

Wydział: Informatyka stosowana	Imię Nazwisko: Paweł Grabacki	Rok: I	Grupa: lab7	Numer Indeksu: 15939	
Pracownia: Berlin WSEI Kraków	Temat: <b>Implementacja interfejsu w klasie reprezentującej ułamki oraz sortowanie obiektów tej klasy przy użyciu algorytmu Bubble sort.</b>			Stanowisko: 17	
Data wykonania: 22.03.2025	Data oddania: 23.03.2025	Nr ćwiczenia: 1	Data zaliczenia:	Zwrot do poprawy:	Ocena:

## 1. CEL ĆWICZENIA

Celem ćwiczenia było zapoznanie się z implementacją interfejsu `Comparable<Ulamek>` w języku C#, a następnie wykorzystanie go do posortowania tablicy obiektów reprezentujących ułamki zwykłe. Sortowanie zostało zrealizowane przy pomocy algorytmu **Bubble sort**.

## 2. OPIS TEORETYCZNY

### 2.1 Interfejs `Comparable<Ulamek>`

Interfejs służy do definiowania sposobu porównywania obiektów danego typu. Wymusza implementację metody:

```
public int CompareTo(Ulamek other)
```

która powinna zwrócić:

- 0, jeśli obiekty są równe,
- wartość ujemną, jeśli bieżący obiekt jest mniejszy niż other,
- wartość dodatnią, jeśli jest większy.

Złożoność czasowa:

- Średnia i pesymistyczna:  $O(n^2)$
- Najlepszy przypadek (gdy dane są już posortowane):  $O(n)$

```
static void BubbleSort(Ulamek[] array)
{
    int n = array.Length;
    bool swapped;

    for (int i = 0; i < n - 1; i++)
    {
        swapped = false;

        for (int j = 0; j < n - i - 1; j++)
        {
            if (array[j].CompareTo(array[j + 1]) > 0)
            {
                // Swap
                Ulamek temp = array[j];
                array[j] = array[j + 1];
                array[j + 1] = temp;

                swapped = true;
            }
        }

        if (!swapped)
            break;
    }
}
```

## 2.2 Ułamki

Ułamek zwykły reprezentowany jest przez licznik i mianownik. W celu porównania dwóch ułamków można skorzystać z operacji równości po sprowadzeniu do wspólnego mianownika lub poprzez porównanie wartości dziesiętnej: licznik / mianownik.

## 2.3 Algorytm sortowania bąbelkowego ang.(bubble sort)

Sortowanie bąbelkowe to prosty algorytm polegający na wielokrotnym przechodzeniu przez tablicę i zamienianiu miejscami sąsiadujących elementów, jeśli są w złej kolejności. Proces ten powtarzany jest aż do momentu, gdy cała tablica zostanie posortowana

## 3. WYKONANIE ĆWICZENIA

1. Utworzono klasę Ułamek, zawierającą pola: licznik, mianownik, konstruktor, metody pomocnicze oraz przeciążone operatory  $>$  i  $<$ .
2. W klasie Ułamek zaimplementowano interfejs `Comparable<Ułamek>` poprzez zdefiniowanie metody `CompareTo()`, która porównuje dwa ułamki wykorzystując porównanie przekątne ( $a/b$  vs  $c/d \rightarrow ad$  vs  $cb$ ).
3. Utworzono tablicę obiektów Ułamek i wypełniono ją kilkoma przykładowymi wartościami.
4. Zaimplementowano metodę `BubbleSort(Ułamek[] array)`, w której wykorzystano metodę `CompareTo()` do porównywania elementów i sortowania ich rosnąco.

Przed i po sortowaniu wypisywano tablicę do konsoli w celu sprawdzenia poprawności działania algorytmu.

Tablica przed sortowaniem:

1/7  
6/7  
3/7  
2/7

Tablica po sortowaniu (bubble sort):

1/7  
2/7  
3/7  
6/7

## 4. WNIOSKI

- Implementacja interfejsu `Comparable<Ułamek>` pozwala na elastyczne i wielokrotne wykorzystywanie operacji porównania dla niestandardowych typów danych.
- Bubble sort, choć prosty w implementacji, jest mało wydajny dla dużych zbiorów danych. Sprawdza się jednak dobrze jako przykład do nauki działania algorytmów sortowania.

**Program poprawnie posortował tablicę ułamków rosnąco według ich wartości dziesiętnej.**

## 5.KOD

```
using System;
using System.Collections.Generic;
using System.Diagnostics.Contracts;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Gr7Lab1
{
    class Ułamek : IComparable<Ułamek>
    {
        public int licznik;
        public int mianownik;

        public Ułamek(int inLicznik, int inMianownik)
        {
            if (inMianownik == 0)
                throw new ArgumentException("Mianownik nie może być zerem!");

            licznik = inLicznik;
            mianownik = inMianownik;
        }

        public override string ToString()
        {
            return licznik + "/" + mianownik;
        }

        public static Ułamek operator *(Ułamek a, Ułamek b)
        {
            return new Ułamek(a.licznik * b.licznik, a.mianownik * b.mianownik);
        }

        public static bool operator >(Ułamek a, Ułamek b)
        {
            return a.licznik * b.mianownik > b.licznik * a.mianownik;
        }

        public static bool operator <(Ułamek a, Ułamek b)
        {
            return a.licznik * b.mianownik < b.licznik * a.mianownik;
        }

        public static explicit operator double(Ułamek a)
        {
            return (double)a.licznik / a.mianownik;
        }

        public int CompareTo(Ułamek other)
        {
            if (this > other) return 1;
            if (this < other) return -1;
            return 0;
        }
    }

    class Program
    {
        // Manual bubble sort for Ułamek array
        static void BubbleSort(Ułamek[] array)
        {
            int n = array.Length;
            bool swapped;

            for (int i = 0; i < n - 1; i++)
            {
                swapped = false;

                for (int j = 0; j < n - i - 1; j++)
```

```

        {
            if (array[j].CompareTo(array[j + 1]) > 0)
            {
                // Swap
                Ułamek temp = array[j];
                array[j] = array[j + 1];
                array[j + 1] = temp;

                swapped = true;
            }
        }

        if (!swapped)
            break;
    }
}

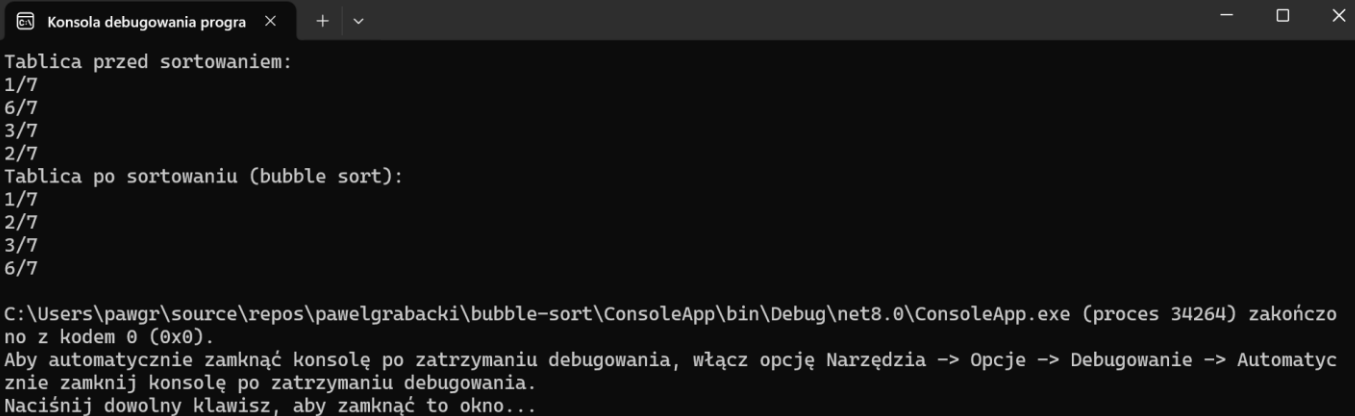
static void Main(string[] args)
{
    Ułamek[] tablica = {
        new Ułamek(1, 7),
        new Ułamek(6, 7),
        new Ułamek(3, 7),
        new Ułamek(2, 7)
    };

    Console.WriteLine("Tablica przed sortowaniem:");
    foreach (Ułamek u in tablica)
    {
        Console.WriteLine(u);
    }

    // Use bubble sort instead of Array.Sort
    BubbleSort(tablica);

    Console.WriteLine("Tablica po sortowaniu (bubble sort):");
    foreach (Ułamek u in tablica)
    {
        Console.WriteLine(u);
    }
}
}
}

```



```

Konsola debugowania progra
Tablica przed sortowaniem:
1/7
6/7
3/7
2/7
Tablica po sortowaniu (bubble sort):
1/7
2/7
3/7
6/7

C:\Users\pawgr\source\repos\pawelgrabacki\bubble-sort\ConsoleApp\bin\Debug\net8.0\ConsoleApp.exe (proces 34264) zakończył z kodem 0 (0x0).
Aby automatycznie zamknąć konsolę po zatrzymaniu debugowania, włącz opcję Narzędzia -> Opcje -> Debugowanie -> Automatycznie zamknij konsolę po zatrzymaniu debugowania.
Naciśnij dowolny klawisz, aby zamknąć to okno...

```