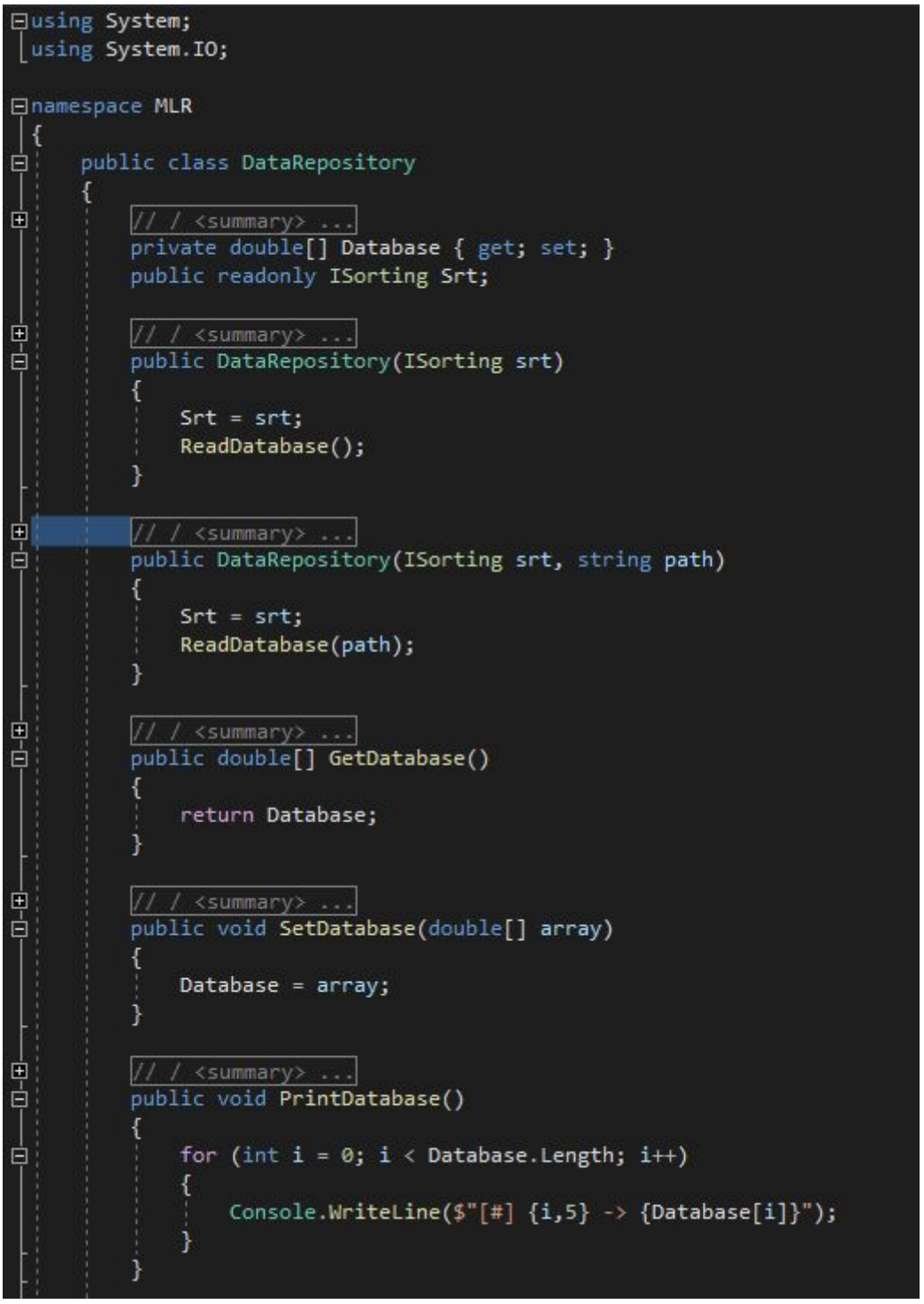
- DataRepository - przechowująca nasza bazę liczb rzeczywistych
- Sorting - klasa zawierająca zaimplementowane różne algorytmy sortowania
- Mediana - klasa zawierająca algorytm obliczający medianę na uporządkowanym zbiorze.

Program.cs - funkcja Main

```
using System;

namespace MLR
{
    internal static class Program
    {
        private static void Main()
        {
            //Inicjalizacja klasy obliczającej medianę
            Mediana md = new Mediana();

            //Inicjalizacja Klasy bazy danych
            DataRepository db = new DataRepository(new Sorting(), @"ExDatabases/database-10000.txt");
            //DataRepository db = new DataRepository(new Sorting(), @"ExDatabases/database-100000.txt");
            //DataRepository db = new DataRepository(new Sorting());


            //Wykorzystanie algorytmów sortujących w celu
            //przygotowania bazy pod obliczenie mediany
            db.SetDatabase(db.Srt.HeapSort(db.GetDatabase()));
            //db.SetDatabase(db.Srt.ShellSort(db.GetDatabase()));
            //db.SetDatabase(db.Srt.BubbleSort(db.GetDatabase()));


            //Wypisanie bazy danych i wypisanie mediany zbioru
            db.PrintDatabase();
            Console.WriteLine($"\n[#] Mediana zbioru to: {md.GetMedian(db.GetDatabase())}");

            Console.ReadKey();
        }
    }
}
```

W głównej funkcji programu inicjalizujemy obiekt klasy *Mediana*, następnie inicjalizujemy obiekt *DataRepository* który w konstruktorze oczekuje podania klasy sortowania, oraz opcjonalnie ścieżki do pliku zbioru liczb. W przypadku gdy ścieżka nie zostanie podana, program wczyta bazę z lokalizacji jego wywołania.Następnie przy użyciu algorytmów sortujących (w naszym przypadku *HeapSort*) porządkujemy nasz zbiór liczb. Następuje wywołanie metody wypisującej zbiór a następnie za pomocą metody *GetMedian(double[])* otrzymujemy medianę zbioru.

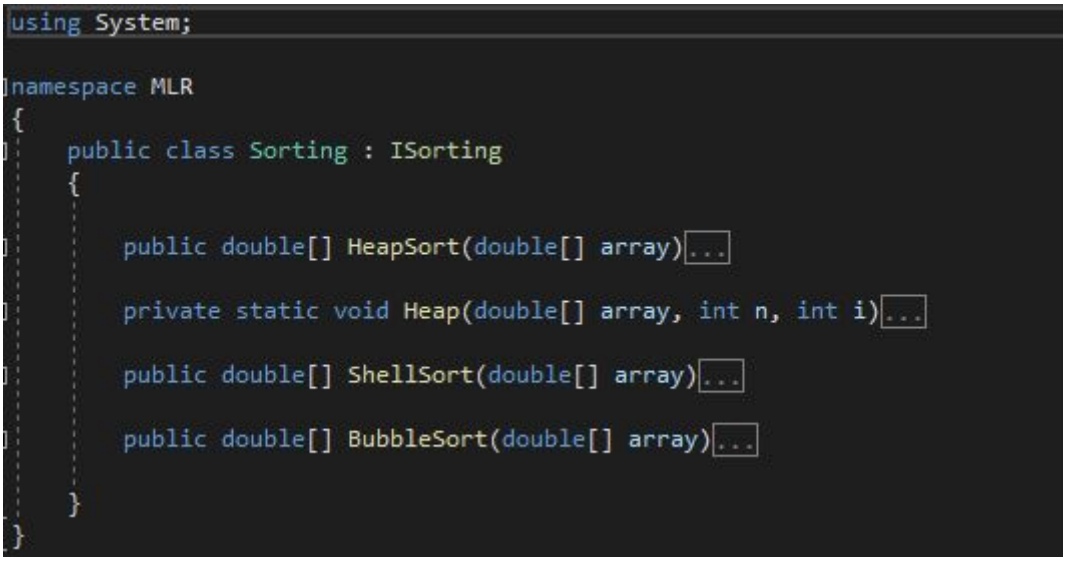
DataRepository.cs - zbiór liczb



```
using System;
using System.IO;

namespace MLR
{
    public class DataRepository
    {
        /// <summary> ...
        private double[] Database { get; set; }
        public readonly ISorting Srt;

        /// <summary> ...
        public DataRepository(ISorting srt)
        {
            Srt = srt;
            ReadDatabase();
        }

        /// <summary> ...
        public DataRepository(ISorting srt, string path)
        {
            Srt = srt;
            ReadDatabase(path);
        }

        /// <summary> ...
        public double[] GetDatabase()
        {
            return Database;
        }

        /// <summary> ...
        public void SetDatabase(double[] array)
        {
            Database = array;
        }

        /// <summary> ...
        public void PrintDatabase()
        {
            for (int i = 0; i < Database.Length; i++)
            {
                Console.WriteLine($"[#] {i,5} -> {Database[i]}");
            }
        }
```

Podstawowe metody klasy DataRepository pozwalają na pobranie bazy, ustawienie bazy, wypisanie w przejrzysty sposób naszego zbioru.
Konstruktor bazy został przygotowany na możliwość opcjonalnego podania ścieżki do zewnętrznej bazy zbioru liczb rzeczywistych.
Konstruktory korzystają z interfejsu ISorting pozwalającego na inicjalizację sortowania w trakcie inicjalizacji Bazy.

Sorting.cs - implementacja algorytmów sortowania



Użytkownik programu ma możliwość wybrania jednego z trzech sortowań:

- HeapSort
- ShellSort
- BubbleSort

w zależności od używanej bazy liczb.

Mediana.cs - obliczanie mediany zbioru

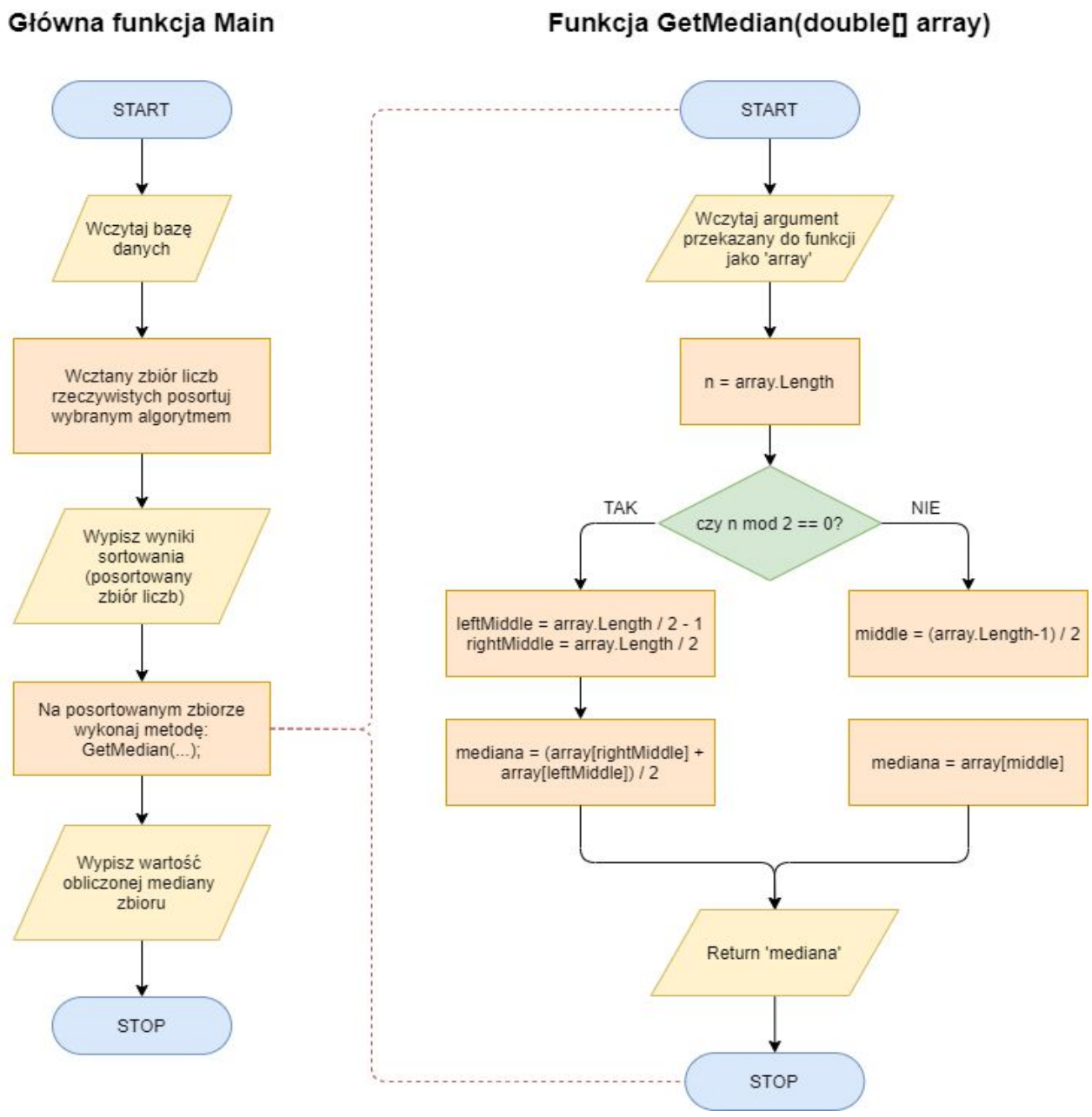
```
namespace MLR
{
    public class Mediana
    {
        /// <summary>
        ///     Metoda obliczająca medianę z posortowanej bazy danych.
        /// </summary>
        /// <param name="array">Posortowana baza danych typu double.</param>
        /// <returns>Wartość mediany dla zbioru, typu double</returns>
        public double GetMedian(double[] array)
        {
            double median;

            //parzysta liczba elementów
            if (array.Length % 2 == 0)
            {
                int leftMiddle = array.Length / 2 -1;
                int rightMiddle = array.Length / 2 ;

                median = (array[rightMiddle] + array[leftMiddle]) / 2;
            }
            //nieparzysta liczba elementów
            else
            {
                int middle = (array.Length-1) / 2;
                median = array[middle];
            }

            return median;
        }
    }
}
```

Implementacja prostego algorytmu przygotowanego na dwa przypadki:

- liczebność zbioru liczb jest parzysta
- liczebność zbioru liczb jest nieparzysta

Funkcja zwraca medianę jako wartość typu double.

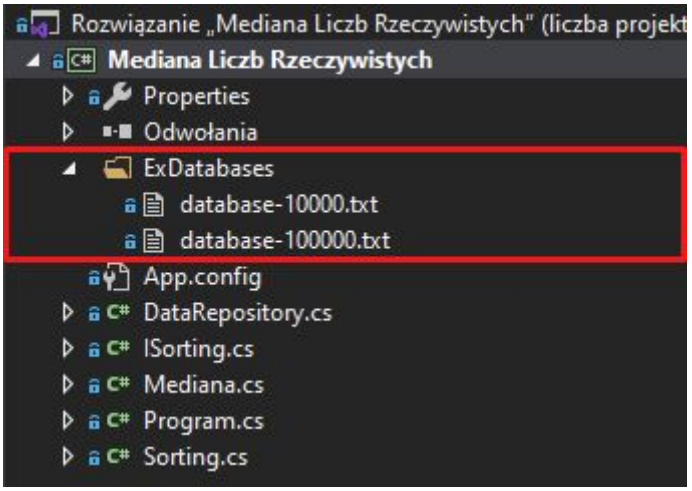
3. Schematy blokowe



Schematy algorytmów sortujących znajdują się na moim repozytorium GitHub:

- HeapSort - Link
- ShellSort - Link
- BubbleSort - Link

4. Testowanie programu



Na potrzeby testów programu zostały wygenerowane 2 bazy zawierające zbiory liczb o liczebności kolejno: 10000 oraz 100000 elementów.
Przedział generowanych liczb to 0-1000, a generowane liczby są typu rzeczywistego z dokładnością do 7 miejsc po przecinku.

Bazy wygenerowano przy użyciu narzędzia onlinerandomtools.

Uruchomienie programu:

```
[#]  9973 -> 9974,6083385
[#]  9974 -> 9975,2958684
[#]  9975 -> 9975,4587543
[#]  9976 -> 9976,3976534
[#]  9977 -> 9976,8133646
[#]  9978 -> 9977,027887
[#]  9979 -> 9977,7128428
[#]  9980 -> 9977,9290549
[#]  9981 -> 9978,006976
[#]  9982 -> 9979,0574807
[#]  9983 -> 9979,5342475
[#]  9984 -> 9979,9765097
[#]  9985 -> 9980,8817885
[#]  9986 -> 9982,1439902
[#]  9987 -> 9982,3917967
[#]  9988 -> 9986,8945952
[#]  9989 -> 9990,909171
[#]  9990 -> 9991,4003117
[#]  9991 -> 9991,5719781
[#]  9992 -> 9992,9745815
[#]  9993 -> 9995,3265244
[#]  9994 -> 9996,095728
[#]  9995 -> 9996,2959951
[#]  9996 -> 9996,838199
[#]  9997 -> 9998,0717866
[#]  9998 -> 9998,7758834
[#]  9999 -> 9999,3694981

[#] Mediana zbioru to: 5012,8445503
```

Program wypisuje posortowaną bazę liczb, a następnie wypisuje obliczoną medianę zbioru.

5. Użyte biblioteki

- using System;
- using System.IO;