Отчет по практической работе №11

Наименование работы: механизм наследования. Отношения между классами.

Индивидуальные задания (вариант №9):

Описать базовый класс Строка в соответствии с вариантом 3.

Описать базовый класс Строка.

Обязательные поля класса:

* поле для хранения символов строки;
* значение типа ushort для хранения длины строки в байтах.

Реализовать обязательные методы следующего назначения:

* конструктор без параметров;
* конструктор, принимающий в качестве параметра строковый литерал;
* конструктор, принимающий в качестве параметра символ;
* метод получения длины строки;
* метод очистки строки (сделать строку пустой).

Описать производный от Строка класс Десятичная\_строка.

Строки данного класса могут содержать только символы десятичных цифр и символы - и +, задающие знак числа. Символы - или + могут находиться только в первой позиции числа, причем символ + может отсутствовать, в этом случае число считается положительным. Если в составе инициализирующей строки будут встречены любые символы, отличные от допустимых, класс Десятичная строка принимает нулевое значение. Содержимое данных строк рассматривается как десятичное число.

Для класса определить следующие методы:

* конструктор, принимающий в качестве параметра число;
* арифметическая разность строк;
* проверка на больше (по значению);
* проверка на меньше (по значению).

Ход работы:

1. Разработал программу в соответствии с заданием №1. Были реализованы методы, классы, наследование.

Листинг программы:

using System;

using System.Text;

public class Program

{

public static void Main(string[] args)

{

var s = new DecString(12);

var x = new DecString(8);

Console.WriteLine(s.bs);

DecString t = s - x;

Console.WriteLine(t.bs);

var l = new DecString(16);

var r = new DecString(20);

bool result = l > r;

if (result)

{

Console.WriteLine($"{l.bs} Больше {r.bs}");

}

else Console.WriteLine($"{l.bs} Меньше {r.bs}");

}

}

public class Str

{

public ushort dlina;

public string str;

// конструктор без параметро

public Str()

{

}

// конструктор, принимающий в качестве параметра строковый литерал

public Str(string str)

{

this.str = str;

dlina = (ushort)str.Length;

}

// конструктор, принимающий в качестве параметра символ;

public Str(char ch)

{

str = Convert.ToString(ch);

dlina = 1;

}

//Метод, возвращающий длину строки

public ushort Leng(string str)

{

return dlina = Convert.ToUInt16(str.Length);

}

// Метод очищающий стоку.

public void Clear()

{

str = "";

dlina = 0;

}

}

public class DecString : Str

{

// Конструктор для десятичной строки.

public DecString(int x)

{

var sb = new StringBuilder();

if (x < 0)

{

sb.Append('-');

}

x = Math.Abs(x);

while (x > 0)

{

int temp = x % 10;

sb.Insert(0, temp.ToString());

x /= 10;

}

bs = sb.ToString();

}

public string bs { get; set; }

public static DecString operator -(DecString l, DecString r)

{

int result = Int32.Parse(l.bs) - Int32.Parse(r.bs);

return new DecString(result);

}

public static bool operator <(DecString l, DecString r)

{

int li = Int32.Parse(l.bs);

int ri = Int32.Parse(r.bs);

return li < ri;

}

public static bool operator >(DecString l, DecString r)

{

int li = Int32.Parse(l.bs);

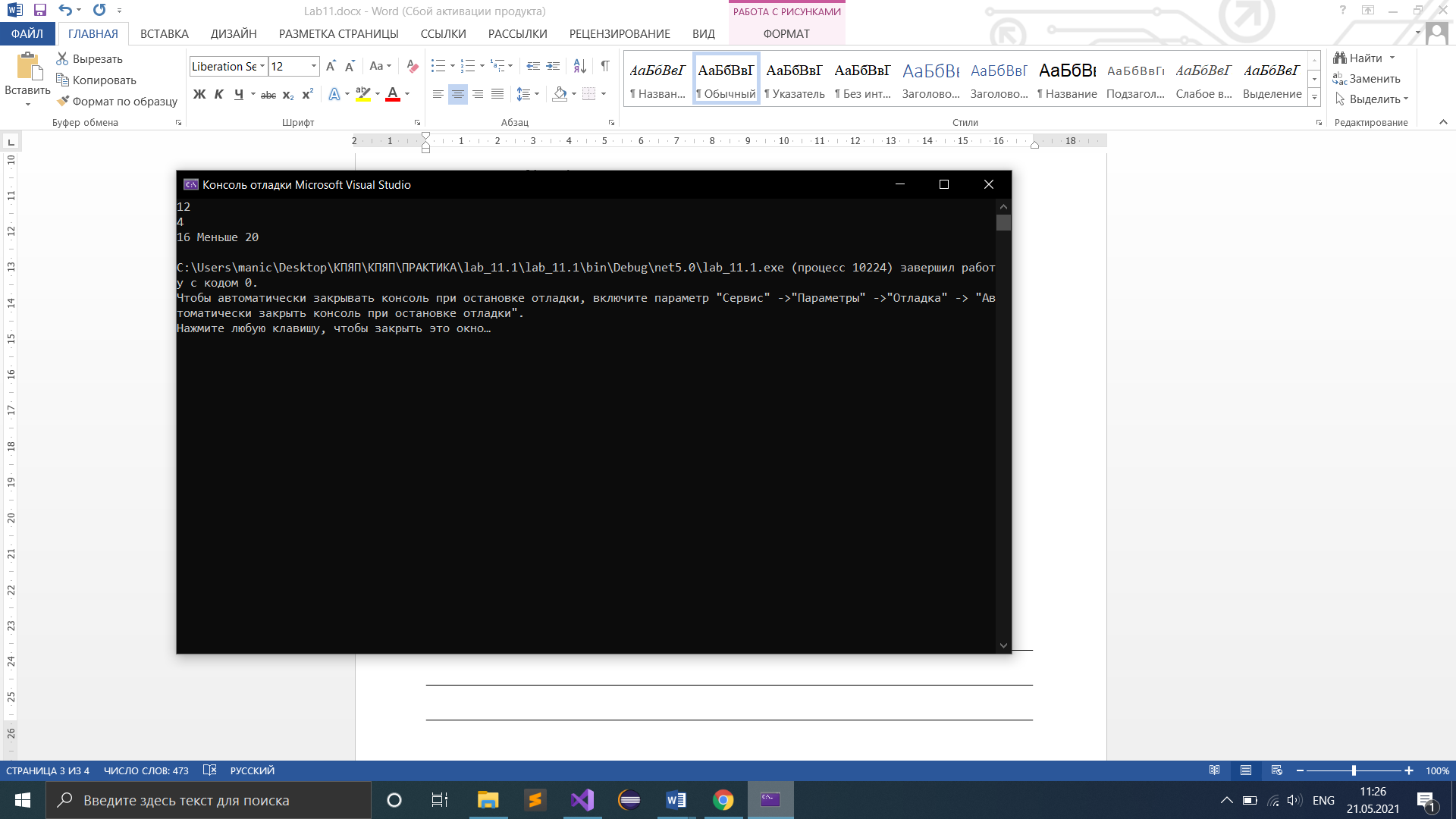
int ri = Int32.Parse(r.bs);

return li > ri;

}

}

Результат выполнения программы Рис 1.1.



**Рисунок 1.1 – Пример работы программы c наследованием**

Отметка\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Подпись преподавателя\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_