Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра Информатики

Дисциплина «Программирование»

**ОТЧЕТ**

к лабораторной работе №7

на тему:

**«ПЕРЕГРУЗКА ОПЕРАТОРОВ»**

БГУИР 6-05-0612-02 36

|  |
| --- |
| Выполнил студент группы 353505  ДАНИЛОВ Дмитрий Игоревич |
|  |
| (дата, подпись студента) |
| Проверил ассистент каф. Информатики  РОМАНЮК Максим Валерьевич |
|  |
| (дата, подпись преподавателя) |

Минск 2024

# 1 Индивидуальное задание

Задание 1. Вариант 8. Класс – множество символов. В качестве поля – массив символов. Добавить метод, проверяющий корректность множества (нет повторений) – нужен сеттеру. Перегрузить +, -, \* (пересечение). Сравнить на == и !=. (d) если множество пустое, оно false. Преобразовать в String – неявно, назад – явно.

1. Спроектировать класс согласно варианту индивидуального задания. Для класса использовать отдельный модуль.
2. Спроектировать конструкторы и свойства с контролем корректности вводимых значений.
3. Перегрузить метод toString()
4. Добавить индексирование для получения полей класса
5. Перегрузить операции: a) математические (имеющие смысл для объектов класса) b) инкремент и декремент (изменить поля на 1) c) отношения (==, !=, ) d) true и false e) преобразования типа В методе main()
6. Создать несколько объектов класса. Продемонстрировать использование конструкторов и свойств.
7. Продемонстрировать работу всех методов и операций.

# 2 Выполнение работы

Для выполнения вашей лабораторной работы была создана библиотека классов, в которой был реализован класс «CharSet», представляющий собой множество символов. В этом классе содержится одно поле - массив символов. Для этого поля было создано свойство. Аксессор get описывается стандартно, а аксессор set имеет дополнительную реализацию. Он проверяет, что присваиваемое значение не содержит повторяющихся символов, и только тогда присваивает новое значение в поле. В ином случае генерируется исключение.

Был разработан метод ToString, который возвращает множество символов в виде строки.

После этого были переопределены следующие операторы: +, -, \*, ==, !=, true, false. Эти операторы переопределяются для двух объектов класса.

Также были реализованы операции явного приведения «CharSet» в string и неявное из string в «CharSet». Результат работы программы представлен на рисунке (рисунок 1).

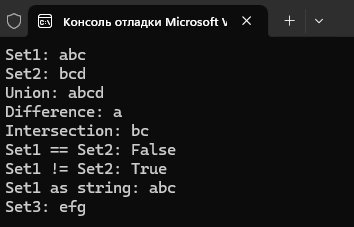


Рисунок 1 – Демонстрация работы программы

using System;

using System.Linq;

using System.Collections.Generic;

public class CharSet

{

private char[] \_chars;

public CharSet(char[] chars)

{

if (chars.Distinct().Count() != chars.Length)

throw new ArgumentException("Множество символов не должно содержать повторяющихся символов.");

\_chars = chars;

}

public char this[int index]

{

get { return \_chars[index]; }

set

{

if (\_chars.Contains(value))

throw new ArgumentException("Этот символ уже присутствует в множестве.");

\_chars[index] = value;

}

}

public override string ToString()

{

return new string(\_chars);

}

public static CharSet operator +(CharSet a, CharSet b)

{

return new CharSet(a.\_chars.Union(b.\_chars).ToArray());

}

public static CharSet operator -(CharSet a, CharSet b)

{

return new CharSet(a.\_chars.Except(b.\_chars).ToArray());

}

public static CharSet operator \*(CharSet a, CharSet b)

{

return new CharSet(a.\_chars.Intersect(b.\_chars).ToArray());

}

public static bool operator ==(CharSet a, CharSet b)

{

return a.\_chars.SequenceEqual(b.\_chars);

}

public static bool operator !=(CharSet a, CharSet b)

{

return !(a == b);

}

public static bool operator true(CharSet a)

{

return a.\_chars.Length > 0;

}

public static bool operator false(CharSet a)

{

return a.\_chars.Length == 0;

}

public static implicit operator string(CharSet a)

{

return a.ToString();

}

public static explicit operator CharSet(string s)

{

return new CharSet(s.ToCharArray());

}

}

class Program

{

static void Main()

{

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

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

Console.WriteLine("Set1: " + set1);

Console.WriteLine("Set2: " + set2);

CharSet union = set1 + set2;

Console.WriteLine("Union: " + union);

CharSet difference = set1 - set2;

Console.WriteLine("Difference: " + difference);

CharSet intersection = set1 \* set2;

Console.WriteLine("Intersection: " + intersection);

Console.WriteLine("Set1 == Set2: " + (set1 == set2));

Console.WriteLine("Set1 != Set2: " + (set1 != set2));

string set1AsString = set1;

Console.WriteLine("Set1 as string: " + set1AsString);

CharSet set3 = (CharSet)"efg";

Console.WriteLine("Set3: " + set3);

}

}

# Вывод

В ходе лабораторной работы мною была изучена одна из парадигм ООП, а именно полиморфизм, суть которого заключается в возможности переопределения существующей реализации методов как внутри класса, так и в классах наследниках. Было укреплено понимание свойств и авто свойств. Было на практике отработано переопределение методов и операторов. Была отработана генерация собственных исключений, которая помогает дополнять логику проверки данных получаемых и обрабатываемых программой. Так же была изучена схема создания библиотек классов, которая позволяет подключать ранее разработанные классы к проекту.