

ГУАП

КАФЕДРА № 42

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ _____
ПРЕПОДАВАТЕЛЬ

Старший преподаватель		С.Ю. Гуков
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ

СИСТЕМЫ КОНТРОЛЯ ВЕРСИЙ (VCS)
Вариант 4

по курсу: ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №	4128		А.М. Деев
		подпись, дата	инициалы, фамилия

Санкт-Петербург 2023

1. Цель работы

Изучить предназначение и различные способы организаций систем контроля версий (Version Control System, VCS) Git. Познакомиться с операциями над файлами в репозитории и с приемами групповой работы над проектом.

2. Задание

Необходимо объединиться в команды по 2-3 человека. У каждого участника команды должен быть свой зарегистрированный аккаунт GitHub (логин). Один из участников команды создает репозиторий и присоединяет к нему остальных участников. Необходимо придумать общий интерфейс программы (один из участников делает коммит созданного интерфейса в репозиторий, остальные обновляют у себя локальную копию репозитория). Далее каждый из участников в своей отдельной ветке выполняет свое задание по варианту (задания в команде должны различаться), периодически делая коммиты своих классов и изменений в коде в репозиторий, при этом обновляя (дополняя) свой локальный проект кодом коллег по команде. После того как все участники команды сделают свое задание, ветки сливаются в главную ветку master, и оформляется файл README.md с пояснениями о выполненных заданиях

3. Описание разработки и технологии

Git — система управления версиями с распределенной архитектурой. В отличие от некогда популярных систем вроде CVS и Subversion (SVN), где полная история версий проекта доступна лишь в одном месте, в Git каждая рабочая копия кода сама по себе является репозиторием. Это позволяет всем разработчикам хранить историю изменений в полном объеме. Разработка в Git ориентирована на обеспечение высокой производительности, безопасности и гибкости распределенной системы.

В ходе работы в репозитории была создана ветка Master, в которую был добавлен общий интерфейс программы. После чего каждый член команды разработал код по индивидуальному заданию в собственной ветке. По выполнению задания было произведено слияние веток с веткой Master.

Варианты заданий:

4. Пользователь открывает файл с данными о ВВП и ВНП России за последние 15 лет. Вывести эту информацию на экран в удобном формате. Также по этим данным построить графики. Вычислить ежегодный процент роста либо падения.

8. Пользователь открывает файл с данными о медианной заработной плате в России за последние 15 лет. Вывести эту информацию на экран в удобном формате. Также по этим данным построить графики. Вычислить процент роста зарплат у мужчин и у женщин.

4. Примеры работы программы

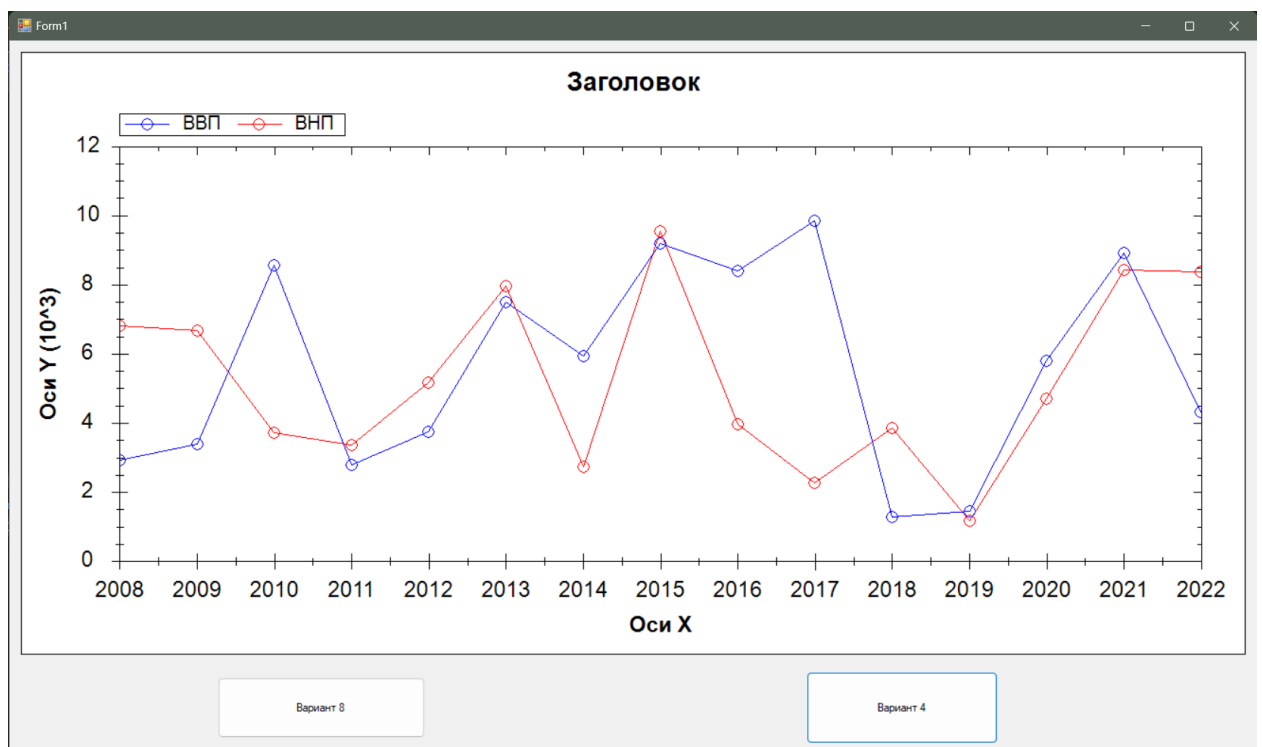


Рисунок 1 – форма с выполненным заданием варианта 4

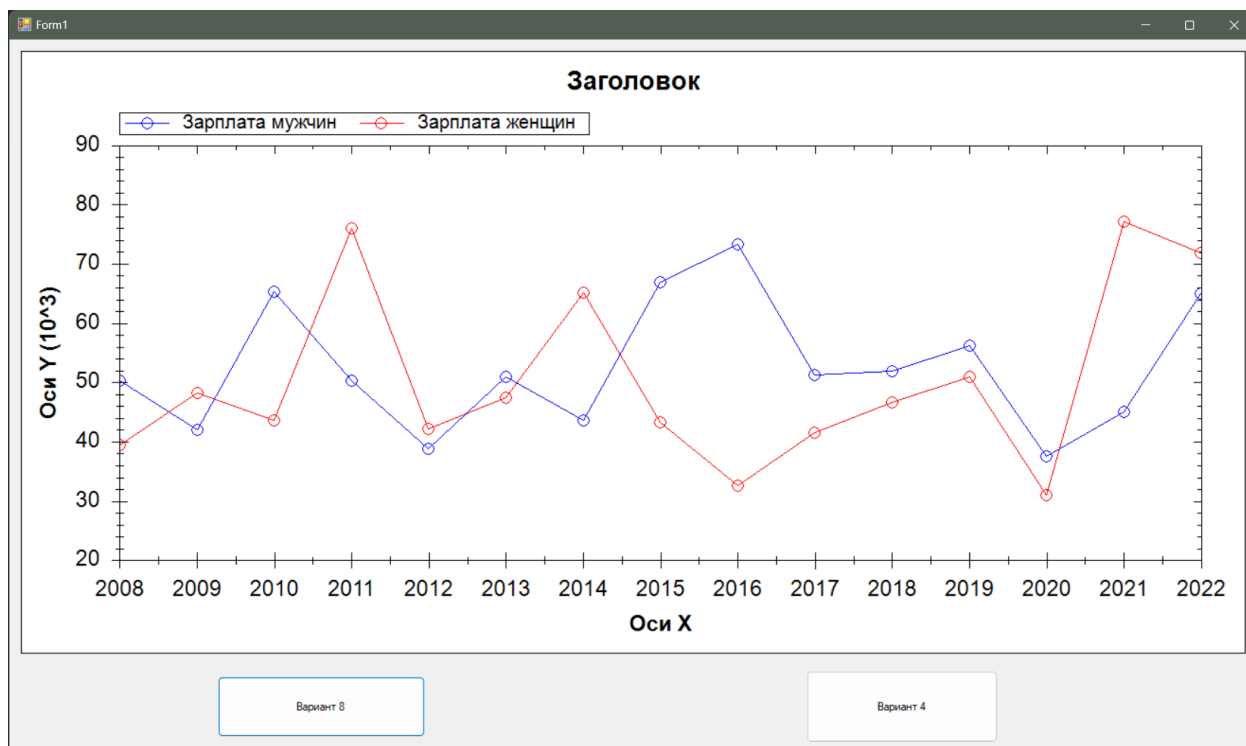


Рисунок 2 – форма с выполненным заданием варианта 8

```

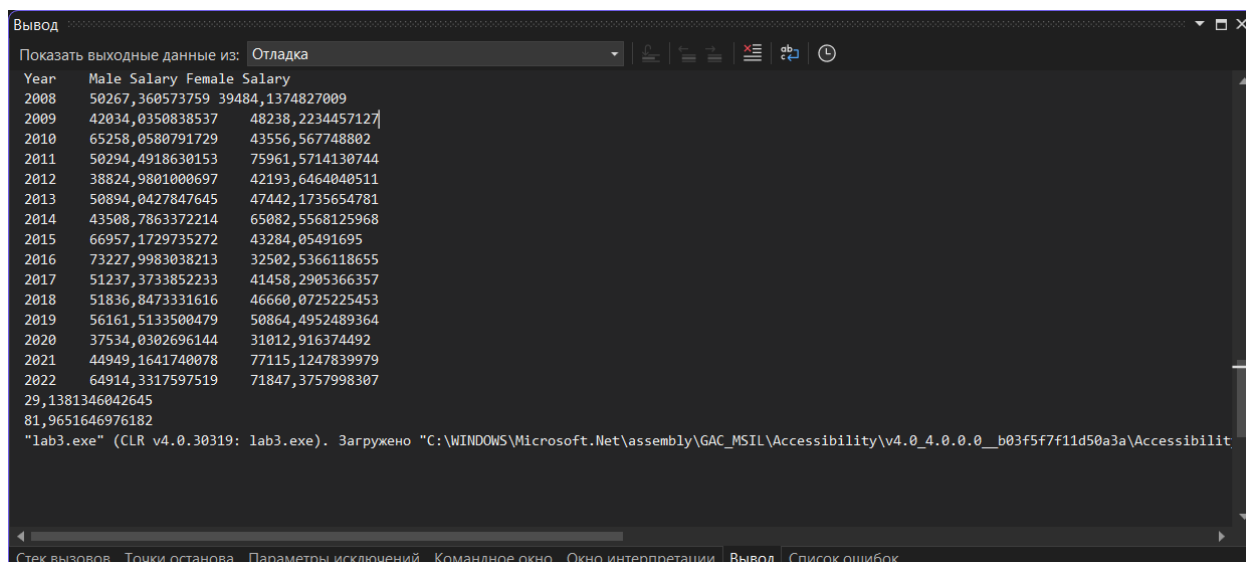
Вывод
Показать выходные данные из: Отладка

Вычисление процента роста/падения ВВП и ВВП:
Год 2009:
Процент роста/падения ВВП: 15,4529914529915%
Процент роста/падения ВВП: -1,91317144959529%
Год 2010:
Процент роста/падения ВВП: 153,568255848386%
Процент роста/падения ВВП: -44,4111027756939%
Год 2011:
Процент роста/падения ВВП: -67,6164895480556%
Процент роста/падения ВВП: -9,60863697705803%
Год 2012:
Процент роста/падения ВВП: 35,1604760187522%
Процент роста/падения ВВП: 54,493878769782%
Год 2013:
Процент роста/падения ВВП: 100,106723585912%
Процент роста/падения ВВП: 53,7495168148434%
Год 2014:
Процент роста/падения ВВП: -20,84%
Процент роста/падения ВВП: -65,757385292269%
Год 2015:
Процент роста/падения ВВП: 54,6235472460839%
Процент роста/падения ВВП: 250,03671071953%
Год 2016:
Процент роста/падения ВВП: -8,7037037037037%
Процент роста/падения ВВП: -58,3848977451494%
Год 2017:
Процент роста/падения ВВП: 17,3487650638349%
Процент роста/падения ВВП: -43,1451612903226%
Год 2018:
Процент роста/падения ВВП: -86,9547534316218%
Процент роста/падения ВВП: 70,8776595744681%
Год 2019:
Процент роста/падения ВВП: 11,53546375682%
Процент роста/падения ВВП: -69,9610894941634%
Год 2020:
Процент роста/падения ВВП: 305,380852550664%
Процент роста/падения ВВП: 306,390328151986%
Год 2021:
Процент роста/падения ВВП: 53,7665919669023%
Процент роста/падения ВВП: 78,8780280492988%
Год 2022:
Процент роста/падения ВВП: -51,6367713004484%
Процент роста/падения ВВП: -0,510810168686149%

Стек вызовов Точки останова Параметры исключений Командное окно Окно интерпретации Вывод Список ошибок

```

Рисунок 3 – вывод вычисленных значений роста/падения ВВП и ВВП по годам в консоль (вариант4)



```
Вывод
Показать выходные данные из: Отладка
Year Male Salary Female Salary
2008 50267,360573759 39484,1374827009
2009 42034,0350838537 48238,2234457127
2010 65258,0580791729 43556,567748802
2011 50294,4918630153 75961,5714130744
2012 38824,9801000697 42193,6464040511
2013 50894,0427847645 47442,1735654781
2014 43508,7863372214 65082,5568125968
2015 66957,1729735272 43284,05491695
2016 73227,9983038213 32502,5366118655
2017 51237,3733852233 41458,2905366357
2018 51836,8473331616 46660,0725225453
2019 56161,5133500479 50864,4952489364
2020 37534,0302696144 31012,916374492
2021 44949,1641740078 77115,1247839979
2022 64914,3317597519 71847,3757998307
29,1381346042645
81,9651646976182
"lab3.exe" (CLR v4.0.30319: lab3.exe). Загружено "C:\WINDOWS\Microsoft.Net\assembly\GAC_MSIL\Accessibility\v4.0.0.0__b03f5f7f11d50a3a\Accessibilit

Стек вызовов Точки останова Параметры исключений Командное окно Окно интерпретации Вывод Список ошибок
```

Рисунок 4 – вывод средней зарплаты мужчин и женщин в консоль (вариант 8)

5. Вывод

В ходе выполнения лабораторной работы была создана программа на языке C# реализующая поставленную задачу. В ходе работы был использована платформа GitHub, предоставляющая возможность работы с системой контроля версий Git. В результате были написаны программы использующие единый интерфейс, но выполняющие разные задачи.

Исходный код программы представлен в приложениях А-В

Приложение А

Variant4.cs

```
using NPOI.SS.UserModel;
using NPOI.XSSF.UserModel;
using System;
using System.Collections.Generic;
using System.IO;

namespace lab3.Variant4
{
    public class GDPDataProcessor
    {
        public List<GDPData> GDPDataList;

        public GDPDataProcessor()
        {
            GDPDataList = new List<GDPData>();
        }

        public void GenerateData()
        {
            Random random = new Random();

            for (int i = 1; i <= 15; i++)
            {
                double gdp = random.Next(1000, 10000);
                double gnp = random.Next(1000, 10000);

                GDPData data = new GDPData(i, gdp, gnp);
                GDPDataList.Add(data);
            }
        }

        public void SaveDataToExcel(string filePath)
        {
            IWorkbook workbook = new XSSFWorkbook();
            ISheet worksheet = workbook.CreateSheet("GDPData");

            // Заголовки столбцов
            IRow headerRow = worksheet.CreateRow(0);
            headerRow.CreateCell(0).SetCellValue("Year");
            headerRow.CreateCell(1).SetCellValue("GDP");
            headerRow.CreateCell(2).SetCellValue("GNP");

            int row = 1;
            foreach (var data in GDPDataList)
            {
                IRow dataRow = worksheet.CreateRow(row);
                dataRow.CreateCell(0).SetCellValue(data.Year);
                dataRow.CreateCell(1).SetCellValue(data.GDP);
                dataRow.CreateCell(2).SetCellValue(data.GNP);

                row++;
            }

            using (FileStream fileStream = new FileStream(filePath, FileMode.Create,
                FileAccess.Write))
            {
                workbook.Write(fileStream, false);
            }
        }

        public void LoadDataFromExcel(string filePath)
        {

```

```

    {
        if (File.Exists(filePath))
        {
            GDPDataList.Clear();

            using (FileStream fileStream = new FileStream(filePath,
FileMode.Open, FileAccess.Read))
            {
                IWorkbook workbook = new XSSFWorkbook(fileStream);
                ISheet worksheet = workbook.GetSheetAt(0);

                for (int row = 1; row <= worksheet.LastRowNum; row++)
                {
                    IRow dataRow = worksheet.GetRow(row);

                    int year = (int)dataRow.GetCell(0).NumericCellValue;
                    double gdp = dataRow.GetCell(1).NumericCellValue;
                    double gnp = dataRow.GetCell(2).NumericCellValue;

                    GDPData data = new GDPData(year, gdp, gnp);
                    GDPDataList.Add(data);
                }
            }
        }
        else
        {
            Console.WriteLine("Файл не найден.");
        }
    }

    public void DisplayData()
    {
        Console.WriteLine("Данные о ВВП и ВВП России за последние 15 лет:");
        Console.WriteLine("Year\tGDP\tGNP");
        foreach (var data in GDPDataList)
        {
            Console.WriteLine($"{data.Year}\t{data.GDP}\t{data.GNP}");
        }
    }

    public void CalculateGrowthRate()
    {
        Console.WriteLine("\nВычисление процента роста/падения ВВП и ВВП:");

        for (int i = 1; i < GDPDataList.Count; i++)
        {
            double currentGDP = GDPDataList[i].GDP;
            double previousGDP = GDPDataList[i - 1].GDP;
            double gdpGrowthRate = ((currentGDP - previousGDP) / previousGDP) *
100;

            double currentGNP = GDPDataList[i].GNP;
            double previousGNP = GDPDataList[i - 1].GNP;
            double gnpGrowthRate = ((currentGNP - previousGNP) / previousGNP) *
100;

            Console.WriteLine($"Год {GDPDataList[i].Year}:");
            Console.WriteLine($"Процент роста/падения ВВП: {gdpGrowthRate}%");
            Console.WriteLine($"Процент роста/падения ВВП: {gnpGrowthRate}%");
        }
    }
}

public class GDPData
{

```

```
public int Year { get; set; }
public double GDP { get; set; }
public double GNP { get; set; }

public GDPData(int year, double gdp, double gnp)
{
    Year = year;
    GDP = gdp;
    GNP = gnp;
}
}
```


Приложение В

Variant8.cs

```
using NPOI.SS.UserModel;
using NPOI.XSSF.UserModel;
using System;
using System.Collections.Generic;
using System.IO;

namespace lab3.Variant8
{
    public class SalaryDataProcessor
    {
        public List<int> Years { get; private set; }
        public List<double> MaleSalaries { get; private set; }
        public List<double> FemaleSalaries { get; private set; }

        public SalaryDataProcessor()
        {
            Years = new List<int>();
            MaleSalaries = new List<double>();
            FemaleSalaries = new List<double>();
        }

        public void GenerateData(int numberOfYears)
        {
            // Генерация случайных данных о медианной заработной плате
            Random random = new Random();

            for (int i = 0; i < numberOfYears; i++)
            {
                int year = DateTime.Now.Year - numberOfYears + i + 1;
                double maleSalary = random.NextDouble() * 50000 + 30000; // Пример
случайных данных
                double femaleSalary = random.NextDouble() * 50000 + 30000; //
Пример случайных данных

                Years.Add(year);
                MaleSalaries.Add(maleSalary);
                FemaleSalaries.Add(femaleSalary);
            }
        }

        public void SaveDataToExcel(string filePath)
        {
            // Сохранение данных в файл Excel
            IWorkbook workbook = new XSSFWorkbook();
            ISheet sheet = workbook.CreateSheet("SalaryData");

            // Создание заголовков столбцов
            IRow headerRow = sheet.CreateRow(0);
            headerRow.CreateCell(0).SetCellValue("Year");
            headerRow.CreateCell(1).SetCellValue("Male Salary");
            headerRow.CreateCell(2).SetCellValue("Female Salary");

            // Запись данных
            for (int i = 0; i < Years.Count; i++)
            {
                IRow dataRow = sheet.CreateRow(i + 1);
                dataRow.CreateCell(0).SetCellValue(Years[i]);
                dataRow.CreateCell(1).SetCellValue(MaleSalaries[i]);
                dataRow.CreateCell(2).SetCellValue(FemaleSalaries[i]);
            }
        }
    }
}
```

```

        // Сохранение файла
        using (FileStream fileStream = new FileStream(filePath, FileMode.Create,
FileAccess.Write))
        {
            workbook.Write(fileStream, false);
        }
    }

    public void LoadDataFromExcel(string filePath)
    {
        Years.Clear();
        MaleSalaries.Clear();
        FemaleSalaries.Clear();

        using (FileStream fileStream = new FileStream(filePath, FileMode.Open,
FileAccess.Read))
        {
            IWorkbook workbook = new XSSFWorkbook(fileStream);
            ISheet sheet = workbook.GetSheetAt(0);

            // Чтение данных
            for (int i = 1; i <= sheet.LastRowNum; i++)
            {
                IRow row = sheet.GetRow(i);
                if (row != null)
                {
                    int year = (int)row.GetCell(0).NumericCellValue;
                    double maleSalary = row.GetCell(1).NumericCellValue;
                    double femaleSalary = row.GetCell(2).NumericCellValue;

                    Years.Add(year);
                    MaleSalaries.Add(maleSalary);
                    FemaleSalaries.Add(femaleSalary);
                }
            }
        }

        public void DisplayData()
        {
            // Вывод данных на экран
            Console.WriteLine("Year\tMale Salary\tFemale Salary");

            for (int i = 0; i < Years.Count; i++)
            {
                Console.WriteLine($"{Years[i]}\t{MaleSalaries[i]}\t{FemaleSalaries[i]}");
            }
        }

        public double CalculateSalaryGrowthRate(List<double> salaries)
        {
            // Вычисление процента роста зарплаты
            if (salaries.Count < 2)
            {
                throw new ArgumentException("Data should contain at least two values
for calculating the growth rate.");
            }

            double firstValue = salaries[0];
            double lastValue = salaries[salaries.Count - 1];

            double growthRate = (lastValue - firstValue) / firstValue * 100;

            return growthRate;
        }
    }
}

```

```
    }  
  
    public double CalculateMaleSalaryGrowthRate()  
    {  
        return CalculateSalaryGrowthRate(MaleSalaries);  
    }  
  
    public double CalculateFemaleSalaryGrowthRate()  
    {  
        return CalculateSalaryGrowthRate(FemaleSalaries);  
    }  
    }  
}
```