МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА - Российский технологический университет»
РТУ МИРЭА

Институт перспективных технологий и индустриального программирования
(ИПТИП)

ОТЧЕТ ПО ПРАКТИЧЕСКИМ РАБОТАМ

по дисциплине

«Технологии создания программного обеспечения»

Практическая работа №2

Выполнил студент группы ЭФМО-02-23 Мурадов Н.Н.

Москва 2023

СОДЕРЖАНИЕ

Задача

1.Даны строковые последовательности *A* и *B*; все строки в каждой последовательности различны, имеют ненулевую длину и содержат только цифры и заглавные буквы латинского алфавита. Получить последовательность всевозможных комбинаций вида «E_A=$E_в$», где E_A - некоторый элемент из *A*, $E_в$ - некоторый элемент из *B*, причем оба элемента оканчиваются цифрой (например, «AF3=D78»). Упорядочить полученную последовательность в лексикографическом порядке по возрастанию элементов *EA*, а при одинаковых элементах *EA* - в лексикографическом порядке по убыванию элементов *EB* (для перебора комбинаций использовать методы SelectMany и Select).

2.Даны последовательности положительных целых чисел *A* и *B*; все числа в последовательности *A* различны. Получить последовательность строк вида «*S*:*E*», где *S* обозначает среднее арифметическое тех чисел из *B*, которые оканчиваются на ту же цифру, что и число *E* - один из элементов последовательности *A* (например, «74:23»); если для числа *E* не найдено ни одного подходящего числа из последовательности *B*, то в качестве *S* указать 0. Расположить элементы полученной последовательности по возрастанию значений найденных сумм, а при равных суммах - по убыванию значений элементов *A*.

3.В организации имеется 3 отдела. В каждом отделе имеется от 3 до 5 сотрудников. Используя группировку по отделу, вывести список сотрудников и средний оклад по каждому отделу. Определите долю суммы окладов всех сотрудников одного отдела в общей сумме окладов по всему предприятию.

4. Дано целое число *K* - код одного из клиентов фитнес-центра. Исходная последовательность содержит сведения о клиентах этого фитнес-центра. Каждый элемент последовательности включает следующие целочисленные поля:

<Код клиента> <Год> <Номер месяца>
<Продолжительность занятий (в часах)>

Для каждого года, в котором клиент с кодом *K* посещал центр, определить месяц, в котором продолжительность занятий данного клиента была наименьшей для данного года (если таких месяцев несколько, то выбирать первый из этих месяцев в исходном наборе; месяцы с нулевой продолжительностью занятий не учитывать). Сведения о каждом годе выводить на новой строке в следующем порядке: наименьшая продолжительность занятий, год, номер месяца. Упорядочивать сведения по возрастанию продолжительности занятий, а при равной продолжительности - по возрастанию номера года. Если данные о клиенте с кодом *K* отсутствуют, то записать в результирующий файл строку «Нет данных».

Указание. Для отбора данных, связанных с клиентом K, использовать метод Where. Затем выполнить группировку по полю «год» и для каждой полученной последовательности выбрать требуемый месяц с помощью сортировки по набору ключей «продолжительность занятий, номер месяца». Обработку особой ситуации, связанной с отсутствием требуемых данных, выполнять с использованием метода DefaultIfEmpty с параметром «Нет данных».

5. Исходная последовательность содержит сведения об абитуриентах. Каждый элемент последовательности включает следующие поля:

<Номер школы> <Год поступления> <Фамилия>

Для каждого года, присутствующего в исходных данных, вывести число различных школ, которые окончили абитуриенты, поступившие в этом году (вначале указывать число школ, затем год). Сведения о каждом годе выводить на новой строке и упорядочивать по возрастанию числа школ, а для совпадающих чисел — по возрастанию номера года.

6. Из последовательности (см. п.5) определить, в какие годы общее число абитуриентов для всех школ было наибольшим и наименьшим, и вывести это число, а также годы, в которые оно было достигнуто (годы упорядочивать по возрастанию, каждое число выводить на новой строке).

7. Исходная последовательность содержит сведения о задолжниках по оплате коммунальных услуг, живущих в 144-квартирном 9-этажном доме. Каждый элемент последовательности включает следующие поля:

<Задолженность> <Фамилия> <Номер квартиры>

Задолженность указывается в виде дробного числа (целая часть — рубли, дробная часть — копейки). В каждом подъезде на каждом этаже располагаются по 4 квартиры. Для каждого из 4 подъездов дома найти трех жильцов с наибольшей задолженностью и вывести сведения о них: задолженность (выводится с двумя дробными знаками), номер подъезда, номер квартиры, фамилия жильца. Считать, что в наборе исходных данных все задолженности имеют различные значения. Сведения о каждом задолжнике выводить на отдельной строке и упорядочивать по убыванию размера задолженности (номер подъезда при сортировке не учитывать). Если в каком-либо подъезде число задолжников меньше трех, то включить в полученный набор всех задолжников этого подъезда.

8. Даны последовательности A и B, включающие следующие поля: *A: категория, артикул товара, страна производитель; B: артикул товара, цена, название магазина.* Для каждой категории товаров определить количество магазинов, предлагающих товары данной категории, а также количество стран, в которых произведены товары данной категории, представленные в магазинах (вначале выводится количество магазинов, затем название

категории, затем количество стран). Если для некоторой категории не найдено ни одного товара, представленного в каком- либо магазине, то информация о данной категории не выводится. Сведения о каждой категории выводить на новой строке и упорядочивать по убыванию количества магазинов, а в случае одинакового количества — по названиям категорий в алфавитном порядке.

9. Даны последовательности A, B и C, включающие следующие поля: *A: улица, код потребителя, год рождения; B: страна производителя, категория, артикул товара; C: артикул товара, код потребителя, название магазина.* Для каждого года рождения из A определить страну, в которой было произведено максимальное количество товаров, приобретенных потребителями этого года рождения (вначале выводится год, затем название страны, затем максимальное количество покупок). Если для некоторой пары «год-страна» отсутствует информация о проданных товарах, то эта пара не обрабатывается (в частности, если потребители некоторого года рождения не сделали ни одной покупки, то информация об этом годе не выводится). Если для какого-либо года рождения имеется несколько стран с наибольшим числом приобретенных товаров, то выводятся данные о первой из таких стран (в алфавитном порядке). Сведения о каждом годе выводить на новой строке и упорядочивать по убыванию номера года.

Решение

Листинг кода с 1 по 9 задание.

```
using System;
using System.Linq;
using System.Collections.Generic;
using System.Runtime.Versioning;
using Bogus;
using Pract2N3;
using Pract2N4;
using Pract2N56;
using Pract2N7;
using Pract2N8;
using Pract2N9;

namespace Pract2N3 {
   public class Department {

      public string name { get; set; }
      public List<Division> div { get; set; }

      public Department(Faker fakerRu, Random rnd){
         div = new();
         name = fakerRu.Commerce.Department();
         for (int i = 0; i < 3; i++) {
            List<Person> pers = new();
            div.Add(new Division{name = fakerRu.Commerce.ProductName(), pers =
pers});
            for (int i1 = 0; i1 < rnd.Next(3, 6); i1++) {
               pers.Add(new Person{fio = fakerRu.Name.FullName(), salary =
rnd.Next(10000, 1000000)});
            }
         }
      }

      public override string ToString() {
         return $"Department: name = {name} | div = ({string.Join(", ", div)})";
      }
   }

   public class Division {

      public string name { get; set; }
      public List<Person> pers { get; set; }

      public override string ToString() {
         return $"\n\tDivision:  name = {name} | pers = ({string.Join(", ", pers)})";
      }
   }

   public class Person {
```

```
            public string fio { get; set; }
            public int salary { get; set; }

            public override string ToString() {
                return $"\n\t\tPerson: fio = {fio} | salary = {salary}";
            }
        }
    }

    namespace Pract2N4 {
        public class FitnessCenter {

            public string name { get; set; }
            public List<MonthDur> monthDurs { get; set; }

            public FitnessCenter(Faker fakerRu, Random rnd) {
                monthDurs = new();
                name = fakerRu.Commerce.Department();
                monthDurs.Add(new MonthDur{idClient = 0, year = 2003, month = 2, duration =
1});
                monthDurs.Add(new MonthDur{idClient = 0, year = 2003, month = 3, duration =
1});
                monthDurs.Add(new MonthDur{idClient = 0, year = 2003, month = 4, duration =
1});
                for (int i = 1, ni = 8; i < rnd.Next(30, 41); i++) {
                    if(ni == i) ni += rnd.Next(8, 12);
                    int y = rnd.Next(2005, 2023), m = rnd.Next(1, 13);
                    if(monthDurs.Any(n => n.year == y && n.month == m && n.idClient == ni))
continue;
                    monthDurs.Add(new MonthDur{idClient = ni, year = y, month = m, duration =
rnd.Next(0, 721)});
                }
            }

            public override string ToString() {
                return $"FitnessCenter: name = {name} | monthDurs = ({string.Join(", ",
monthDurs)})";
            }
        }

        public class MonthDur {

            public int idClient { get; set; }
            public int year { get; set; }
            public int month { get; set; }
            public int duration { get; set; }

            public override string ToString() {
                return $"\n\tMonthDur: idClient = {idClient} | duration = {duration} | year =
{year} | month = {month}";
            }
```

```
            public string ToStringWI() {
                return $"\n\tMonthDur: duration = {duration} | year = {year} | month = {month}";
            }
        }
    }

    namespace Pract2N56 {
    public class College {

        public string name { get; set; }
        public List<Applicant> applicants { get; set; }

        public College(Faker fakerRu, Random rnd) {
            applicants = new();
            name = "Колледж №" + rnd.Next(30, 1000000);
            for (int i = 1, ni = 8; i < rnd.Next(30, 41); i++) {
                if(ni == i) ni += rnd.Next(8, 12);
                int y = rnd.Next(2005, 2023);
                string lN = fakerRu.Name.LastName();
                if(applicants.Any(n => n.year == y && n.numSch == ni && n.lastName ==
lN)) continue;
                applicants.Add(new Applicant{numSch = ni, year = y, lastName = lN});
            }
        }

        public override string ToString() {
            return $"College: name = {name} | applicants = ({string.Join(", ", applicants)})";
        }
    }

    public class Applicant {

        public int numSch { get; set; }
        public int year { get; set; }
        public string lastName { get; set; }

        public override string ToString() {
            return $"\n\tApplicant: numSch = {numSch} | year = {year} | lastName =
{lastName}";
        }
    }
    }

    namespace Pract2N7 {
    public class House {

        public string name { get; set; }
        public List<Debt> debts { get; set; }

        public House(Faker fakerRu, Random rnd) {
            debts = new();
            name = "House №" + rnd.Next(30, 1000000);
```

```
                for (int i = 0, ni = 0; i < rnd.Next(30, 41); i++) {
                    ni = rnd.Next(1, 145);
                    int t = rnd.Next(1, 1000000);
                    string lN = fakerRu.Name.LastName();
                    if(debts.Any(n => n.numFlat == ni)) continue;
                    debts.Add(new Debt{numFlat = ni, total = t, lastName = lN});
                }
            }

            public override string ToString() {
                return $"House: name = {name} | debts = ({string.Join(", ", debts)})";
            }
        }

        public class Debt {

            public int numFlat { get; set; }
            public int total { get; set; }
            public string lastName { get; set; }

            public override string ToString() {
                return $"\n\tDebt: numFlat = {numFlat} | total = {total} | lastName =
{lastName}";
            }
        }
    }

    namespace Pract2N8 {
    public class HoldingA {

            public string name { get; set; }
            public List<A> listA { get; set; }
            public List<B> listB { get; set; }

            public HoldingA(Faker fakerRu, Random rnd) {
                listA = new();
                listB = new();
                name = fakerRu.Company.CompanyName();
                string madeIn = fakerRu.Address.Country();
                for (int i = 0, ni = 0, st = 0; i < rnd.Next(30, 41); i++) {
                    ni = rnd.Next(1, 10000000);
                    if(st == i) {
                        st += rnd.Next(5, 12);
                        madeIn = fakerRu.Address.Country();
                    }
                    if(listA.Any(n => n.itemNum == ni) || listB.Any(n => n.itemNum == ni))
continue;
                    listA.Add(new A{category = fakerRu.Commerce.Categories(1).First(),
itemNum = ni, madeIn = madeIn});
                    listB.Add(new B{itemNum = ni, price = rnd.Next(1, 1000000), nameStore =
fakerRu.Commerce.Department()});
                }
```

```
            }

            public override string ToString() {
                return $"HoldingA: name = {name} | listA = ({string.Join(", ", listA)}) | listB =
({string.Join(", ", listB)})";
            }
        }

        public class A {

            public string category { get; set; }
            public int itemNum { get; set; }
            public string madeIn { get; set; }

            public override string ToString() {
                return $"\n\tA: category = {category} | itemNum = {itemNum} | madeIn =
{madeIn}";
            }
        }

        public class B {

            public int itemNum { get; set; }
            public int price { get; set; }
            public string nameStore { get; set; }

            public override string ToString() {
                return $"\n\tB: itemNum = {itemNum} | price = {price} | nameStore =
{nameStore}";
            }
        }
    }

    namespace Pract2N9 {
    public class HoldingB {

            public string name { get; set; }
            public List<A9> listA { get; set; }
            public List<B9> listB { get; set; }
            public List<C9> listC { get; set; }

            public HoldingB(Faker fakerRu, Random rnd) {
                listA = new();
                listB = new();
                listC = new();
                name = fakerRu.Company.CompanyName();
                string madeIn = fakerRu.Address.Country();
                for (int i = 0, ni = 0, st = 0; i < rnd.Next(30, 41); i++) {
                    ni = rnd.Next(1, 10000000);
                    if(st == i) {
                        st += rnd.Next(5, 12);
                        madeIn = fakerRu.Address.Country();
```

```
                    }
                    if(listB.Any(n => n.itemNum == ni)) continue;
                    listA.Add(new A9{street = fakerRu.Address.StreetName(), clientId = st, year =
rnd.Next(2005, 2023)});
                    listB.Add(new B9{category = fakerRu.Commerce.Categories(1).First(),
itemNum = ni, madeIn = madeIn});
                    listC.Add(new C9{itemNum = ni, clientId = st, nameStore =
fakerRu.Commerce.Department()});
                }
            }

            public override string ToString() {
                return $"HoldingB: name = {name} | listA = ({string.Join(",  ", listA)}) | listB =
({string.Join(", ", listB)}) | listC = ({string.Join(", ", listC)})";
            }
        }

        public class A9 {

            public string street { get; set; }
            public int clientId { get; set; }
            public int year { get; set; }

            public override string ToString() {
                return $"\n\tA9: street = {street} | clientId = {clientId} | year = {year}";
            }
        }

        public class B9 {

            public string category { get; set; }
            public int itemNum { get; set; }
            public string madeIn { get; set; }

            public override string ToString() {
                return $"\n\tB9: category = {category} | itemNum = {itemNum} | madeIn =
{madeIn}";
            }
        }

        public class C9 {

            public int itemNum { get; set; }
            public int clientId { get; set; }
            public string nameStore { get; set; }

            public override string ToString() {
                return $"\n\tC9: itemNum = {itemNum} | clientId = {clientId} | nameStore =
{nameStore}";
            }
        }
    }
```

```
public class Pract2 {

    public Faker fakerRu = new("ru");
    public Random rnd = new();

    public void run() {
        Console.WriteLine("Hello World Pract2");
        num9();
    }

    public void num1() {
        Console.WriteLine("num1:");
        List<string> A = new List<string>() {"AXS1", "2TER", "B2GF", "LK3IK1",
"1K3IK1"},
            B = new List<string>() {"LK3IK1", "1K3IK1", "1K4IK1", "B3GF", "B4GF"};
        var rez = A.SelectMany(el => B, (el, el1) => new {nA = el, nB = el1}).Where(n =>
char.IsDigit(n.nA.Last()) && char.IsDigit(n.nB[^1]));
        foreach (var n in rez.OrderBy(n => n.nA).ThenByDescending(n => n.nB).Select(n
=> n.nA + "=" + n.nB))
            Console.WriteLine(n);
        Console.WriteLine("Конец num1\n");
    }

    public void num2() {
        Console.WriteLine("num2:");
        List<int> B = new() { 204, 178, 4345, 5435, 565, 676, 184 },
            A = new() { 0, 1, 22, 45, 56, 4 };
        var rez = A.SelectMany(el => B, (el, el1) => new {elA = el, elB = el1}).GroupBy(o
=> o.elA)
            .Select(n => new {S = n.Where(n => n.elA % 10 == n.elB %
10).DefaultIfEmpty(new {elA = 0, elB = 0}).Average(n1 => n1.elB),
            E = n.Key}).OrderBy(n => n.S).ThenByDescending(n => n.E).Select(n => n.S +
":" + n.E);
        foreach (var n in rez)
            Console.WriteLine(n);
        Console.WriteLine("Конец num2\n");
    }

    public void num3() {
        Console.WriteLine("num3:");
        // Console.WriteLine("Dep: " + dep + "\nQuery: ");
        Department dep = new(fakerRu, rnd);
        float sym = dep.div.Sum(n => n.pers.Sum(n => n.salary));
        var rez = dep.div.GroupBy(o => o)
            .Select(n => new {Name = n.Key.name, Avg = n.Key.pers.Average(n =>
n.salary), Ratio = n.Key.pers.Sum(n => n.salary) / sym, Pers = string.Join(", ", n.Key.pers)});
        foreach (var n in rez)
            // Console.WriteLine(n);
            Console.WriteLine(string.Join(",  ", n));
        Console.WriteLine("Конец num3\n");
    }
```

```
        public void num4() {
            Console.WriteLine("num4:");
            FitnessCenter center = new(fakerRu, rnd);
            int K = 2;
            if(fakerRu.Random.Bool()) K = center.monthDurs.ElementAt(rnd.Next(0,
center.monthDurs.Count)).idClient;
            var rez = center.monthDurs.Where(n => n.duration != 0 && n.idClient == K)
                .GroupBy(o => o.year, o => o, (k, v) => v.MinBy(n => n.duration))
                .OrderBy(n => (n.duration, n.year)).Select(n =>
n.ToStringWI()).DefaultIfEmpty("Нет данных");
            Console.WriteLine(string.Join(",  ", rez));
            Console.WriteLine($"K = {K}");
            Console.WriteLine(center);
            Console.WriteLine("Конец num4\n");
        }

        public void num5() {
            Console.WriteLine("num5:");
            College college = new(fakerRu, rnd);
            var rez = college.applicants
                .GroupBy(o => o.year, o => o, (k, v) => new {k = k, v = v.Count()})
                .OrderBy(n => (n.v, n.k));
            foreach (var n in rez.Select(n => $"countSch = {n.v}, year = {n.k}"))
                Console.WriteLine(string.Join(",  ", n));
            Console.WriteLine(college);
            Console.WriteLine("Конец num5\n");
        }

        public void num6() {
            Console.WriteLine("num6:");
            College college = new(fakerRu, rnd);
            var rez = college.applicants
                .GroupBy(o => o.year, o => o, (k, v) => new {year = k, count = v.Count()});
            var l = new [] {rez.MinBy(n=>n.count), rez.MaxBy(n=>n.count)}.ToList();
            var rez1 = l.OrderBy(n => n.year);
            foreach (var n in rez1)
                Console.WriteLine(string.Join(",  ", n));
            Console.WriteLine(college);
            Console.WriteLine("Конец num6\n");
        }

        public int getNum(int flat) {
            return flat == 0 ? 16 : flat;
        }

        public void num7() {
            Console.WriteLine("num7:");
            House house = new(fakerRu, rnd);
            var rez = house.debts
                .Select(n => new {total = n.total, numEntr = ((getNum(n.numFlat % 16) - 1) / 4 +
1), numFlat = n.numFlat, lastName = n.lastName})
```

```
                .GroupBy(n => n.numEntr, n => n, (k, v) => v.OrderByDescending(n=>n.total)
                .Take(3).Select(n => new {total = n.total.ToString("#.00"), numEntr = n.numEntr,
numFlat = n.numFlat, lastName = n.lastName}));
            foreach (var n in rez)
                Console.WriteLine(string.Join(",  ", n));
            Console.WriteLine(house);
            Console.WriteLine("Конец num7\n");
        }

        public void num8() {
            Console.WriteLine("num8:");
            HoldingA holding = new(fakerRu, rnd);
            var rez = holding.listA.Join(holding.listB,
                a => a.itemNum,
                b => b.itemNum,
                (a, b) => new {itemNum = b.itemNum, category = a.category, madeIn =
a.madeIn, nameStore = b.nameStore}
            ).GroupBy(n => n.category, n => n, (k, v) => new{countStore = v.GroupBy(n =>
n.nameStore).Count(), category = k, countMadeIn = v.GroupBy(n => n.madeIn).Count()})
                .OrderByDescending(n => n.countStore).ThenBy(n => n.category);
            foreach (var n in rez)
                Console.WriteLine(string.Join(",  ", n));
            Console.WriteLine(holding);
            Console.WriteLine("Конец num8\n");
        }

        public void num9() {
            Console.WriteLine("num9:");
            HoldingB holding = new(fakerRu, rnd);
            var rez = holding.listA.Join(holding.listC,
                a => a.clientId,
                c => c.clientId,
                (a, c) => new {aEl = a, cEl = c}
            ).Join(holding.listB,
                ac => ac.cEl.itemNum,
                b => b.itemNum,
                (ac, b) => new {aEl = ac.aEl, b = b, cEl = ac.cEl}
            ).GroupBy(n => n.aEl.year, n => n, (k, v) => v.GroupBy(n=>n.b.madeIn, n=>n, (k1,
v1) => new {year = k, madeIn = k1, count = v1.Count()}).MaxBy(n=>n.count))
                .OrderByDescending(n => n.year).ThenBy(n => n.madeIn);
            foreach (var n in rez)
                // foreach (var n1 in n)
                    Console.WriteLine(string.Join(",  ", n));
            Console.WriteLine(holding);
            Console.WriteLine("Конец num9\n");
        }
    }
```

Результаты

Результаты выполнения с 1 по 9 задачи представлены на Рисунке 1.

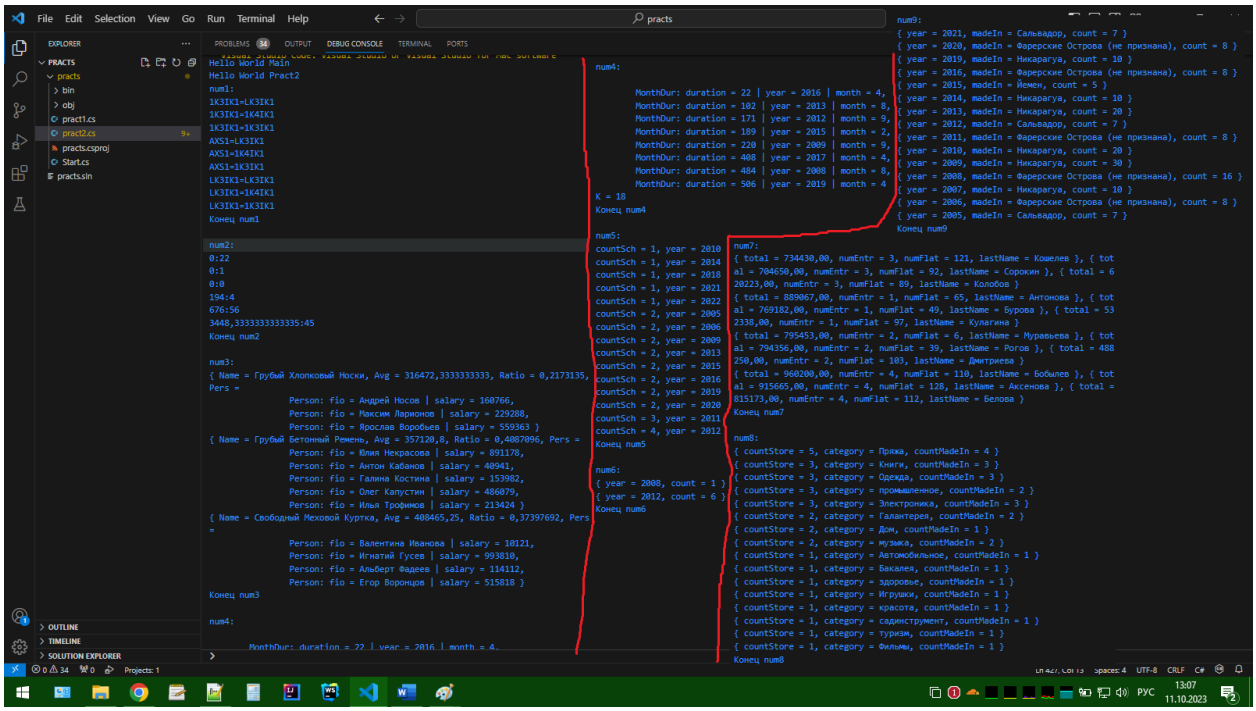

Рисунок 1 – Результаты компиляции с 1 по 9 заданий