### Кам’янець-Подільський національний університет імені Івана Огієнка

### КАФЕДРА КОМП’ЮТЕРНИХ НАУК

### Навчальна дисципліна «Об’єктно-орієнтоване програмування»

### ЛАБОРАТОРНА РОБОТА #0110

## Тема:

Перевантаження операцій

Варіант №2

**Виконав**:  
студент 1-го курсу  
групи KNms1-B23  
Веретко С. І.

**Прийняла**:  
старший викладач,  
Мястковська М.О.

Кам’янець-Подільський – 2023

1. Теоретичні відомості

Мова C# дозволяє визначити значення операції щодо створюваного класу. Цей процес називається перевантаженням операцій. Перенавантажуючи операцію, ви розширюєте її використання для класу.

## Основи перевантаження операцій

Перевантаження операцій тісно пов'язане з перевантаженням методів. Для перевантаження операцій використовується ключове слово *operator*, що дозволяє створити *операційний метод*, який визначає дію операції, пов'язану з її класом.

Існує дві форми методів *operator*: одна використовується для унарних операцій, а інша – для бінарних.

//загальний формат перевантаження для унарної операції

public static тип\_повернення operator op( тип\_параметра операнд){ //тіло\_метода

}

// загальний формат перевантаження для бінарної операції

public static тип\_повернення operator op(

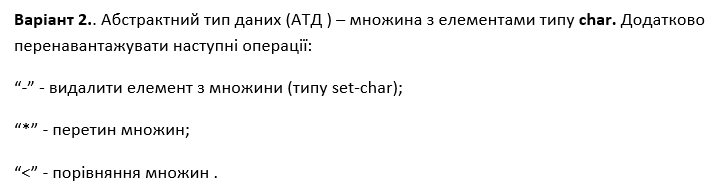
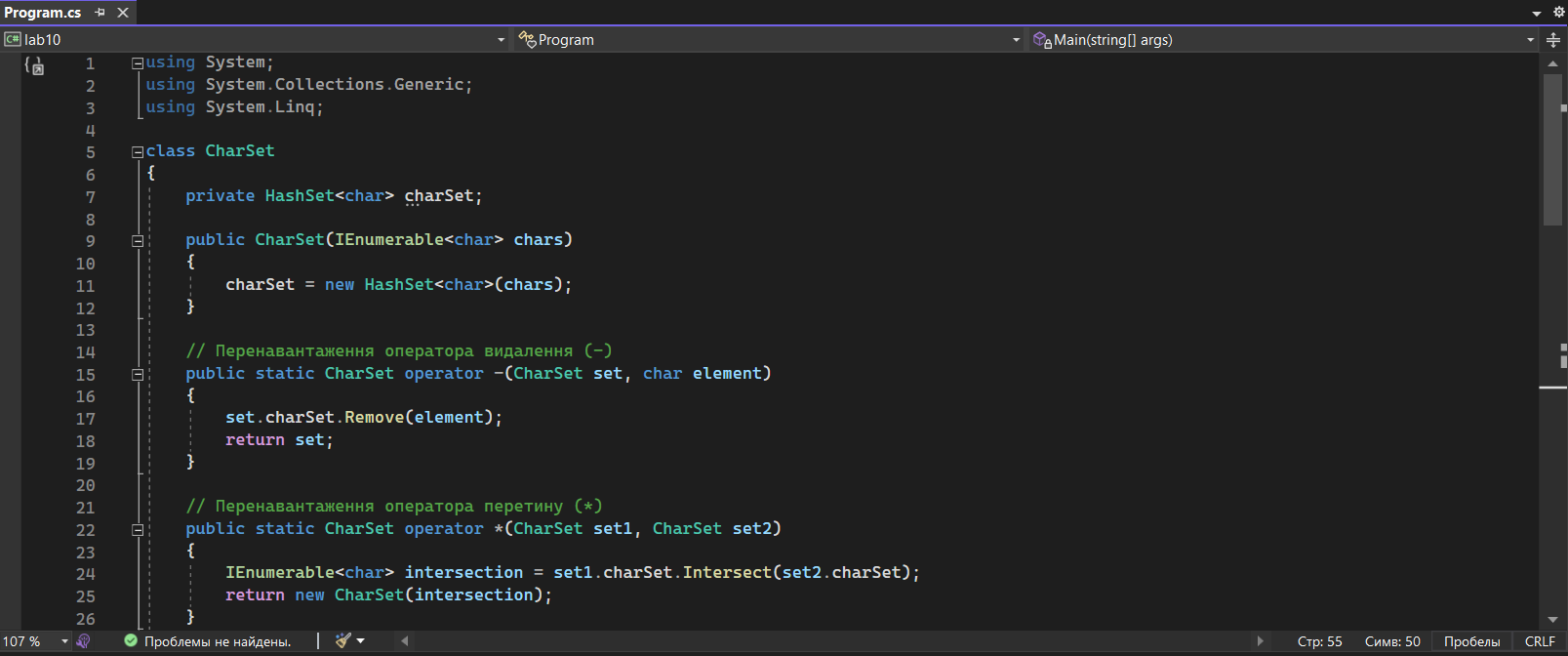
тип\_параметра1 операнд1, тип\_параметра2 операнд2) {

//тіло\_метода

}

Тут елемент *op* – це операція (наприклад “+” або “/”), яка перевантажується. Елемент *тип\_повернення* – це тип значення, що повертається при виконанні заданої операції. Для унарних операцій тип операнда повинен співпадати з класом, для якого визначена операція. Що стосується бінарних операцій, то тип хоч би одного операнда повинен співпадати з відповідним класом. Таким чином, C#-операції не можна перенавантажувати для класів, не створених вами. Наприклад, ви не можете перенавантажувати операцію “+” для типів int або string.

2. Завдання для виконання

  
  
3. Послідовність виконання завданнь  


4. Лістинг програми

using System;

using System.Collections.Generic;

using System.Linq;

class CharSet

{

private HashSet<char> charSet;

public CharSet(IEnumerable<char> chars)

{

charSet = new HashSet<char>(chars);

}

// Перенавантаження оператора видалення (-)

public static CharSet operator -(CharSet set, char element)

{

set.charSet.Remove(element);

return set;

}

// Перенавантаження оператора перетину (\*)

public static CharSet operator \*(CharSet set1, CharSet set2)

{

IEnumerable<char> intersection = set1.charSet.Intersect(set2.charSet);

return new CharSet(intersection);

}

// Перенавантаження оператора порівняння (<)

public static bool operator <(CharSet set1, CharSet set2)

{

return set1.charSet.IsProperSubsetOf(set2.charSet);

}

// Перенавантаження оператора порівняння (>)

public static bool operator >(CharSet set1, CharSet set2)

{

return set1.charSet.IsProperSupersetOf(set2.charSet);

}

public override string ToString()

{

return "{" + string.Join(", ", charSet) + "}";

}

}

class Program

{

static void Main(string[] args)

{

CharSet set1 = new CharSet(new char[] { 'a', 'b', 'c', 'd' });

CharSet set2 = new CharSet(new char[] { 'c', 'd', 'e', 'f' });

// Використання перевантажених операторів

CharSet resultDifference = set1 - 'b';

CharSet resultIntersection = set1 \* set2;

bool resultSubset = set1 < set2;

bool resultSuperset = set1 > set2;

Console.WriteLine("Set 1: " + set1);

Console.WriteLine("Set 2: " + set2);

Console.WriteLine("Difference of Set 1 after removing 'b': " + resultDifference);

Console.WriteLine("Intersection of Set 1 and Set 2: " + resultIntersection);

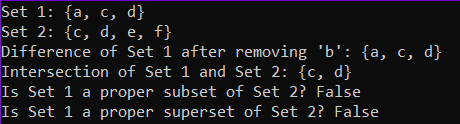
Console.WriteLine("Is Set 1 a proper subset of Set 2? " + resultSubset);

Console.WriteLine("Is Set 1 a proper superset of Set 2? " + resultSuperset);

}

}

5. Результат роботи програми



6. [xanax4rl/lab (github.com)](https://github.com/xanax4rl/lab)