**Занятие № 16**

**Дата выполнения работы:** 08.05.2023

# **Тема работы:** «Процедуры и функции. Перегрузка функций (методов) и операторов»

**Ход работы**

**Задание 1**

1) Создать заданный в варианте класс. Определить в классе необходимые

методы, конструкторы, индексаторы и заданные перегруженные операции.

Написать программу тестирования, в которой проверяется использование

перегруженных операций.

2) Добавьте в свой класс вложенный объект Owner, который содержит Id, имя

и организацию создателя. Проинициализируйте его

3) Добавьте в свой класс вложенный класс Date (дата создания).

Проинициализируйте

4) Создайте статический класс StatisticOperation, содержащий 3 метода для

работы с вашим классом (по варианту п.1): сумма, разница между

максимальным и минимальным, подсчет количества элементов.

5) Добавьте к классу StatisticOperation методы расширения для типа string и

вашего типа из заданияNo1. См. задание по вариантам.

Вариант 1

Класс Одномерный массив. Дополнительно перегрузить

следующие операции: \* умножение массивов; true истина

если массив не сдержит отрицательных элементов, int()

операция приведения – возвращает размер массива; ==

проверка на равенство; < сравнение. Методы расширения:

1) Проверка на содержание определённого символа в

строке

2) Удаление отрицательных элементов

**Листинг программы:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace App

{

public partial class Form1 : Form

{

private OneDimensionalArray array;

public Form1()

{

InitializeComponent();

array = new OneDimensionalArray();

}

private void button1\_Click(object sender, EventArgs e)

{

int value;

if (int.TryParse(elementTextBox.Text, out value))

{

array.Add(value);

elementTextBox.Clear();

UpdateArrayInfo();

}

else

{

MessageBox.Show("Invalid element value. Please enter an integer.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void UpdateArrayInfo()

{

arraySizeLabel.Text = $"Size: {array.Size}";

arrayContentsLabel.Text = $"Contents: {array}";

}

private void multiplyButton\_Click(object sender, EventArgs e)

{

int value;

if (int.TryParse(multiplierTextBox.Text, out value))

{

OneDimensionalArray multipliedArray = array \* value;

resultLabel.Text = multipliedArray.ToString();

}

else

{

MessageBox.Show("Invalid multiplier value. Please enter an integer.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void containsButton\_Click(object sender, EventArgs e)

{

char character = characterTextBox.Text.FirstOrDefault();

bool contains = array.ContainsCharacter(character);

resultLabel.Text = contains ? "Array contains the specified character." : "Array does not contain the specified character.";

}

private void removeNegativeButton\_Click(object sender, EventArgs e)

{

array.RemoveNegativeElements();

UpdateArrayInfo();

}

}

public class OneDimensionalArray

{

private int[] array;

public int Size => array.Length;

public int this[int index]

{

get { return array[index]; }

set { array[index] = value; }

}

public OneDimensionalArray()

{

array = new int[0];

}

public void Add(int element)

{

Array.Resize(ref array, Size + 1);

array[Size - 1] = element;

}

public static OneDimensionalArray operator \*(OneDimensionalArray array, int multiplier)

{

OneDimensionalArray result = new OneDimensionalArray();

for (int i = 0; i < array.Size; i++)

{

result.Add(array[i] \* multiplier);

}

return result;

}

public override bool Equals(object obj)

{

if (obj is OneDimensionalArray otherArray)

{

if (Size != otherArray.Size)

return false;

for (int i = 0; i < Size; i++)

{

if (array[i] != otherArray[i])

return false;

}

return true;

}

return false;

}

public override int GetHashCode()

{

return array.GetHashCode();

}

public bool ContainsCharacter(char character)

{

foreach (int element in array)

{

if (element.ToString().Contains(character.ToString()))

return true;

}

return false;

}

public void RemoveNegativeElements()

{

array = array.Where(x => x >= 0).ToArray();

}

public override string ToString()

{

return string.Join(", ", array);

}

}

public static class StringExtensions

{

public static bool ContainsCharacter(this string str, char character)

{

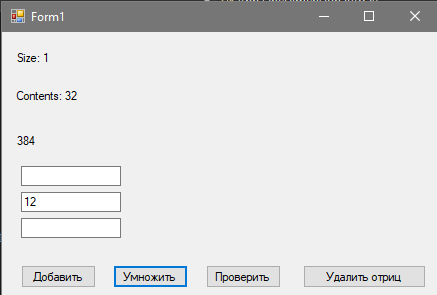
return str.Contains(character);

}

}

}

**Результат:**

****

**Вывод:** Процедуры и функции - основные блоки программирования для выполнения задач. Перегрузка функций и методов позволяет определить их с разными параметрами. Перегрузка операторов позволяет определить их поведение для пользовательских типов данных. Это делает код более гибким и удобочитаемым.