**ПРАКТИЧНА РОБОТА № 7**

**«ШАБЛОНИ ФУНКЦІЙ ТА ПАРАМЕТРИЗОВАНІ ПІДПРОГРАМИ СОРТУВАННЯ»**

**Мета роботи:** Навчитися застосовувати різні шаблони функцій для вирішення задач при програмуванні мові С#.

using System;

namespace prac7\_console

{

class Car: ICloneable, IComparable

{

public int speed;

public string name;

public Car(int speed, string name)

{

this.speed = speed;

this.name = name;

}

public object Clone()

{

return new Car(speed, name);

}

public int CompareTo(object other)

{

Car otherCar = other as Car;

if (speed>otherCar.speed)

return 1;

else if(speed<otherCar.speed)

return -1;

else

return 0;

}

public static bool operator >(Car first, Car second)

{

return first.CompareTo(second)>0;

}

public static bool operator <(Car first, Car second)

{

return first.CompareTo(second) < 0;

}

public static bool operator >=(Car first, Car second)

{

return first.speed>=second.speed;

}

public static bool operator <=(Car first, Car second)

{

return first.speed <= second.speed;

}

public static bool operator ==(Car first, Car second)

{

return first.speed == second.speed;

}

public static bool operator !=(Car first, Car second)

{

return first.speed != second.speed;

}

public override string ToString()

{

return $"{name} - {speed}";

}

public override bool Equals(object obj)

{

return obj is Car car &&

speed == car.speed &&

name == car.name;

}

public override int GetHashCode()

{

return HashCode.Combine(speed, name);

}

}

internal class Program

{

public static void Sort<T>(T[] arr, bool direction = false) where T : IComparable

{

if (direction)

{

for (int i = 1; i < arr.Length; ++i)

{

T key = arr[i];

int j = i - 1;

while (j >= 0 && key.CompareTo(arr[j]) > 0)

{

arr[j + 1] = arr[j];

j = j - 1;

}

arr[j + 1] = key;

}

}

else

{

for (int i = 1; i < arr.Length; ++i)

{

T key = arr[i];

int j = i - 1;

while (j >= 0 && key.CompareTo(arr[j]) < 0)

{

arr[j + 1] = arr[j];

j = j - 1;

}

arr[j + 1] = key;

}

}

}

static void Main(string[] args)

{

int[] array = { 6, 8, 5, 4, 2, 9, 10, 8 };

Sort(array, true);

foreach (var item in array)

{

Console.WriteLine(item);

}

Console.WriteLine("Integer array\n");

double[] doubles = { 0.68, 8.3, 7.6, 3.8, 0.5 };

Sort(doubles);

foreach (var item in doubles)

{

Console.WriteLine(item);

}

Console.WriteLine("Double array\n");

Car[] cars = { new Car(120, "Volvo"), new Car(80, "Bugatti"), new Car(200, "Ferrari") };

Sort(cars);

foreach (var item in cars)

{

Console.WriteLine(item);

}

Console.WriteLine("Car array\n");

}

}

}

Клас Program має опис параметризованої підпрограми сортування public static void Sort<T> котра реалізовує інтерфейс Icomparable.

public static void Sort<T>(T[] arr, bool direction = false) where T : IComparable

{

if (direction)

{

for (int i = 1; i < arr.Length; ++i)

{

T key = arr[i];

int j = i - 1;

while (j >= 0 && key.CompareTo(arr[j]) > 0)

{

arr[j + 1] = arr[j];

j = j - 1;

}

arr[j + 1] = key;

}

}

else

{

for (int i = 1; i < arr.Length; ++i)

{

T key = arr[i];

int j = i - 1;

while (j >= 0 && key.CompareTo(arr[j]) < 0)

{

arr[j + 1] = arr[j];

j = j - 1;

}

arr[j + 1] = key;

}

}

}

В цьому класі знаходиться статичний метод Main, що ініціалізовує масиви, та викликає підпрограму сортування передаючи туди дані масиви. Реалізований вивід відсортованих масивів у консоль.

static void Main(string[] args)

{

int[] array = { 6, 8, 5, 4, 2, 9, 10, 8 };

Sort(array, true);

foreach (var item in array)

{

Console.WriteLine(item);

}

Console.WriteLine("Integer array\n");

double[] doubles = { 0.68, 8.3, 7.6, 3.8, 0.5 };

Sort(doubles);

foreach (var item in doubles)

{

Console.WriteLine(item);

}

Console.WriteLine("Double array\n");

Car[] cars = { new Car(120, "Volvo"), new Car(80, "Bugatti"), new Car(200, "Ferrari") };

Sort(cars);

foreach (var item in cars)

{

Console.WriteLine(item);

}

Console.WriteLine("Car array\n");

}

Клас Car об’єктів масиву, призначений для сортування. Має перевантажені операції порівняння <, <=, ==, >=, >, =.

class Car: ICloneable, IComparable

{

public int speed;

public string name;

public Car(int speed, string name)

{

this.speed = speed;

this.name = name;

}

public object Clone()

{

return new Car(speed, name);

}

public int CompareTo(object other)

{

Car otherCar = other as Car;

if (speed>otherCar.speed)

return 1;

else if(speed<otherCar.speed)

return -1;

else

return 0;

}

public static bool operator >(Car first, Car second)

{

return first.CompareTo(second)>0;

}

public static bool operator <(Car first, Car second)

{

return first.CompareTo(second) < 0;

}

public static bool operator >=(Car first, Car second)

{

return first.speed>=second.speed;

}

public static bool operator <=(Car first, Car second)

{

return first.speed <= second.speed;

}

public static bool operator ==(Car first, Car second)

{

return first.speed == second.speed;

}

public static bool operator !=(Car first, Car second)

{

return first.speed != second.speed;

}

public override string ToString()

{

return $"{name} - {speed}";

}

public override bool Equals(object obj)

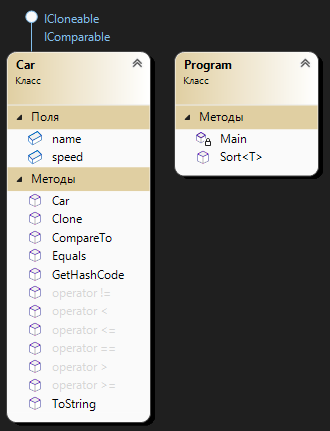
{

return obj is Car car &&

speed == car.speed &&

name == car.name;

}

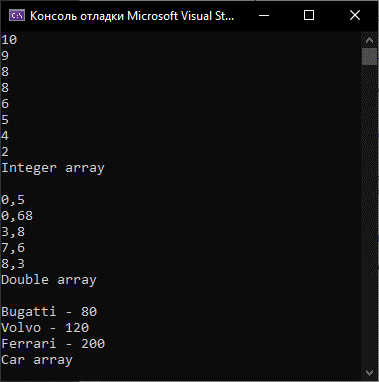
 public override int GetHashCode()

{

return HashCode.Combine(speed, name);

}

}



**Висновок:** ми навчилися застосовувати різні шаблони функцій для вирішення задач при програмуванні мові С#.