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

|  |
| --- |
| 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;      }  } |

**1. CEL ĆWICZENIA**  
  
Celem ćwiczenia było zapoznanie się z implementacją interfejsu IComparable<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 IComparable<Ulamek>**

|  |
| --- |
| public int CompareTo(Ulamek other) |

Interfejs służy do definiowania sposobu porównywania obiektów danego typu. Wymusza implementację metody:  
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(n²)
* Najlepszy przypadek (gdy dane są już posortowane): O(n)

**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 Ulamek, zawierającą pola: licznik, mianownik, konstruktor, metody pomocnicze oraz przeciążone operatory > i <.
2. W klasie Ulamek zaimplementowano interfejs IComparable<Ulamek> poprzez zdefiniowanie metody CompareTo(), która porównuje dwa ułamki wykorzystując porównanie przekątne (a/b vs c/d → ad vs cb).
3. Utworzono tablicę obiektów Ulamek i wypełniono ją kilkoma przykładowymi wartościami.
4. Zaimplementowano metodę BubbleSort(Ulamek[] 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 IComparable<T> 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 sortował tablicę ułamków rosnąco według ich wartości dziesiętnych.

**5.KOD**

|  |
| --- |
|  |

**Obraz zawierający tekst, zrzut ekranu, oprogramowanie

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.**