

Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний
інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 3
з дисципліни «Основи програмування – 2.
Методології програмування»

«Перевантаження операторів»

Варіант 13

Виконав студент ПІ-13 Жмайло Дмитро Олександрович
(шифр, прізвище, ім'я, по батькові)

Перевірив Вечерковська Анастасія Сергіївна
(прізвище, ім'я, по батькові)

Київ 2022

Лабораторна робота 3
Перевантаження операторів
Варіант 13

Визначити клас "Число", членами якого є його розряди (кількість одиниць, десятків, сотень та тисяч). Реалізувати для нього декілька конструкторів, геттери, метод обчислення числа (отримання десяткового еквівалента).
Перевантажити оператори: префіксний "++" / префіксний "--" - для інкрементування / декрементування усіх розрядів числа, "+" - для додавання двох чисел, , заданих своїми розрядами, ">" - для знаходження більшого із двох таких чисел. Створити чотири числа (N1, N2, N3, N4), використовуючи різні конструктори. Інкрементувати число N1, а число N2 декрементувати. Знайти суму змінених чисел N1 та N2 і зберегти її в N3. Знайти більше із чисел N3 і N4. Отримати його десятковий еквівалент.

Код програми

C#

Program.cs

```
using System;

namespace Lab3CSharp
{
    class Program
    {
        static void Main(string[] args)
        {
            Number N1 = new Number();

            Console.WriteLine("Second number: ");
            int decNumber = Operations.InputInt("decimal number", 0, 10000);
            Number N2 = new Number(decNumber);

            Number N3 = new Number(N1);

            Console.WriteLine("\nFourth number: ");
            int thousands = Operations.InputInt("number of thousands", 0, 10);
            int hundreds = Operations.InputInt("number of hundreds", 0, 10);
            int tens = Operations.InputInt("number of tens", 0, 10);
            int ones = Operations.InputInt("number of ones", 0, 10);
            Number N4 = new Number(ones, tens, hundreds, thousands);

            Operations.ShowNumber(N1, "\nFirst number (rand) is: ");
            Operations.ShowNumber(N2, "Second number (dec) is: ");
            Operations.ShowNumber(N3, "Third number (copy) is: ");
            Operations.ShowNumber(N4, "Fourth number (hand) is:");

            ++N1;
            --N2;

            Operations.ShowNumber(N1, "\nIncremented first number is: ");
            Operations.ShowNumber(N2, "Decrement second number is: ");
        }
    }
}
```

```
N3 = N1 + N2;

Operations.ShowNumber(N3, "\nNew third number (N1+N2) is: ");

if (N3 > N4)
{
    Console.WriteLine($"N3 is bigger than N4
({N3.ReturnDecimalEquivalent()} > {N4.ReturnDecimalEquivalent()})");
    Console.WriteLine("N3 dec is: " + N3.ReturnDecimalEquivalent());
}
else
{
    Console.WriteLine($"N4 is bigger than N3
({N4.ReturnDecimalEquivalent()} > {N3.ReturnDecimalEquivalent()})");
    Console.WriteLine("N4 dec is: " + N4.ReturnDecimalEquivalent());
}

Console.ReadKey();
}
}
```

Number.cs

```
using System;

using System.Collections.Generic;

using System.Text;

namespace Lab3CSharp
{
    class Number
    {
        private int ones;
        private int tens;
        private int hundreds;
        private int thousands;

        public Number(int ones, int tens, int hundreds, int thousands)
        {
            this.ones = ones;
            this.tens = tens;
            this.hundreds = hundreds;
            this.thousands = thousands;
        }

        public Number()
        {
            Random rand = new Random();
            this.ones = rand.Next(10);
            this.tens = rand.Next(10);
            this.hundreds = rand.Next(10);
            this.thousands = rand.Next(10);
        }

        public Number(int DecNumber)
        {
            this.ones = DecNumber % 10;
            this.tens = DecNumber % 100 / 10;
            this.hundreds = DecNumber % 1000 / 100;
        }
    }
}
```

```
        this.thousands = DecNumber / 1000;
    }

    public Number(Number number)
    {
        this.ones = number.GetOnes();
        this.tens = number.GetTens();
        this.hundreds = number.GetHundreds();
        this.thousands = number.GetThousands();
    }

    public int GetOnes()
    {
        return ones;
    }

    public int GetTens()
    {
        return tens;
    }

    public int GetHundreds()
    {
        return hundreds;
    }

    public int GetThousands()
    {
        return thousands;
    }

    public int ReturnDecimalEquivalent()
    {
        int result = thousands * 1000 + hundreds * 100 + tens * 10 + ones;
        return result;
    }
}
```

```

public static Number operator +(Number a, Number b)
{
    int cOnes = a.GetOnes() + b.GetOnes();
    int cTens = a.GetTens() + b.GetTens();
    int cHundreds = a.GetHundreds() + b.GetHundreds();
    int cThousands = a.GetThousands() + b.GetThousands();
    Number c = new Number(cOnes, cTens, cHundreds, cThousands);
    c.UpdateNumberToItsBoundaries();
    return c;
}

public static Number operator ++(Number a)
{
    a.ones++;
    a.tens++;
    a.hundreds++;
    a.thousands++;
    a.UpdateNumberToItsBoundaries();
    return a;
}

public static Number operator --(Number a)
{
    a.ones--;
    a.tens--;
    a.hundreds--;
    a.thousands--;
    a.UpdateNumberToItsBoundaries();
    return a;
}

public static bool operator <(Number a, Number b)
{
    if (a.ReturnDecimalEquivalent() < b.ReturnDecimalEquivalent())
    {

```

```
        return true;
    }
    return false;
}
```

```
public static bool operator >(Number a, Number b)
{
    if (a.ReturnDecimalEquivalent() > b.ReturnDecimalEquivalent())
    {
        return true;
    }
    return false;
}
```

```
public void UpdateNumberToItsBoundaries()
{
    if (ones < 0)
    {
        ones = 0;
    }
    if (tens < 0)
    {
        tens = 0;
    }
    if (hundreds < 0)
    {
        hundreds = 0;
    }
    if (thousands < 0)
    {
        thousands = 0;
    }

    if (ones > 9)
    {
```



```
        tens += ones / 10;
        ones %= 10;
    }
    if (tens > 9)
    {
        hundreds += tens / 10;
        tens %= 10;
    }
    if (hundreds > 9)
    {
        thousands += hundreds / 10;
        hundreds %= 10;
    }
    if (thousands > 9)
    {
        ones = 9;
        tens = 9;
        hundreds = 9;
        thousands = 9;
    }
}

}
```

Operations.cs

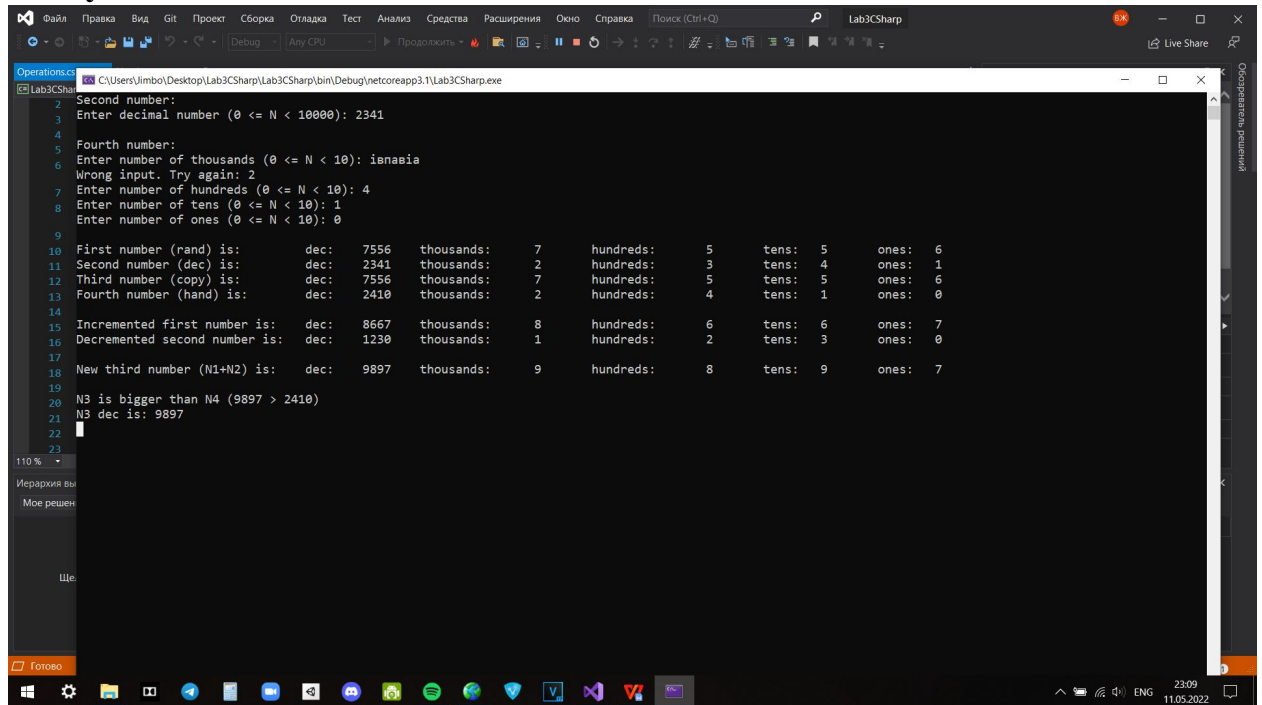
```
using System;
using System.Collections.Generic;
using System.Text;

namespace Lab3CSharp
{
    class Operations
    {
        public static int InputInt(string message, int minLimit, int maxLimit)
        {
            Console.Write($"Enter {message} ({minLimit} <= N < {maxLimit}): ");
            while (true)
            {
                string input = Console.ReadLine();
                if (int.TryParse(input, out int number) && number >= minLimit && number <
maxLimit)
                {
                    return number;
                }
                else
                {
                    Console.Write("Wrong input. Try again: ");
                }
            }
        }

        public static void ShowNumber(Number number, string message)
        {
            string result = message + "\tdec: \t" +
Convert.ToString(number.ReturnDecimalEquivalent()) + "\tthousands: \t" +
Convert.ToString(number.GetThousands()) + "\thundreds: \t" +
Convert.ToString(number.GetHundreds()) + "\ttens: \t" + Convert.ToString(number.GetTens())
+ "\tones: \t" + Convert.ToString(number.GetOnes());

            Console.WriteLine(result);
        }
    }
}
```

Тестування:



```
Operations: C:\Users\Jimbo\Desktop\Lab3CSharp\Lab3CSharp\bin\Debug\netcoreapp3.1\Lab3CSharp.exe
2 Second number:
3 Enter decimal number (0 <= N < 10000): 2341
4
5 Fourth number:
6 Enter number of thousands (0 <= N < 10): isnasia
7 Wrong input. Try again: 2
8 Enter number of hundreds (0 <= N < 10): 4
9 Enter number of tens (0 <= N < 10): 1
10 Enter number of ones (0 <= N < 10): 0
11
12 First number (rand) is: dec: 7556 thousands: 7 hundreds: 5 tens: 5 ones: 6
13 Second number (dec) is: dec: 2341 thousands: 2 hundreds: 3 tens: 4 ones: 1
14 Third number (copy) is: dec: 7556 thousands: 7 hundreds: 5 tens: 5 ones: 6
15 Fourth number (hand) is: dec: 2410 thousands: 2 hundreds: 4 tens: 1 ones: 0
16
17 Incremented first number is: dec: 8667 thousands: 8 hundreds: 6 tens: 6 ones: 7
18 Decremented second number is: dec: 1230 thousands: 1 hundreds: 2 tens: 3 ones: 0
19
20 New third number (N1+N2) is: dec: 9897 thousands: 9 hundreds: 8 tens: 9 ones: 7
21
22 N3 is bigger than N4 (9897 > 2410)
23 N3 dec is: 9897
```

Висновок: на цій лабораторній роботі я детальніше познайомився з переваженням операторів, реалізуючи програму на мові програмування C#; закріпив свої теоретичні знання щодо цього механізму на практиці та зробив висновки.