Министерство образования и науки Российской Федерации
ФГАОУ ВО «КФУ им. В.И. Вернадского»
физико-технический институт (структурное подразделение)

Кафедра компьютерной инженерии и моделирования

Шенгелай Всеволод Михайлович

отчет по лабораторной работе №3
по дисциплине **«ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ»**

Направление подготовки:
09.03.04 "Программная инженерия"

Оценка - 90

Симферополь, 2021

Лабораторная работа №3

Тема: Описание и вызов методов

Цель работы: изучить на практике использование перегрузки и перекрытия (Override) методов, статические и виртуальные методы, научиться передават в методы простые типы по ссылке, передавать и возвращать из методов несколько значений, в том числе и неопределенное значение параметров.

Описание ключевых понятий:

Перегрузка – возможность использования одноимённых подпрограмм: процедур или функций в языках программирования;

перекрытие (Override) методов – Override используется в классе-потомке, чтобы указать что функция должна переопределять виртуальную функцию, объявленную в базовом классе. Это позволяет избавиться от ошибок, когда из-за опечатки вместо переопределения существующей виртуальной функции была создана новая (с другим именем или сигнатурой);

закрытые и открытые методы – методы, недоступные и доступные (соответственно) для использования вне класса или его потомка или в данной сборки;

статические и виртуальные методы – методы, не имеющие доступа к переменной this, т.е. не может взаимодействовать с полями данного класса, и методы, требующие обязательного определения в каждом потомке данного базового класса;

кортежи – тип неизменяемого контейнера;

P**arams** – ключевое слово, обозначающее, что следующие значения, переданные в функцию через ее объявление, будут приемлемы и учтены в контексте данной функции как массив переменных.

Перед выполнением лабораторной работы изучена следующая литература:

1. Презентация лектора курса: «Базовые понятия и принципы ООП в C#».
2. Шилдт Герберт. Полный справочник по C#. Пер. с англ. - М.: Издательский дом "Вильямс", 2008. - 752с.: ил. (главы 6,8,11).
3. Биллиг В.А. Основы объектного программирования на языке C# -М.: Интуит, 2009. (9,10 лекции)
4. Герберт Шилдт C# Полное руководство (главы 1-7).
5. Сайт MSDN Microsoft.
6. Сайт Metanit.com

Выполнены 3 задания, описанных в методических указания к выполнению лабораторных работ.

Задание 1: Создано WPF приложение с функционалом, описанным в методических указаниях.

В этом задании мы написали метод для нахождения наибольшего общего делителя двух целых чисел с помощью алгоритма Евклида (и визуальную часть программы). Также мы создали Юнит-тест, который помог автоматизировать задачу тестирования написанного нами метода.

```
ссылка: 1
public static int NOD (int a, int b) //Метод Евклида
{
    if (a == 0) return b;
    while (b != 0)
    {
        if (a > b)
        {
            a = a - b;
        }
        else
        {
            b = b - a;
        }
    }
    return a;
```

Рисунок 1. Алгоритм Евклида для нахождения НОД двух целых чисел

```
public class UnitTest1
{
    [TestMethod]
    Ссылок: 0
    public void NOD_2806and345_23returned()
    {
        //arrange
        int x = 2806;
        int y = 345;
        int expected = 23;

        //act
        Calculations c = new Calculations();
        int actual = c.NOD(x, y);

        //assert
        Assert.AreEqual(expected, actual);
    }
}
```

Рисунок 2. Код Юнит-теста для метода NOD

Задание 2: Модифицировано WPF приложение с функционалом, описанным в методических указаниях к первому заданию.

В этом задании мы перегружаем метод, реализующий алгоритм Евклида. Используя ключевое слово params, мы можем передавать неопределенное количество параметров в вызов метода. В новом, перегруженном методе NOD используется рекурсия, чтобы вычислить НОД для любого количества чисел.

```
// Перегружаем метод
// !!! Нельзя скармливать ему меньше, чем 2 числа.
// С 2-мя числами рекурсия работать не будет, но мы это предусмотрели
Ссылок: 3
public static int NOD(params int[] list)
{
    if (list.Length == 2)
    {
        return NOD(list[0], list[1]);
    }
    int temp = list[list.Length - 1];
    int[] new_list = new int[list.Length - 1];

    for (int i = 0; i < new_list.Length; i++)
        new_list[i] = list[i];

    int back = 0;

    if (new_list.Length == 2)
    {
        back = NOD(new_list[0], new_list[1]);
        return back;
    }

    back = NOD(new_list);
    temp = NOD(back, temp);

    return temp;
}
```

Рисунок 3. Перегруженный вариант метода NOD

Задание 3: Модифицировано WPF приложение с функционалом, описанным в методических указаниях к первому заданию.

В этом задании мы сравниваем эффективность двух алгоритмов поиска НОД нескольких чисел (перегруженные варианты поиска НОД двух чисел) – алгоритма Евклида и алгоритма Штейна. Для малых значений чисел (в пределах нескольких десятков) и их количества, алгоритм Штейна имеет незначительный выигрыш по времени. Но когда введено много чисел, их абсолютные значения приближаются к границам типа int – алгоритм Штейна значительно выигрывает. Для точного измерения прошедшего времени мы использовали класс Stopwatch.

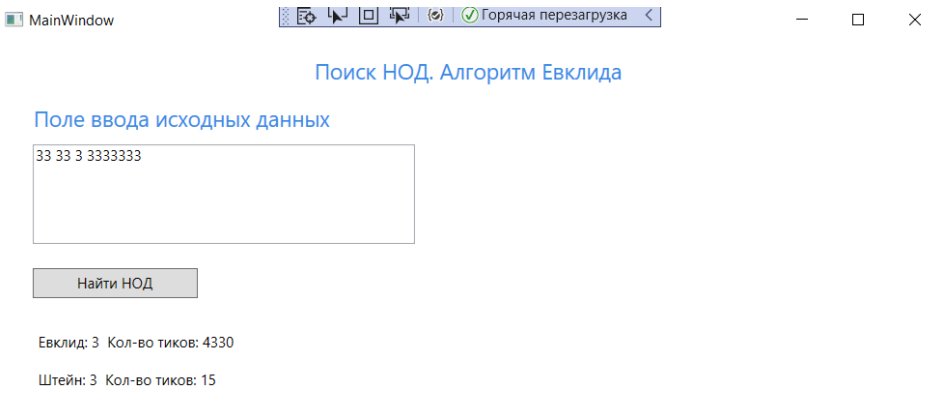

Рисунок 4. Графический интерфейс приложения для задания 3

Представлены 3 проекта, реализованных в Visual Studio Common Eddition 2019. Проекты представлены преподавателю в электронной форме, продемонстрирована их работоспособность, разъяснены детали программного кода.